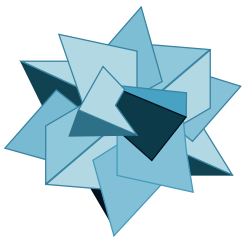

Phing User Guide

Andreas Aderhold <andi@binarycloud.com>
Alex Black <enigma@turingstudio.com>
Manuel Holtgrewe <grin@gmx.net>
Hans Lellelid <hans@xmpl.org>
Michiel Rook <mrook@php.net>
Johan Persson <johan162@gmail.com>

Phing User Guide

by Andreas Aderhold, Alex Black, Manuel Holtgrewe, Hans Lellelid, Michiel Rook, and Johan Persson

Publication date 2013-08-15

Preface

PHing **I**s **N**ot **GN**U make; it's a PHP project build system or build tool based on Apache Ant. You can do anything with it that you could do with a traditional build system like GNU make, and its use of simple XML build files and extensible PHP "task" classes make it an easy-to-use and highly flexible build framework. Features include running PHPUnit and SimpleTest unit tests (including test result and coverage reports), file transformations (e.g. token replacement, XSLT transformation, Smarty template transformations), file system operations, interactive build support, SQL execution, CVS/SVN operations, tools for creating PEAR packages, documentation generation (DocBlox, PhpDocumentor) and much more.

If you find yourself writing custom scripts to handle the packaging, deploying, or testing of your applications, then we suggest looking at the Phing framework. Phing comes packaged with numerous out-of-the-box operation modules (tasks), and an easy-to-use OO model for adding your own custom tasks.

Phing provides the following high level features:

- Easy to read XML buildfiles
- Rich set of predefined tasks
- Easily extendible via PHP classes
- Platform-independent: works on UNIX, Windows, MacOSX
- No required external dependencies apart from a working PHP5 installation
- Built & optimized for ZendEngine2/PHP5

Chapter 1. About this book

1.1. Authors

- Andreas Aderhold, andi@binarycloud.com
- Alex Black, enigma@turingstudio.com
- Manuel Holtgrewe, grin@gmx.net
- Hans Lellelid, hans@xmpl.org
- Michiel Rook, mrook@php.net
- Johan Persson, johan162@gmail.com

1.2. Copyright

1.3. License

1.4. DocBook

This book is written using the DocBook 5 XML markup (see DocBook Project [http://docbook.sourceforge.net/]). The main advantage with DocBook is that it is a single source but multiple outputs. These document sources can be rendered into many possible output formats such as (X)HTML, PDF, EPub, Webhelp, RTF, Text and many more. Another advantage, inherit with the text based XML format, is that the document sources are all completely text based written using UTF-8 encoding. Only a plain text editor is requires to extend or edit this documentation. However, XML tends to be quite verbose and even if a plain text editor technically is all you need the actual entering of text will be made much easier with custom XML editor. These editors can be used to hide the XML tags

and do auto-completion and on-the-fly validation to make sure that what you write is a valid DocBook document.

To work with the documentation we recommend to use one of the free XML/DocBook aware editors available. For example

- Emacs with the nXML mode (see nXML mode [http://www.thaiopensource.com/nxml-mode/])
- Serna Free, (Free of charge) A Java based XML editor with extended support for DocBook5 (see Serna Free - Open Source XML Editor [http://www.syntext.com/products/serna-free/])
- XMLMind XML Editor, Personal Edition (Free of charge), A java based XML editor with extended support for DocBook5 (see XMLMind Personal Edition [http://www.xmlmind.com/xmleditor/persoedition.html])

The sources for the documentation are included under the `docs/docbook5` directory. The DocBook sources are split into several files in order to make it more maintainable using the XML standard XInclude (see XML Inclusions (XInclude) Version 1.0 [http://www.w3.org/TR/xinclude/]).

For the writing of the book only a subset of all available DocBook elements are used as shown in Section 1.4.2, “DocBook v5 elements used in the manual and their meaning”

1.4.1. Building the documentation

In order to build the documentation it is necessary to have the DocBook5 XSL stylesheets installed together with "`xsltproc`" which is used to transform the source into various output formats. In addition, to build the versions (either HTML or PDF) that supports highlighting of included source (within the `<programlisting>` element) the Saxon 6.5.5 XSL processor must be used. This is necessary since the syntax highlighting in DocBook is based on a Java extension (`xslthl-2.x.x`) which requires a Java based processor (such as Saxon).

All DocBook sources are structured in a tree under `docs/docbook5`. The top level is the language of the manual. As of this writing only an English manual is available and hence the only top level directory available is "`en`". Under this directory the following structure applies (also for any new language translation that is added):

```
### scripts
### source
#   ### appendixes
#   ### chapters
### stylesheets
    ### css
    #   ### img
    ### xsl
        ### images
```

All document sources are stored under the subdirectory "`source`" and the master document is aptly named "`master.xml`". This document pulls in all chapters and appendixes in the right order. For example, new tasks added should normally be documented in the "`appendix/optionaltasks.xml`" file. Look at the existing tasks and follow the same structure.

Important

In order to get highlighting to work both the "`xslthl-2.x.x.jar`" package must be installed as well as Saxon 6.5.x. The jar file must be installed somewhere in the `CLASSPATH` , for example "`/usr/share/java`" if you run this on Linux. The xslthl package is available on SourceForge, please see XSLT syntax highlighting [http://sourceforge.net/projects/xslthl/]

The customized stylesheets used are stored under "`stylesheets`" which uses one sub-folder for the customized XSL stylesheets (responsible for the transformation from DocBook to the chosen output format) and one sub-folder for the CSS stylesheets used to give the generated HTML documents there "*look&feel*".

Finally the "`scripts`" directory stores utility scripts. As of this writing it only contains one utility script "`hlsaxon`" which sets up a suitable environment to call the Saxon translator (a Java based XSL procesor) with highlighting enabled. In this script the path to the DocBook installed stylesheets must be correct and depending on your system they might need updating. Mutatis mutandis.

In order to drive the transformation a Phing build script is available in the docbook root, `build.xml`. The build script supports the following public targets

```
all*         Builds all available targets (default)
chunk        Builds the chunked HTML
clean        Removes all output files
epub         Builds the EPUB version
hlhtml       Builds the HTML version with syntax highlight
hlpdf        Builds the PDF version with syntax highlight
html         Builds the HTML version
htmlfancy    Builds the HTML version with an alternative styling for screen output
pdf          Builds the PDF version
validate     Validates all sources against the DocBook5 grammaar
```

All generated output is stored under the directory "`output`" (which is created if it doesn't exist) with a subdirectory corresponding to the name of the chosen output format.

1.4.2. DocBook v5 elements used in the manual and their meaning

To keep things simple the manual uses only a small subset of all available elements in the DocBook schema. This makes it fairly easy to quickly get up to speed with adding and editing the manual. It also helps to keep the look&feel consistent and makes the writing of the CSS and XSL stylesheets a little bit easer.

The following list shows the supported elements and how they should be used in the manual

`<chapter>`, `<appendix>` This is the top element for each chapter and appendix in the manual. Each <chapter> or <appendix> must also have a title.

Table 1.1: *Required attributes*

Attribute	Value	Description
`xmlns`	`http://docbook.org/ns/docbook`	Name space for DocBook. Always needed.
`xmlns:xi`	`http://www.w3.org/2001/XInclude`	Name space for `XInclude`. Needed since we use `XInclude` to split the manual into different files.
`xmlns:xlink`	`http://www.w3.org/1999/xlink`	Name space for `xlink`. Needed sine we make use of link and xref elements to link to other sites and cross references within the manual.
`version`	`5.0`	Versions of DocBook. Always needed
`xml:id`	`app.XXX` , `ch.XXX`	The id for the chapter or the appendix. Used in other part of the manu-

Attribute	Value	Description
		al to refer to this chapter/appendix with an <xref> element.

Table 1.2: *Required nested elements*

Element	Value
<title>	The title of the chapter/appendix

Example:

```
<appendix xmlns="http://docbook.org/ns/docbook"
    xmlns:xi="http://www.w3.org/2001/XInclude"
 xmlns:xlink="http://www.w3.org/1999/xlink"
    version="5.0"
    xml:id="app.coretasks">
 <title>Core tasks</title>
...
</appendix>
```

<sectN>

The section tags divides each chapter and appendix into logical parts. Each task description must be contained in a <sect1> element and each example section for the task must be contained within a <sect2> element. Depending on the description needed for each task additional <sect2> may be added as needed to make the text logically structured. If needed, a further nesting level may be used by using <sect3> elements within each <sect2> element. No deeper nestings than <sect3> should ever be used.

Each top level section must have the xml:id attribute which is used to reference the section from other parts of the document. Each section must have a nested title element.

Table 1.3: *Required attributes*

Attribute	Value	Description
role	taskdef	This is only used and required for <sect1> elements for task description. This role is not currently used in the any of the XSL sheets. This is for future use.
xml:id	Name of section	The id for task definition should be the same as the task name for task description. For other sections the id should be a logical name that descrobes the content.

Table 1.4: *Required nested elements*

Element	Value
<title>	The title of the section

Example:

```
<sect1 role="taskdef" xml:id="AdhocTaskdefTask">
    <title>AdhocTaskdefTask</title>
...
</sect1>
```

<para>

Division between paragraphs in flowing text.

`<screen>` Used to mark command lines and multi-line computer output. For inline screen output use the <literal> element

`<programlisting>` Used for all PHP and XML program listings in the manual. Please note that this tag should not be used for command lines as entered in a terminal. Use the `<screen>` element for this.

Note: Remember to write all opening '<' as `&lt;`

Table 1.5: *Required attributes*

Attribute	Value	Description
`language`	`php`, `xml`	The language attribute should indicate what programming language the programlisting contains. This is used to control what syntax highlighting should be used.

Example:

```
<programlisting language="xml">
  &amp;lt;append
   destFile="${process.outputfile}">
  &amp;lt;filterchain>
    &amp;lt;xsltfilter style="${process.stylesheet}">
      &amp;lt;param name="mode" expression="${process.xslt.mode}"/>
    &amp;lt;/xsltfilter>
  &amp;lt;/filterchain>
  &amp;lt;filelist dir="book/" listfile="book/PhingGuide.book"/>
&amp;lt;/append></programlisting>
```

`<acronym>` Used to indicate acronym in running text

`<literal>` Used to indicate literal names in running text such as program variables, name of attributes, XML-elements etc.

`<filename>` Used to indicate a file- or directory name in running text.

Table 1.6: *Required attributes*

Attribute	Value	Description
role	dir	Used when the filename is a directory

Example:

```
<filename role="dir">/etc/php5</filename>
```

`<link>` Used to include a URL link to other sites or documents outside the manual.

Table 1.7: *Required attributes*

Attribute	Value	Description
`xlink:href`	URL Link	The link to an external reference

Example:

```
<link xlink:href="http://qbnz.com/highlighter/"
>GeSHi Homepage</link>
```

`<xref>`

A link to another part of the document. When the link is generated in the rendered document the name of the section, chapter or appendix that the link refers to is included literal.

Table 1.8: *Required attributes*

Attribute	Value	Description
`xlink:href`	Internal reference to an ID element	Internal links must be prefixed with a '#' character.

Example:

```
<xref xlink:href="#ch.projcomponents"/>
```

`<table>`

The CALS model for table should be used. The generated rendered version will be styled by the CSS stylesheet automatically. For this to work as expected for the required attribute for a task the columns needs to have the following names (they are used in the CSS sheets). The column width specified is not important since that will be overridden by the CSS stylesheets.

```
...
<colspec colname="name" colnum="1" colwidth="1.5*"/>
<colspec colname="type" colnum="2" colwidth="0.8*"/>
<colspec colname="description" colnum="3" colwidth="3.5*"/>
<colspec colname="default" colnum="4" colwidth="0.8*"/>
<colspec colname="required" colnum="5" colwidth="1.2*"/>
...
```

A CALS model table should have the following required nested elemenets. For more information on more advanced CALS formatting such as joining rows or columns please see Chapter 30. Tables [http://www.sagehill.net/docbookxsl/CellSpans.html] in Bob Stayton's book "DocBook XSL: The Complete Guide - 4th Edition" [http://www.sagehill.net/docbookxsl/]

Table 1.9: *Required nested elements*

Attribute	Description
title	The descriptive title for the table.
tgroup	Groups a set of columns together
colspec	Defines the sizing of the table
thead	Header row for table
tbody	Body of table

Example:

```
<table>
    <title>Required attributes</title>
    <tgroup cols="3">
        <colspec colname="attribute"   colnum="1"
                 colwidth="1.0*"/>
        <colspec colname="value"       colnum="2"
                 colwidth="1.0*"/>
        <colspec colname="description" colnum="3"
                 colwidth="1.0*"/>
        <thead>
```

```
            <row>
                <entry>Attribute</entry>
                <entry>Value</entry>
                <entry>Description</entry>
            </row>
        </thead>
        <tbody>
            <row>
                <entry>...</entry>
                <entry>...</entry>
                <entry>...</entry>
            </row>
            <row>
                <entry>...</entry>
                <entry>...</entry>
                <entry>...</entry>
            </row>
        </tbody>
    </tgroup>
</table>
```

`<emphasis role="bold">` Should only be used when certain effects in flowing text are wanted that warrents the text to be rendered in a bold style to be shown as emphasised.

Example:

```
<emphasis role="bold">PH</emphasis>ing <emphasis
role="bold">I</emphasis>s <emphasis
role="bold">N</emphasis>ot <emphasis
role="bold">GN</emphasis>U make;
```

The above example will then be rendered as: "**PH**ing **I**s **N**ot **GN**U make;"

`<application>` This tag is used to indicate the name of a application. The line between a command (marked with `<literal>`) and an application is not cut in stone but an application is usually a complex computer program with its own user interface. Examples of what we would mark as applications are "Emacs", "OpenOffice", "MatLab" etc.

This element is rarely used.

Chapter 2. Introduction

2.1. What Phing Is

Phing is a project build system based on Apache ant (See ant). You can do anything with Phing that you could do with a traditional build system like Gnu make (See gnumake), and Phing's use of simple XML build files and extensible PHP task classes make it an easy-to-use and highly flexible build framework.

Because Phing is based on Ant, parts of this manual are also adapted from the ant manual (see ant). We are extremely grateful to the folks in the Ant project for creating (and continuing to create) such an inspiring build system model, and for the open-source licensing that makes it possible for us to learn from each other and build increasingly better tools.

2.2. Phing & Binarycloud: History

Phing was originally a subproject of Binarycloud. Binarycloud is a highly engineered application framework, designed for use in enterprise environments. Binarycloud uses XML extensively for storing metadata about a project (configuration, nodes, widgets, site structure, etc.). Because Binarycloud is built for PHP, performing extensive XML processing and transformations on each page request is an unrealistic proposition. Phing is used to "compile" the XML metadata into PHP arrays that can be processed without overhead by PHP scripts.

Of course, XML compilation is only one of many ways that Binarycloud uses the Phing build system. The Phing build system makes it possible for you to:

- Build multi language pages from one source tree,
- Centralize metadata (e.g. your data model) in one XML file and generate several files from that XML with different XSLT.

It the beginning, Binarycloud used the GNU make system; however, this approach had some drawbacks: The space-before-tab-problem in makefiles, the fact that it is only natively available for Unix systems etc. So, the need for a better build system arose. Due to it's XML build files and modular design, Apache Ant was a logical choice. The problem was that Ant is written in Java, so you need to install a JVM on your computer to use it. Besides the need for yet another interpreter (i.e. besides PHP), there was also legal/ideological conflict in requiring a commercial JVM (there were problems with Ant on JVMs other than Sun's) for an LGPL'd Binarycloud.

So, the development of Phing began. Phing is a build system written in PHP and uses the ideas of Ant. The first release was designed & developed simultaneously, and thus not very sophisticated. This original system was quickly pushed to its limits and the need for a better Phing became a priority. Andreas Aderhold, who was responsible for Phing/r1, designed and wrote much of the Phing/r2 that followed. Phing/r2 became the Phing-1.0 that exists today for PHP4.

The current development version of Phing 2.x, requires PHP5 (at least 5.2.x) and makes use of many of the available features in PHP5.2 to achieve a high degree of modularization, code efficiency as well as stability and testability.

2.3. How Phing Works

Phing uses XML buildfiles that contain a description of the things to do. The buildfile is structured into targets that contain the actual commands to perform (e.g. commands to copy a file, delete a directory,

perform a DB query, etc.). So, to use Phing, you would first write your buildfile and then you would run phing, specifying the target in your buildfile that you want to execute.

```
% phing -f mybuildfile.xml mytarget
```

By default Phing will look for a buildfile named `build.xml` (so you don't have to specify the buildfile name unless it is not `build.xml`) and if no target is specified Phing will try to execute the default target, as specified in the `<project>` tag.

In the same way as traditional make files (but without most of the traditional drawbacks) targets can have dependencies. They can depend on both other targets as well as other files.

2.4. Cool, so how can I help?

Phing is under active development and there are many things to be done. The project will also welcome non-coders to help keep the documentation up to date. If you don't already know about DocBook participåating in the documentation is a great opportunity to get experience!

To get involved start by doing the following:

- Read this manual to understandPhing ;-)
- Go to http://phing.tigris.org and subscribe to the `Phing dev` mailing list (this is usually a low volume, high quality mailing list)
- Visit the Phing website (http://www.phing.info/) [http://www.phing.info/] and look for open bugs / tickets
- ... and of course, start to actively participate in the development by forking the repository (see below)

2.4.1. Participating in the development

As of 1 January 2012 all Phing development is based on Git and the project is hosted at GitHub (https://github.com/)

In order to participate in the development you will only need to follow three basic steps

1. Register a free account at GitHub [https://github.com/]
2. Clone the Official Git repository [https://github.com/phingofficial/phing]
3. Read up on the (very well written) documentation at GitHub on how to setup your own repository and do things like cloning an existing repository and creating pull requests asking the official Phing maintainers to take in your proposed additions/changes.

The chances to have a change set accepted greatly increases if you adhere to the following recommendations

- Follow the naming and coding principle used by Phing
- Make sure you have added documentation for *all* your additions, including examples.
- Make sure you have added unit-test code as needed
- Be polite in all communication!

Note

If you have not worked with Git before and are coming from subversion there is a bit of re-adjustment needed. Fortunately there are several *SVN-To-Git* re-learning guides available (for example http://git.or.cz/course/svn.html which might make the initial transition easier.

However, it is probably best to forget about your mental picture on Subversion and realize that Git is a different animal. So trying to think of everything in terms of Subversion is not really helpful in the long run. You should therefore take the time to read the (free!) book "Pro Git", by Scott Chacon available from http://progit.org/.

Chapter 3. Setting-up Phing

The goal of this chapter is to help you obtain and correctly setup and execute Phing on your operating system. Once you setup Phing properly you shouldn't need to revisit this chapter, unless you're re-installing or moving your installation to another platform.

3.1. System Requirements

To use Phing you must have installed PHP version 5.2 or above compiled `--with-libxml2`, as well as `--with-xsl` if you want to make use of advanced functionality.

For more information on PHP and the required modules see the PHP [php] [Bibliography.html#php] website. For a brief list of software dependencies see below.

3.1.1. Operating Systems

Designed for portability from the get go, Phing runs on all platforms that run PHP. However some advanced functionality may not work properly or is simply ignored on some platforms (i.e. `chmod` on the Windows platform).

To get the most out of Phing, a Unix style platform is recommended. Namely: Linux, FreeBSD, OpenBSD, etc.

3.1.2. Software Dependencies

For a detailed list of required and/or optional software and libraries, refer to the following table of Software Dependencies.

NB: This list is not exhaustive, please refer to individual Phing tasks to find out additional software requirements.

Table 3.1: *Software Dependencies*

Software	Required for	Source
PHP 5.2+	Execution	http://www.php.net
PHPUnit 3.4.0+	Optional; enables additional task(s)	http://www.phpunit.de
Xdebug 2.0.0+	Optional; enables additional task(s)	http://www.xdebug.org
SimpleTest 1.0.1 beta+	Optional; enables additional task(s)	http://simpletest.sourceforge.net
PhpDocumentor 1.4.0+ (PEAR package)	Optional; enables additional task(s)	http://pear.php.net/package/PhpDocumentor
VersionControl_SVN (PEAR package)	Optional; enables additional task(s)	http://pear.php.net/package/VersionControl_SVN
VersionControl_Git (PEAR package)	Optional; enables additional task(s)	http://pear.php.net/package/VersionControl_Git

Software	Required for	Source
PHP_CodeSniffer (PEAR package)	Optional; enables addi-tional task(s)	http://pear.php.net/package/PHP_CodeSniffer
Archive_Tar (PEAR package)	Optional; enables addi-tional task(s)	http://pear.php.net/package/Archive_Tar
Services_Amazon_S3 (PEAR package)	Optional; enables addi-tional task(s)	http://pear.php.net/package/Services_Amazon_S3
HTTP_Request2 (PEAR package)	Optional; enables addi-tional task(s)	http://pear.php.net/package/HTTP_Request2
PHP Depend	Optional; enables addi-tional task(s)	http://www.pdepend.org
PHP Mess Detector	Optional; enables addi-tional task(s)	http://www.phpmd.org
PHP Copy/Paste Detector	Optional; enables addi-tional task(s)	http://pear.phpunit.de
DocBlox 0.17.0+	Optional; enables addi-tional task(s)	http://www.docblox-project.org

Warning

Phing does not work with safe mode enabled in PHP!

3.2. Obtaining Phing

Phing is free software distributed under the terms of the LGPL.

3.2.1. Distribution Files

There are several ways to get a Phing distribution package. If you do not want to participate in developing Phing itself it is recommended that you get the latest snapshot or stable packaged distribution. If you are interested in helping with Phing development, register an account at GitHub as described below.

The easiest way to obtain the distribution package is to visit the Phing website [phing][Bibliography.html#phing]and download the current distribution package in the format you desire.

As of version 2.0, you have the option of downloading a PEAR-installable package or the full phing distribution. If you wish to modify phing we suggestion downloading the full version so that you can (e.g.) create your own PEAR package. If you simply wish to use Phing for your own project or need it to build another package, download & install the PEAR package.

3.2.2. Getting the latest source from Phing Git repository

The latest snapshot can always be downloaded directly the official Phing Git repository. However, be warned that there is not guarantee that the momentous state of the repository represents a completely stable application without any problems.

You can download a snapshot as a zip-tarball from:

- https://github.com/phingofficial/phing

3.3. PEAR Install

The easiest way to install Phing is using the PEAR installer. Provided that the package you downloaded is a the PEAR-ready package, you can install Phing simply from the command line (Unix or Windows):

```
$> pear channel-discover pear.phing.info
$> pear install phing/phing
```

The pear installer will check any dependencies and place the phing script (phing or phing.bat) into your PHP script directoy (i.e. where the "pear" script resides).

3.4. Non-PEAR Install

If you are not using the PEAR installer, you will need to setup your environment in order to run Phing. The distribution of Phing consists of three directories: `bin` , `docs` and `classes`. Only the `bin`, `classes` and `etc` directories are required to run Phing. To install Phing, choose a directory and uncompress the distribution file in that directory (you may already have done this in a prior step). This directory will be known as `PHING_HOME` .

Warning

On earlier Windows installations, the script used to launch Phing will have problems if `PHING_HOME` is a long filepath. This is due to limitations in the OS's handling of the "for" batch-file statement. It is recommended, therefore, that Phing be installed in a short path, such as `C:\opt\phing`.

Before you can run Phing there is some additional set up you will need to do perform:

- Add the full path to the `bin/` directory to your path.
- Set the `PHING_HOME` environment variable to the directory where you installed Phing. On some operating systems the Phing wrapper scripts can guess PHING_HOME (Unix dialects and Windows). However, it is better to not rely on this behavior.
- Set the `PHP_COMMAND` environment variable to where your Php binary is located (including the binary i.e. PHP_COMMAND=/usr/bin/php).
- Set the `PHP_CLASSPATH` environment variable (see the section below). This should be set at least point to PHING_HOME/classes. Alternatively, you can also just add the phing/classes directory to your PHP include_path ini setting.
- Check your php.ini file to make sure that you have the following settings:
  - `max_execution_time = 0 // unlimited execution time`
  - `memory_limit = 32M // you may need more memory depending on size of your build files`

If you are using Phing in conjunction with another application, you may need to add additional paths to `PHP_CLASSPATH`.

3.4.1. Unix

Assuming you are running a Unix dialect operating system with the bash bourne shell and Phing is installed in `/opt/phing` . The following sets up the environment properly:

```
export PHP_COMMAND=/usr/bin/php
export PHING_HOME=/opt/phing
export PHP_CLASSPATH=${PHING_HOME}/classes
export PATH=${PATH}:${PHING_HOME}/bin
```

3.4.2. Windows

On the Windows platform, assuming Phing is installed in `c:\opt\phing`. The following sets up your environment:

```
set PHP_COMMAND=c:\opt\php\php.exe
set PHING_HOME=c:\opt\phing
set PHP_CLASSPATH=c:\opt\phing\classes
set PATH=%PATH%;%PHING_HOME%\bin
```

3.4.3. Advanced

There are lots of variants that can be used to run/prepare Phing. You need at least the following:

- If you want Phing to be able to use other packages / classes, you can either add them to the `PHP_CLASSPATH` or to PHP's include_path.
- Some Tasks in `phing/tasks/ext` may require 3rd party libraries to be installed. Generally, tools with compatible license (and stable releases) are included in phing/lib so that outside dependencies can be avoided. PEAR libs will not, however, be bundled with Phing since they are generally bundled with PHP. If you are using a 3rd party task, see the Task documentation to be aware of any dependencies.

You are now ready to use the **phing** command at your command prompt, from everywhere in your directory tree.

3.5. Calling Phing

Now you are prepared to execute Phing on the command line or via script files. The following section briefly describe how to properly execute phing.

3.5.1. Command Line

Phing execution on the command line is simple. Just change to the directory where your buildfile resides and type

```
$ phing [target [target2 [target3] ...]]
```

at the command line (where [target...] are the target(s) you want to be executed). If no target is specified Phing will try to execute the default target, as specified in the `project` tag. When calling multipe targets, Phing will invoke each target independently of the other targets. Optionally, you may specify command line arguments as listed in Appendix A [appendixes/AppendixA-FactSheet.html#CommandLineArguments].

For example, the following command line calls the default buildscript `build.xml` using the default target with the property `ftp.upload` set to true.

```
$ phing -Dftp.upload=true
```

3.5.2. Supported command line arguments

As of version 2.4.9 the following command line arguments are supported

```
  -h -help               print this message
  -l -list               list available targets in this project
  -v -version            print the version information and exit
  -q -quiet              be extra quiet
  -verbose               be extra verbose
  -debug                 print debugging information
  -longtargets           show target descriptions during build
  -logfile <file>        use given file for log
  -logger <classname>    the class which is to perform logging
  -f -buildfile <file>   use given buildfile
  -D<property>=<value>   use value for given property
  -propertyfile <file>   load all properties from file
  -find <file>           search for buildfile towards the root of the
                         filesystem and use it
  -inputhandler <file>   the class to use to handle user input
```

Chapter 4. Getting started

Phing buildfiles are written in XML, and so you will need to know at least some basic things about XML to understand the following chapter. There is a lot of information available on the web:

- The Standard Recommendation of XML by the W3C http://www.w3.org/TR/2000/REC-xml: very technical but exhaustive.
- XML In 10 Points http://www.w3.org/XML/1999/XML-in-10-points: Quick introduction into XML.
- A technical introduction to XML http://www.xml.com/pub/a/98/10/guide0.html: Interesting article by the creator of DocBook.

4.1. XML And Phing

A valid Phing buildfile has the following basic structure:

- The document prolog
- Exactly one root element called `<project>` .
- Several Phing `type` elements (i.e. `<property>` , `<fileset>` , `<patternset>` etc.)
- One or more `<target>` elements containing built-in or user defined Phing `task` elements (i.e. `<install>` , `<bcc>` , etc).

4.2. Writing A Simple Buildfile

The `Foobar` project installs some PHP files from a source location to a target location, creates an archive of this files and provides an optional clean-up of the build tree:

```
<?xml version="1.0" encoding="UTF-8"?>

<project name = "FooBar" default = "dist">

    <!-- ============================================  -->
    <!-- Target: prepare                               -->
    <!-- ============================================  -->
    <target name = "prepare">
        <echo msg = "Making directory ./build" />
        <mkdir dir = "./build" />
    </target>

    <!-- ============================================  -->
    <!-- Target: build                                 -->
    <!-- ============================================  -->
    <target name = "build" depends = "prepare">
        <echo msg = "Copying files to build directory..." />

        <echo msg = "Copying ./about.php to ./build directory..." />
        <copy file = "./about.php" tofile = "./build/about.php" />

        <echo msg = "Copying ./browsers.php to ./build directory..." />
        <copy file = "./browsers.php" tofile = "./build/browsers.php" />
```

```
        <echo msg = "Copying ./contact.php to ./build directory..." />
        <copy file = "./contact.php" tofile = "./build/contact.php" />
    </target>

    <!-- ============================================  -->
    <!-- (DEFAULT)  Target: dist                       -->
    <!-- ============================================  -->
    <target name = "dist" depends = "build">
        <echo msg = "Creating archive..." />

        <tar destfile = "./build/build.tar.gz" compression = "gzip">
            <fileset dir = "./build">
                <include name = "*" />
            </fileset>
        </tar>

        <echo msg = "Files copied and compressed in build directory OK!" />
    </target>
</project>
```

A phing build file is normally given the name `build.xml` which is the default file name that the Phing executable will look for if no other file name is specified.

To run the above build file and execute the default target (assuming it is stored in the current directory with the default name) is only a matter of calling: `$ phing`

This will then execute the `dist` target. While executing the build file each task performed will print some information on what actions and what files have been affected.

To run any of the other target is only a matter of providing the name of the target on the command line. So for example to run the `build` target one would have to execute `$ phing build`

It is also possible to specify a number of additional command line arguments as described in Appendix A, *Fact Sheet*

4.2.1. Project Element

The first element after the document prolog is the root element named `<project>` on line 3. This element is a container for all other elements and can/must have the following attributes:

Table 4.1: *<project> Attributes*

Attribute	Description	Required
`name`	The name of the project	No
`basedir`	The base directory of the project, use "." do denote the current directory. **Note:** if none is specified, the parent directory of the build file is used!	No
`default`	The default target that is to be executed if no target(s) are specified when calling this build file.	Yes
`description`	The description of the project.	No

4.2.2. Target Element

A target can `depend` on other targets. You might have a target for installing the files in the build tree, for example, and a target for creating a distributable tar.gz archive. You can only build a distributable when you have installed the files first, so the distribute target depends on the install target. Phing resolves these dependencies.

It should be noted, however, that Phing's depends attribute only specifies the order in which targets should be executed - it does not affect whether the target that specifies the dependency(s) gets executed if the dependent target(s) did not (need to) run.

Phing tries to execute the targets in the depends attribute in the order they appear (from left to right). Keep in mind that it is possible that a target can get executed earlier when an earlier target depends on it, in this case the dependant is only executed once:

```
<target name="A" />
<target name="B" depends="A" />
<target name="C" depends="B" />
<target name="D" depends="C,B,A" />
```

Suppose we want to execute target `D` . From its depends aliteralribute, you might think that first target `C` , then `B` and then `A` is executed. Wrong! `C` depends on `B` , and `B` depends on `A` , so first `A` is executed, then `B` , then `C` , and finally `D` .

A target gets executed only once, even when more than one target depends on it (see the previous example).

The optional description attribute can be used to provide a one-line description of this target, which is printed by the `-projecthelp` command-line option.

Target attributes

You can specify one or more of the following attributes within the target element.

Table 4.2: *<target> Attributes*

Attribute	Description	Required
name	The name of the target	Yes
depends	A comma-seperated list of targets this target depends on.	No
if	The name of the `Property` that has to be set in order for this target to be executed	No
unless	The name of the `Property` that must `not` be set in order for this target to be executed.	

4.2.3. Task Elements

A `task` is a piece of PHP code that can be executed. This code implements a particular action to perform (i.e. install a file). Therefore it must be defined in the buildfile so that it is actually invoked by Phing.

These references will be resolved before the task is executed.

Tasks have a common structure:

```
<name attribute1="value1" attribute2="value2" ... />
```

where `name` is the name of the task, `attributeN` is the attribute name, and `valueN` is the value for this attribute.

There is a set of core tasks (see Appendix B, *Core tasks*) along with a number of optional tasks. It is also very easy to write your own tasks (see Chapter 6, *Extending Phing*).

Tasks can be assigned an `id` attribute:

```
<taskname id="taskID" ... />
```

By doing this you can refer to specific tasks later on in the code of other tasks.

4.2.4. Property Element

`Properties` are essentially variables that can be used in the buildfile. These might be set in the buildfile by calling the Appendix B, *Core tasks*, or might be set outside Phing on the command line (properties set on the command line always override the ones in the buildfile). A property has a name and a value only. Properties may be used in the value of task attributes. This is done by placing the property name between " `${` " and " `}` " in the attribute value. For example, if there is a `BC_BUILD_DIR` property with the value 'build', then this could be used in an attribute like this: `${BC_BUILD_DIR}/en` . This is resolved to `build/en`.

Built-in Properties

Phing provides access to system properties as if they had been defined using a `<property>` task. For example, `${os.name}` expands to the name of the operating system. See Appendix A, *Fact Sheet* for a complete list

4.3. More Complex Buildfile

```
<?xml version="1.0"  encoding="UTF-8" ?>

<project name = "testsite" basedir = "." default = "main">
    <property file = "./build.properties" />

    <property name = "package"  value = "${phing.project.name}" override = "true" />
    <property name = "builddir" value = "./build/testsite" override = "true" />
    <property name = "srcdir"   value = "${project.basedir}" override = "true" />

    <!-- Fileset for all files -->
    <fileset dir = "." id = "allfiles">
        <include name = "**" />
    </fileset>

    <!-- ============================================  -->
    <!-- (DEFAULT) Target: main                        -->
    <!-- ============================================  -->
    <target name = "main" description = "main target">
        <copy todir = "${builddir}">
            <fileset refid = "allfiles" />
        </copy>
    </target>

    <!-- ============================================  -->
    <!-- Target: Rebuild                               -->
    <!-- ============================================  -->
    <target name = "rebuild" description = "rebuilds this package">
        <delete dir = "${builddir}" />
        <phingcall target = "main" />
    </target>
</project>
```

This build file first defines some properties with the `<property>` task call to `PropertyTask`. Then, it defines a fileset and two targets. Let us have a quick rundown of this build file.

The first five four within the `project` tag define properties. They appear in the two ways this tag can occur:

- The second `property` tag contains only the `file` attribute. The value has to be a relative or absolute path to a property file (for the format, see Appendix I, *File Formats*).
- The other times, the tag has a `name` and a `value` attribute. After the call, the value defined in the attribute `value` is available through the key enclosed in "${" and "}".

The next noticeable thing in the build file is the `<fileset>` tag. It defines a `fileset`, i.e. a set of multiple files. You can include and exclude files with the `include` and `exclude` tags within the `fileset` tag. For more information concerning Filesets (i.e. Patterns) see Appendix D, *Core Types*. The `fileset` is given an `id` attribute, so it can be referenced later on.

One thing is worth noting here though and that is the use of double star expression, i.e. `"**"`. This special regexp refers to all files in all subdirectories as well. Compare this with a single `"*"` which would only refer to all files in the `current` subdirectory. So for example the expression `"**/*.phps"` would refer to all files with suffix `"'.phps"` in all subdirectories below the current directory.

The first task only contains a call to `CopyTask` via `<copy>`. The interesting thing is within the `copy` tag. Here, a fileset task is not written out with nested `include` or `exclude` elements, but via the `refid`, the Fileset created earlier is referenced. This way, you can use a once defined fileset multiple times in your build files.

The only noticeable thing in the second target is the call to `PhingTask` with the `<phingcall>` tag (see Appendix B, *Core tasks* for more information). The task executes a specified target within the same build file. So, the second target removes the build directory and calls `main` again, thus rebuilding the project.

A variant is to override properties defined in the build file with properties specified on the command line using the `-D` switch. For example to override the `builddir` in the build file above one could call Phing as

```
$ phing -Dbuilddir=/tmp/system-test
```

4.3.1. Handling source dependencies

A common task required in many build files is to keep some target which has a number of dependencies up to date. In traditional make files this could for example be an executable that needs to be recompiled if any of the source files have been updated. In Phing such a condition is handled by the `UpToDateTask` , see Section B.36, “UpToDateTask ” for examples on how this task us used.

4.4. Relax NG Grammar

With a little bit of experience it is not that difficult to write and understand Phing build files since the XML format in itself tends to be quite verbose. However, it can become a bit tedious and the large (and growing) amount of built-in tasks and filters can sometimes make it difficult to remember the exact syntax of all the available features.

To help with this the Phing distribution contains a Relax NG Grammar (**RE**gular **LA**nguage for **X**ML **N**ext **G**eneration, http://www.relaxng.org/) file that describes the (formal) syntax of the build files. This grammar can be used to validate build files. However, the most beneficial use of the grammar is together with a schema aware XML editor. Such an editor can make auto-completion based on the grammar. This feature makes writing complex build files significantly easier since it is usually enough to enter the

first letter of an element to have the rest of the element written automatically as well as any compulsory attributes.

Most XML editors can be told to what schema (or model) to use for validation and auto-completion by adding a specification in the beginning of the XML file. For example, the following two lines in the beginning of an XML file would do (of course the exact path to the grammar will depend on your system setup)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model xlink:href="/usr/share/php5/PEAR/data/phing/etc/phing-grammar.rng"
            type="application/xml"
            schematypens="http://relaxng.org/ns/structure/1.0" ?>
```

Using auto-completion will make it substantially easier to edit large build files. Please note that since the phing-grammar does not have an official designation we must use the absolute filename to specify the grammar (instead of a canonical URI that is resolved by the systems XML-catalogue).

This grammar is available (as a plain text file) in the distribution at: `/etc/phing-grammar.rng`

Since we do not want to neither endorse nor forget any particular XML editor with this capability we do not make available such a list of editors. Instead, spending a few minutes with Google searching for XML-editors is bound to find a number of editors with this capability.

If you wish to validate your Phing build file, there are numerous options. Links to various validation tools and XML editors are available at the RELAX NG home page, http://www.relaxng.org/. The command line tool `xmllint` that comes with libxml2 is also able to validate a given XML file against the supplied grammar.

For example, to use **xmllint** to validate a Phing build file the following command line could be used:

```
$ xmllint -noout -relaxng phing-grammar.rng build.xml
build.xml validates
```

Chapter 5. Project components

This goal of this chapter is to make you familiar with the basic components of a buildfile. After reading this chapter, you should be able to read and understand the basic structure of any buildfile even if you don't know exactly what the individual pieces do.

For supplemental reference information, you should see Appendix B, *Core tasks*, Appendix D, *Core Types* and Appendix H, *Project Components*.

5.1. Projects

In the structure of a Phing buildfile, there must be exactly one `Project` defined; the `<project>` tag is the root element of the buildfile, meaning that everything else in the buildfile is contained within the `<project >` element.

```
<?xml version="1.0"?>

<project name = "test" description = "Simple test build file" default = "main" >
  <!-- Everything else here -->
<project>
```

The listing above shows a sample `<project>` tag that has all attributes available for Projects. The `name` and `description` attributes are fairly self-explanatory; the `default` attribute specifies the default `Target` to execute if no target is specified (Section H.2, “Targets” are described below). For a complete reference, see Appendix H, *Project Components*.

5.2. Version

Since Phing 2.4.2 it is possible to include a `phingVersion` attribute in the `<project>` tag. This attribute allows you to define the minimum Phing version required to execute a build file, in order to prevent compatibility issues.

```
<?xml version="1.0"?>

<project name = "test" phingVersion = "2.4.2" >
  <!-- Everything else here -->
<project>
```

5.3. Project Components in General

Project Components are all the elements found inside a project, i.e. targets, tasks, types, etc. Project components may have attributes and nested tags. Attributes only contain simple values, i.e. strings, integers etc. Nested elements may be complex Phing types (like FileSets) or simple wrapper classes for values with custom keys (see Appendix D, *Core Types* for example).

Any nested elements must be supported by the class that implements the project component, and because the nested tags are handled by the project component class the same nested tag may have different meanings (and different attributes) depending on the context. So, for example, the nested tag `<param.../>` within the `<phingcall>` tag is handled very differently from the`<param.../>` tag within

the <xsltfilter> tag -- in the first case setting project properties, in the second case setting XSLT parameters.

5.4. Targets

Targets are collections of project components (but not other targets) that are assigned a unique name within their project. A target generally performs a specific task -- or calls other targets that perform specific tasks -- and therefore a target is a bit like a `function` (but a target has no return value).

Targets may `depend` on other targets. For example, if target A depends on a target B, then when target A is called to be executed, target B will be executed first. Phing automatically resolves these dependencies. You cannot have circular references like: "target A depends on target B that depends on target A".

The following code snippet shows an example of the use of targets.

```
<target name = "othertask" depends = "buildpage" description = "Whatever">
  <!-- Task calls here -->
<target>

<target name = "buildpage" description = "Some description">
  <!-- Task calls here -->
<target>
```

When Phing is asked to execute the `othertask` target, it will see the dependency and execute `buildpage` first. Notice that the dependency task can be defined after the dependent task.

5.5. Tasks

Tasks are responsible for doing the work in Phing. Basically, tasks are the individual actions that your buildfile can perform. For example, tasks exist to copy a file, create a directory, TAR files in a directory. Tasks may also be more complex such as XsltTask which copies a file and transforms the file using XSLT, SmartyTask which does something similar using Smarty templates, or CreoleTask which executes SQL statements against a specified DB. See Appendix B, *Core tasks* for descriptions of Phing tasks.

Tasks support parameters in the form of:

- Simple parameters (i.e. strings) passed as XML attributes, or
- More complex parameters that are passed by nested tags

Simple parameters are basically strings. For example, if you pass a value `"A simple string."` as a parameter, it is evaluated as a string and accessible as one. You can also reference properties as described in Chapter 4, *Getting started*.

`Note:` There are special values that are not mapped to strings, but to boolean values instead. The values `true`, `false`, `yes`, `no`, `on` and `off` are translated to `true`/`false` boolean values.

```
<property name = "myprop" value = "value" override = "true"/>
```

However, some tasks support more complex data types as parameters. These are passed to the task with `nested tags`. Consider the following example:

```
<copy>
```

```
  <fileset dir = ".">
    <include name = "**" />
  </fileset>
</copy>
```

Here, `CopyTask` is passed a complex parameter, a Fileset. Tasks may support multiple complex types in addition to simple parameters. Note that the names of the nested tags used to create the complex types depend on the task implementation. Tasks may support default Phing types (see Section 5.6, “ Types ”) or may introduce other types, for example to wrap key/value pairs.

Refer to Appendix B, *Core tasks* for a list of system tasks and their parameters.

5.6. Types

5.6.1. Basics

Besides the simple types (strings, integer, booleans) you can use in the parameters of tasks, there are more complex Phing `Types`. As mentioned above, they are passed to a task by using nesting tags:

```
<task>
  <type />
</task>

<!-- or: -->

<task>
  <type1>
    <subtype1>
      <!-- etc. -->
    </subtype1>
  </type1>
</task>
```

Note that types may consist of multiple nested tags -- and multiple levels of nested tags, as you can see in the second task call above.

5.6.2. Referencing Types

An additional fact about types you should notice is the possibility of `referencing` type instances, i.e. you define your type somewhere in your build file and assign an id to it. Later, you can refer to that type by the id you assigned. Example:

```
<project>
  <fileset id = "foo">
    <include name = "*.php" />
  </fileset>

  <!-- Target that uses the type -->
  <target name = "foo" >
    <copy todir = "/tmp">
      <fileset refid = "foo" />
    </copy>
  </target>
</project>
```

As you can see, the type instance is assigned an id with the `id` attribute and later on called by passing a plain `fileset` tag to `CopyTask` that only contains the `refid` attribute.

5.7. Basic Types

The following section gives you a quick introduction into the basic Phing types. For a complete reference see Appendix D, *Core Types*.

5.7.1. FileSet

FileSets are groups of files. You can include or exclude specific files and patterns to/from a FileSet. The use of patterns is explained below. For a start, look at the following example:

```
<fileset dir = "/tmp" id = "fileset1">
  <include name = "sometemp/file.txt" />
  <include name = "othertemp/**" />
  <exclude name = "othertemp/file.txt" />
</fileset>

<fileset dir = "/home" id = "fileset2">
  <include name = "foo/**" />
  <include name = "bar/**/*.php" />
  <exclude name = "foo/tmp/**" />
</fileset>
```

The use of patterns is quite straightforward: If you simply want to match a part of a filename or dirname, you use *. If you want to include multiple directories and/or files, you use **. This way, filesets provide an easy but powerful way to include files.

5.7.2. FileList

FileLists, like FileSets, are collections of files; however, a FileList is an explicitly defined list of files -- and the files don't necessarily have to exist on the filesystem.

Besides being able to refer to nonexistent files, another thing that `FileLists` allow you to do is specify files `in a certain order`. Files in `FileSets` are ordered based on the OS-level directory listing functions, in some cases you may want to specify a list of files to be processed in a certain order -- e.g. when concatenating files using the <append> task.

```
<filelist dir = "base/" files = "file1.txt,file2.txt,file3.txt"/>

<!-- OR: -->
<filelist dir = "basedir/" listfile = "files_to_process.txt"/>
```

5.7.3. FilterChains and Filters

`FilterChains` can be compared to Unix pipes. Unix pipes add a great deal of flexibility to command line operations; for example, if you wanted to copy just those lines that contained the string `blee` from the first 10 lines of a file called `foo` to a file called `bar`, you could do:

```
cat foo | head -n10 | grep blee > bar
```

Something like this is not possible with the tasks and types that we have learned about thus far, and this is where the incredible usefulness of `FilterChains` becomes apparent. They emulate Unix pipes and provide a powerful dimension of file/stream manipulation for the tasks that support them.

`FilterChain` usage is quite straightforward: you pass the complex Phing type `filterchain` to a task that supports FilterChains and add individual filters to the FilterChain. In the course of executing the task, the filters are applied (in the order in which they appear in the XML) to the contents of the files that are being manipulated by your task.

```
<filterchain>
  <replacetokens>
    <token key = "BC_PATH" value = "${top.builddir}/"/>
    <token key = "BC_PATH_USER" value = "${top.builddir}/testsite/user/${lang}/"/>
  </replacetokens>

  <filterreader classname = "phing.filters.TailFilter">
    <param name = "lines" value = "10"/>
  </filterreader>
</filterchain>
```

The code listing above shows you some example of how to use filter chains. For a complete reference see Appendix D, *Core Types*. This filter chain would replace all occurrences of BC_PATH and BC_PATH_USER with the values assigned to them in lines 4 and 5. Additionally, it will only return the last 10 lines of the files.

Notice above that `FilterChain` filters have a "shorthand" notation and a long, generic notation. Most filters can be described using both of these forms:

```
<replacetokens>
  <token key = "BC_PATH" value = "${top.builddir}/"/>
  <token key = "BC_PATH_USER" value = "${top.builddir}/testsite/user/${lang}/"/>
</replacetokens>

<!-- OR: -->

<filterreader classname = "phing.filters.ReplaceTokens">
  <param type = "token" name = "BC_PATH" value = "${top.builddir}/"/>
  <param type = "token" name = "BC_PATH" value = "${top.builddir}/testsite/user/${lang}/"/>
</filterreader>
```

As the pipe concept in Unix, the filter concept is quite complex but powerful. To get a better understanding of different filters and how they can be used, take a look at any of the many uses of FilterChains in the build files for the binarycloud Bibliography project.

5.7.4. File Mappers

With `FilterChains` and filters provide a powerful tool for changing contents of files, Mappers provide a powerful tool for changing the names of files.

To use a Mapper, you must specify a pattern to match on and a replacement pattern that describes how the matched pattern should be transformed. The simplest form is basically no different from the DOS `copy` command:

```
copy *.bat *.txt
```

In Phing this is the `glob` Mapper:

```
<mapper type = "glob" from = "*.bat" to = "*.txt"/>
```

Phing also provides support for more complex mapping using regular expressions:

```
<mapper type = "regexp" from = "^(.*)\.conf\.xml$$" to = "\1.php"/>
```

Consider the example below to see how Mappers can be used in a build file. This example includes some of the other concepts introduced in this chapter, such as `FilterChains` and `FileSets`. If you don't understand everything, don't worry. The important point is that Mappers are types too, which can be used in tasks that support them.

```
<copy>
```

```
  <fileset dir = ".">
    <include name = "*.ent.xml"/>
  </fileset>

  <mapper type = "regexp" from = "^(.*)\.ent\.xml$" to = "\1.php"/>

  <filterchain>
    <filterreader classname = "phing.filters.XsltFilter">
      <param name = "style" value = "ent2php.xsl"/>
    </filterreader>
  </filterchain>
</copy>
```

For a complete reference, see Appendix D, *Core Types*

5.8. Conditions

Conditions are nested elements of the Appendix B, *Core tasks* and Appendix B, *Core tasks* tasks.

5.8.1. not

The <not> element expects exactly one other condition to be nested into this element, negating the result of the condition. It doesn't have any attributes and accepts all nested elements of the condition task as nested elements as well.

5.8.2. and

The <and> element doesn't have any attributes and accepts an arbitrary number of conditions as nested elements. This condition is true if all of its contained conditions are, conditions will be evaluated in the order they have been specified in the build file.

The <and> condition has the same shortcut semantics as the && operator in some programming languages, as soon as one of the nested conditions is false, no other condition will be evaluated.

5.8.3. or

The <or> element doesn't have any attributes and accepts an arbitrary number of conditions as nested elements. This condition is true if at least one of its contained conditions is, conditions will be evaluated in the order they have been specified in the build file.

The <or> condition has the same shortcut semantics as the || operator in some programming languages, as soon as one of the nested conditions is true, no other condition will be evaluated.

5.8.4. os

Test whether the current operating system is of a given type.

Table 5.1: *OS Attributes*

Attribute	Description	Required
family	The name of the operating system family to expect.	Yes

Supported values for the family attribute are:

- windows (for all versions of Microsoft Windows)
- mac (for all Apple Macintosh systems)
- unix (for all Unix and Unix-like operating systems)

5.8.5. equals

Tests whether the two given Strings are identical

Table 5.2: *equals Attributes*

Attribute	Description	Required
arg1	First string to test.	Yes
arg2	Second string to test.	Yes
casesensitive	Perform a case sensitive comparision. Default is true.	No
trim	Trim whitespace from arguments before comparing them. Default is false.	No

5.8.6. isset

Test whether a given property has been set in this project.

Table 5.3: *isset Attributes*

Attribute	Description	Required
property	The name of the property to test.	Yes

5.8.7. contains

Tests whether a string contains another one.

Table 5.4: *contains Attributes*

Attribute	Description	Required
string	The string to search in.	Yes
substring	The string to search for.	Yes
casesensitive	Perform a case sensitive comparision. Default is true.	No

5.8.8. istrue

Tests whether a string evaluates to true.

Table 5.5: *istrue Attributes*

Attribute	Description	Required
value	value to test	Yes

```
<istrue value = "${someproperty}"/>
```

```
<istrue value = "false"/>
```

5.8.9. isfalse

Tests whether a string evaluates to not true, the negation of <istrue>

Table 5.6: *isfalse Attributes*

Attribute	Description	Required
value	value to test	Yes

```
<isfalse value = "${someproperty}"/>
<isfalse value = "false"/>
```

5.8.10. referenceexists

Tests whether a specified reference exists.

Table 5.7: *referenceexists Attributes*

Attribute	Description	Required
ref	reference to test for	Yes

```
<referenceexists ref = "${someid}"/>
```

5.8.11. available

This condition is identical to the Available [#AvailableTask] task, all attributes and nested elements of that task are supported, the property and value attributes are redundant and will be ignored.

Chapter 6. Extending Phing

Phing was designed to be flexible and easily extensible. Phing's existing core and optional tasks do provide a great deal of flexibility in processing files, performing database actions, and even getting user feedback during a build process. In some cases, however, the existing tasks just won't suffice and because of Phing's open, modular architecture adding exactly the functionality you need is often quite trivial.

In this chapter we'll look primarily at how to create your own tasks, since that is probably the most useful way to extend Phing. We'll also give some more information about Phing's design and inner workings.

6.1. Extension Possibilities

There are three main areas where Phing can be extended: Tasks, Types, Mappers. The following sections discuss these options.

6.1.1. Tasks

Tasks are pieces of codes that perform an atomic action like installing a file. Therefore a special worker class hast to be created and stored in a specific location, that actually implements the job. The worker is just the interface to Phing that must fulfill some requirements discussed later in this chapter, however it can - but not necessarily must - use other classes, workers and libraries that aid performing the operations needed.

6.1.2. Types

Extending types is a rare need; nevertheless, you can do it. A possible type you might implement is `urlset`, for example.

You may end up needing a new type for a task you write; for example, if you were writing the XSLTTask you might discover that you needed a special type for XSLTParams (even though in that case you could probably use the generic name/value Parameter type). In cases where the type is really only for a single task, you may want to just define the type class in the same file as the Task class, rather than creating an official stand-alone `Type`.

6.1.3. Mappers

Creating new mappers is also a rare need, since most everything can be handled by the Appendix D, *Core Types*. The Mapper framework does provide a simple way for defining your own mappers to use instead, however, and mappers implement a very simple interface.

6.2. Source Layout

6.2.1. Files And Directories

Before you are going to start to extend Phing let's have a look at the source layout. You should be comfortable with the organization of files witch in the source tree of Phing before start coding. After

you extracted the source distribution or checked it out from git you should see the following directory structure:

```
$PHING_HOME
  |-- bin
  |-- classes
  |    `-- phing
  |         |-- filters
  |         |    `-- util
  |         |-- mappers
  |         |-- parser
  |         |-- tasks
  |         |    |-- ext
  |         |    |-- system
  |         |    |    `-- condition
  |         |    `-- user
  |         `-- types
  |-- docs
  |    `-- phing_guide
  `-- test
       |-- classes
       `-- etc
```

The following table briefly describes the contents of the major directories:

Table 6.1: *Phing source tree directories*

Directory	Contents
`bin`	The basic applications (phing, configure) as well as the wrapper scripts for different platforms (currently Unix and Windows).
`classes`	Repository of all the classes used by Phing. This is the base directory that should be on the PHP include_path. In this directory you will find the subdirectory phing/ with all the Phing relevant classes.
`docs`	Documentation files. Generated books, online manuals as well as the PHPDoc generated API documentation.
`test`	A set of testcases for different tasks, mappers and types. If you are developing in git you should add a testcase for each implementation you check in.

Currently there is no distinction between the `source` layout and the `build` layout of Phing. The directory layout [#phing.dirlayout] shows the file tree that carries some additional files like the Phing website. Later on there may be a buildfile to create a clean distribution tree of Phing itself.

6.2.2. File Naming Conventions

There are some file naming conventions used by Phing. Here's a quick rundown on the most basic conventions. A more detailed list can be found in [See Naming And Coding Standards]:

- Filenames consist of no more or less than two elements: `name` and `extension` .
- Choose short descriptive filenames (must be less than 31 chars)
- Names must not contain dots.
- Files containing PHP code must end with the extension `.php` .
- There must be only one class per file (no procedural methods allowed, use a separate file for them), with the exception of "inner"-type / helper classes that can be declared in the same file as the "outer" / main class.

- The name portion of the file must be named exactly like the class it contains.
- Buildfiles and configure rulesets must end with the extension `.xml` .

6.2.3. Coding Standards

We are using PEAR coding standards. We are using a less strict version of these standards, but we do insist that new contributions have phpdoc comments and make explicitly declarations about public/protected/private variables and methods. If you have suggestions about improvements to Phing codebase, don't hesitate to let us know.

6.3. System Initialization

PHP installations are typically quite customized -- e.g. different memory_limit, execution timeout values, etc. The first thing that Phing does is modify PHP INI variables to create a standard PHP environment. This is performed by the `init layer` of Phing that uses a three-level initialization procedure. It basically consists of three different files:

- Platform specific wrapper scripts in bin/
- Main application in bin/
- Phing class in classes/phing/

At the first look this may seem to be unnecessary overhead. Why three levels of initialization? The main reason why there are several entry points is that Phing is build so that other frontends (e.g. PHP-GTK) could be used in place of the command line.

6.3.1. Wrapper Scripts

This scripts are technical not required but provided for the ease of use. Imagine you have to type every time you want to build your project:

```
php -qC /path/to/phing/bin/phing.php -verbose all distro snapshot
```

Indeed that is not very elegant. Furthermore if you are lax in setting your environment variables these script can guess the proper variables for you. However you should always set them.

The scripts are platform dependent, so you will find shell scripts for `Unix` like platforms (sh) as well as the batch scripts for `Windows` platforms. If you set-up your path properly you can call Phing everywhere in your system with this command-line (referring to the above example):

```
phing -v2 all distro
```

6.3.2. The Main Application (phing.php)

This is basically a wrapper for the Phing class that actually does all the logic for you. If you look at the source code for phing.php you will see that all real initialization is handled in the Phing class. phing.php is simply the command line entry point for Phing.

6.3.3. The Phing Class

Given that all the prior initialization steps passed successfully the Phing is included and `Phing::startup()` is invoked by the main application script. It sets-up the system components, system constants ini-settings, PEAR and some other stuff. The detailed start-up process is as follows:

- Start Timer
- Set System Constants
- Set Ini-Settings
- Set Include Paths

After the main application completed all operations (successfully or unsuccessfully) it calls `Phing::shutdown(EXIT_CODE)` that takes care of a proper destruction of all objects and a gracefully termination of the program by returning an `exit code` for shell usage (see [See Program Exit Codes] for a list of exit codes).

6.4. System Services

6.4.1. The Exception system

Phing uses the PHP5 try/catch/throw Exception system. Phing defines a number of Exception subclasses for more fine-grained handling of Exceptions. Low level Exceptions that cannot be handled will be wrapped in a `BuildException` and caught by the outer-most catch() {} block.

6.5. Build Lifecycle

This section exists to explain -- or try -- how Phing "works". Particularly, how Phing proceeds through a build file and invokes tasks and types based on the tags that it encounters.

6.5.1. How Phing Parses Buildfiles

Phing uses an ExpatParser class and PHP's native expat XML functions to handle the parsing of build files. The handler classes all extend the phing.parser.AbstractHandler class. These handler classes "handle" the tags that are found in the buildfile.

Core tasks and datatypes are mapped to XML tag names in the defaults.properties files -- specifically phing/tasks/defaults.properties and phing/types/defaults.properties.

It works roughly like this:

1. `phing.parser.RootHandler` is registered to handle the buildfile XML document
2. RootHanlder expects to find exactly one element: `<project>`. RootHandler invokes the ProjectHandler with the attributes from the <project> tag or throws an exception if no <project> is found, or if something else is found instead.
3. `ProjectHandler` expects to find `<target>` tags; for these `ProjectHandler` invokes the `TargetHandler`. ProjectHandler also has exceptions for handling certain tasks that can be performed

at the top-level: <resolve>, <taskdef>, <typedef>, and <property>; for these ProjectHandler invokes the TaskHandler class. If a tag is presented that doesn't match any expected tags, then ProjectHandler assumes it is a datatype and invokes the DataTypeHandler.

4. TargetHandler expects all tags to be either tasks or datatypes and invokes the appropriate handler (based on the mappings provided in the defaults.properties files).

5. Tasks and datatypes can have nested elements, but only if they correspond to a create*() method in the task or datatype class. E.g. a nested <param> tag must correspond to a createParam() method of the task or datatype.

... More to come ...

6.6. Writing Tasks

6.6.1. Creating A Task

We will start creating a rather simple task which basically does nothing more than echo a message to the screen. See [below] for the source code and the following [below] for the XML definition that is used for this task.

```php
<?php

require_once "phing/Task.php";

class MyEchoTask extends Task {

    /**
     * The message passed in the buildfile.
     */
    private $message = null;

    /**
     * The setter for the attribute "message"
     */
    public function setMessage($str) {
        $this->message = $str;
    }

    /**
     * The init method: Do init steps.
     */
    public function init() {
      // nothing to do here
    }

    /**
     * The main entry point method.
     */
    public function main() {
      print($this->message);
    }
}

?>
```

This code contains a rather simple, but complete Phing task. It is assumed that the file is named MyEchoTask.php and placed in classes/phing/tasks/my directory. We'll explain the source

code in detail shortly. But first we'd like to discuss how we should register the task to Phing so that it can be executed during the build process.

6.6.2. Using the Task

The task shown [above] must somehow get called by Phing. Therefore it must be made available to Phing so that the buildfile parser is aware a correlating XML element and it's parameters. Have a look at the minimalistic buildfile example given in [the buildfile below] that does exactly this.

```
<?xml version="1.0" ?>

<project name = "test" basedir = "." default = "test.myecho">
    <taskdef name = "myecho" classname = "phing.tasks.my.MyEchoTask" />

    <target name = "test.myecho">
      <myecho message = "Hello World" />
    </target>
</project>
```

Besides the XML document prolog and the shell elements that are required to properly execute the task (project, target) you'll find the `<taskdef>` element (line 4) that properly registers your custom task to Phing. For a detailed synopsis of the taskdef element see the [description of this task].

Now, as we have registered the task by assigning a name and the worker class ([see source code above]) it is ready for usage within the `<target>` context (line 8). You see that we pass the message that our task should echo to the screen via an XML attribute called "message".

6.6.3. Source Discussion

Now that you've got the knowledge to execute the task in a buildfile it's time to discuss how everything works.

6.6.4. Task Structure

All files containing the definition of a task class follow a common well formed structure:

- Include/require statements to import all required classes
- The class declaration and definition
- The class's properties
- The class's constructor
- Setter methods for each XML attribute
- The `init()` method
- The `main()` method
- Arbitrary `private` (or `protected`) class methods

6.6.5. Includes

Always include/require all the classes needed for this task in full written notation. Furthermore you should always include `phing/Task.php` at the very top of your include block. Then include all other required system or proprietary classes.

6.6.6. Class Declaration

If you look at line 5 in [the source code of the task] you will find the `class declaration`. This will be familiar to you if you are experienced with OOP in PHP (we assume here that you are). Furthermore there are some fine-grained rules you must obey when creating the classes (see also,[naming and coding standards]):

- Your classname must be exactly like the taskname you are going to implement plus the suffix "Task". In our example case the classname is `MyEchoTask` (constructed by the taskname "`myecho`" plus the suffix "`task`"). The upper/lower case casing is currently only for better reading. However, it is encouraged that you use it this way.
- The task class you are creating must at least extend "`Task`" to inherit all task specific methods.

6.6.7. Class Properties

The next lines you are coding are class properties. Most of them are inherited from the Task superclass, so there's not need to redeclare them. Nevertheless you should `declare` the following ones by your own:

- Taskname. Always hard code the `taskname` property that equals the name of the XML element that your task claims. Currently this information is not used - but it will be in the future.
- Your arbitrary properties that reflect the XML attributes/elements which your task accepts.

In the `MyEchoTask` example the coded properties can be found in lines 7 to 11. Give you properties meaningful descriptive names that clearly state their function within the context. A couple of properties are inherited from the superclass that must not be declared in the properties part of the code.

For a list of inherited properties (most of them are reserved, so be sure not to overwrite them with your own) can be found in the "Phing API Reference" in the `docs/api/` directory.

6.6.8. The Constructor

The next block that follows is the class's constructor. It must be present and call at least the constructor or the parent class. Of course, you can add some initialization data here. It is recommended that you `define` your prior declared properties here.

6.6.9. Setter Methods

As you can see in the XML definition of our task ([see buildfile above] , line 9) there is an attribute defined with the task itself, namely "message" with a value of the text string that our task should echo. The task must somehow become aware of the attribute name and the value. Therefore the `setter methods` exist.

For each attribute you want to import to the task's namespace you have to define a method named exactly after the very attribute plus the string "set" prepended. This method accepts exactly one parameter that holds the value of the attribute. Now you can set the a class internal property to the value that is passed via the setter method.

In the setter method you should also perform any casting operations and/or check if the attribute value is a valid value. If this is not the case, throw a `BuildException`. In some cases, such as when you have three attributes and at least one of them should be set, you may want to check the attribute values inside the init() or main() method.

In out example the setter is named `setMessage` , because the XML attribute the echo task accepts is "message". setMessage now takes the string "Hello World" provided by the parser and sets the value of the internal class property `$strMessage` to "Hello World". It is now available to the task for further disposal.

6.6.10. Creator Methods

Creator methods allow you to manage nested XML tags in your new Phing Task.

6.6.11. `init()` Method

The `init` method gets called when the `<taskname>` xml element closes. It must be implemented even if it does nothing like in the example above. You can do init steps here required to setup your task object properly. After calling the Init-Method the task object remains untouched by the parser. Init should not perform operations related somehow to the action the task performs. An example of using init may be cleaning up the $strMessage variable in our example (i.e. `trim($strMessage)`) or importing additional workers needed for this task.

The init method should return true or an error object evaluated by the governing logic. If you don't implement init method, phing will shout down with a fatal error.

6.6.12. `main()` Method

There is exactly one entry point to execute the task. It is called after the complete buildfile has been parsed and all targets and tasks have been scheduled for execution. From this point forward the very implementation of the tasks action starts. In case of our example a message (imported by the proper setter method) is Logged to the screen through the system's "Logger" service (the very action this task is written for). The `Log()` method-call in this case accepts two parameters: a event constant and the message to log.

6.6.13. Arbitrary Methods

For the more or less simple cases (as our example) all the logic of the task is coded in the Main() method. However for more complex tasks common sense dictates that particular action should be swapped to smaller, logically contained units of code. The most common way to do this is separating logic into private class methods - and in even more complex tasks in separate libraries.

```
private function myPrivateMethod() {
    // definition
}
```

6.7. Writing Types

You should only create a standalone Type if the Type needs to be shared by more than one Task. If the Type is only needed for a specific Task -- for example to handle a special parameter or other tag needed for that Task -- then the Type class should just be defined within the same file as the Task. (For example, `phing/filters/XSLTFilter.php` also includes an `XSLTParam` class that is not used anywhere else.)

For cases where you do need a more generic Type defined, you can create your own Type class -- similar to the way a Task is created.

6.7.1. Creating a DataType

Type classes need to extend the abstract DataType class. Besides providing a means of categorizing types, the DataType class provides the methods necessary to support the "`refid`" attribute. (All types can be given an id, and can be referred to later using that id.)

In this example we are creating a DSN type because we have written a number of DB-related Tasks, each of which need to know how to connect to the database; instead of having database parameters for each task, we've created a DSN type so that we can identify the connection parameters once and then use it in all our db Tasks.

```
require_once "phing/types/DataType.php";

/**
 * This Type represents a DB Connection.
 */
class DSN extends DataType {

  private $url;
  private $username;
  private $password;
  private $persistent = false;

  /**
   * Sets the URL part: mysql://localhost/mydatabase
   */
  public function setUrl($url) {
    $this->url = $url;
  }

  /**
   * Sets username to use in connection.
   */
  public function setUsername($username) {
    $this->username = $username;
  }

  /**
   * Sets password to use in connection.
   */
  public function setPassword($password) {
    $this->password = $password;
  }

  /**
   * Set whether to use persistent connection.
   * @param boolean $persist
   */
  public function setPersistent($persist) {
    $this->persistent = (boolean) $persist;
  }

  public function getUrl(Project $p) {
    if ($this->isReference()) {
      return $this->getRef($p)->getUrl($p);
    }
    return $this->url;
  }

  public function getUsername(Project $p) {
    if ($this->isReference()) {
      return $this->getRef($p)->getUsername($p);
```

```
    }
    return $this->username;
  }

  public function getPassword(Project $p) {
    if ($this->isReference()) {
      return $this->getRef($p)->getPassword($p);
    }
    return $this->password;
  }

  public function getPersistent(Project $p) {
    if ($this->isReference()) {
      return $this->getRef($p)->getPersistent($p);
    }
    return $this->persistent;
  }

  /**
   * Gets a combined hash/array for DSN as used by PEAR.
   * @return array
   */
  public function getPEARDSN(Project $p) {
    if ($this->isReference()) {
      return $this->getRef($p)->getPEARDSN($p);
    }

    include_once 'DB.php';
    $dsninfo = DB::parseDSN($this->url);
    $dsninfo['username'] = $this->username;
    $dsninfo['password'] = $this->password;
    $dsninfo['persistent'] = $this->persistent;

    return $dsninfo;
  }

  /**
   * Your datatype must implement this function, which ensures that there
   * are no circular references and that the reference is of the correct
   * type (DSN in this example).
   *
   * @return DSN
   */
  public function getRef(Project $p) {
    if ( !$this->checked ) {
      $stk = array();
      array_push($stk, $this);
      $this->dieOnCircularReference($stk, $p);
    }
    $o = $this->ref->getReferencedObject($p);
    if ( !($o instanceof DSN) ) {
      throw new BuildException($this->ref->getRefId()." doesn't denote a DSN");
    } else {
      return $o;
    }
  }

}
```

6.7.2. Using the DataType

The `TypedefTask` provides a way to "declare" your type so that you can use it in your build file. Here is how you would use this type in order to define a single DSN and use it for multiple tasks. (Of course you could specify the DSN connection parameters each time, but the premise behind needing a DSN datatype was to avoid specifying the connection parameters for each task.)

```
<?xml version="1.0" ?>

<project name = "test" basedir = ".">

  <typedef name = "dsn" classname = "myapp.types.DSN" />

  <dsn
      id = "maindsn"
      url = "mysql://localhost/mydatabase"
      username = "root"
      password = ""
      persistent = "false" />

  <target name = "main">

    <my-special-db-task>
      <dsn refid = "maindsn"/>
    </my-special-db-task>

    <my-other-db-task>
      <dsn refid = "maindsn"/>
    </my-other-db-task>

  </target>

</project>
```

6.7.3. Source Discussion

Getters & Setters

You must provide a setter method for every attribute you want to set from the XML build file. It is good practice to also provide a getter method, but in practice you can decide how your tasks will use your task. In the example above, we've provided a getter method for each attribute and we've also provided an additional method:`DSN::getPEARDSN()` which returns the DSN hash array used by `PEAR::DB`, `PEAR::MDB`, and Creole. Depending on the needs of the Tasks using this DataType, we may only wish to provide the `getPEARDSN()` method rather than a getter for each attribute.

Also important to note is that the getter method needs to check to see whether the current DataType is a reference to a previously defined DataType -- the `DataType::isReference()` exists for this purpose. For this reason, the getter methods need to be called with the current project, because References are stored relative to a project.

The getRef() Method

The `getRef()` task needs to be implemented in your Type. This method is responsible for returning a referenced object; it needs to check to make sure the referenced object is of the correct type (i.e. you can't try to refer to a RegularExpresson from a DSN DataType) and that the reference is not circular.

You can probably just copy this method from an existing Type and make the few changes that customize it to your Type.

6.8. Writing Mappers

Writing your own filename mapper classes will allow you to control how names are transformed in tasks like `CopyTask`, `MoveTask`, `XSLTTask`, etc. In some cases you may want to extend existing mappers

(e.g. creating a GlobMapper that also transforms to uppercase); in other cases, you may simply want to create a very specific name transformation that isn't easily accomplished with other mappers like `GlobMapper` or `RegexpMapper`.

6.8.1. Creating a Mapper

Writing filename mappers is simplified by interface support in PHP5. Essentially, your custom filename mapper must implement `phing.mappers.FileNameMapper`. Here's an example of a filename mapper that creates DOS-style file names. For this example, the "to" and "from" attributes are not needed because all files will be transformed. To see the "to" and "from" attributes in action, look at `phing.mappers.GlobMapper` or `phing.mappers.RegexpMapper`.

```
require_once "phing/mappers/FileNameMapper.php";

/**
 * A mapper that makes those ugly DOS filenames.
 */
class DOSMapper implements FileNameMapper {

  /**
   * The main() method actually performs the mapping.
   *
   * In this case we transform the $sourceFilename into
   * a DOS-compatible name.  E.g.
   * ExtendingPhing.html -> EXTENDI~.DOC
   *
   * @param string $sourceFilename The name to be coverted.
   * @return array The matched filenames.
   */
  public function main($sourceFilename) {

    $info = pathinfo($sourceFilename);
    $ext = $info['extension'];
    // get basename w/o extension
    $bname = preg_replace('/\.\w+\$/', '', $info['basename']);

    if (strlen($bname) > 8) {
      $bname = substr($bname,0,7) . '~';
    }

    if (strlen($ext) > 3) {
      $ext = substr($bname,0,3);
    }

    if (!empty($ext)) {
      $res = $bname . '.' . $ext;
    } else {
      $res = $bname;
    }

    return (array) strtoupper($res);
  }

  /**
   * The "from" attribute is not needed here, but method must exist.
   */
  public function setFrom($from) {}

  /**
   * The "from" attribute is not needed here, but method must exist.
   */
  public function setTo($to) {}

}
```

6.8.2. Using the Mapper

Assuming that this mapper is saved to `myapp/mappers/DOSMapper.php` (relative to a path on PHP's `include_path` or in `PHP_CLASSPATH` env variable), then you would refer to it like this in your build file:

```
<mapper classname = "myapp.mappers.DOSMapper"/>
```

Appendix A. Fact Sheet

A.1. Built-In Properties

Table A.1: *Phing Built-In Properties*

Property	Contents
`application.startdir`	Current work directory
`env.*`	Environment variables, extracted from `$_SERVER`.
`host.arch`	System architecture, i.e. i586. Not available on Windows machines.
`host.domain`	DNS domain name, i.e. `php.net`. Not available on Windows machines.
`host.fstype`	The type of the files ystem. Possible values are UNIX, WINNT and WIN32
`host.name`	Operating System hostname as returned by `posix_uname()`. Not available on Windows machines.
`host.os`	Operating System description as set in PHP_OS variable (see PHP Manual [http://www.php.net/manual/en/reserved.constants.core.php]).
`host.os.release`	Operating version release, i.e. 2.2.10. Not available on Windows machines.
`host.os.version`	Operating system version, i.e. #4 Tue Jul 20 17:01:36 MEST 1999. Not available on Windows machines.
`line.separator`	Character(s) that signal the end of a line, "\n" for Linux, "\r\n" for Windows system, "\r" for Macintosh.
`os.name`	Operating System description as set in PHP_OS variable.
`phing.file`	Full path to current buildfile.
`phing.dir`	Path that contains the current buildfile.
`phing.home`	Phing installation directory, not set in PEAR installations.
`phing.version`	Current Phing version.
`phing.project.name`	Name of the currently processed project.
`php.classpath`	The value of the environment variable `PHP_CLASSPATH`.
`php.version`	Version of the PHP interpreter. Same as PHP constant `PHP_VERSION` (see PHP Manual [http://www.php.net/manual/en/reserved.constants.core.php]).
`project.basedir`	The current project basedir.
`user.home`	Value of the environment variable `HOME`.

A.2. Command Line Arguments

The following table lists the command line arguments currently available.

Table A.2: *Phing Command Line Arguments*

Parameter	Meaning
`-h -help`	Display the help screen

Parameter	Meaning
`-v -version`	Print version information and exit
`-l -list`	List all available targets in buildfile (excluding targets that have their `hidden` attribute set to `true`)
`-q -quiet`	Quiet operation, no output at all
`-verbose`	Verbose, give some more output
`-debug`	Output debug information
`-logfile <file>`	Use given file for log
`-D<property>=<value>`	Set the property to the specified value to be used in the buildfile
`-find <file>`	
`-buildfile <file>`	Specify an alternate buildfile name. Default is build.xml
`-logger path.to.Logger`	Specify an alternate logger. Default is phing.listener.DefaultLogger. Other options include phing.listener.NoBannerLogger, phing.listener.AnsiColorLogger, phing.listener.XmlLogger, phing.listener.TargetLogger and phing.listener.HtmlColorLogger.
`-propertyfile <file>`	Load properties from the specified file

A.3. Distribution File Layout

```
$PHING_HOME
  |-- bin
  |-- classes
  |    `-- phing
  |         |-- filters
  |         |    `-- util
  |         |-- mappers
  |         |-- parser
  |         |-- tasks
  |         |    |-- ext
  |         |    |-- system
  |         |    |    `-- condition
  |         |    `-- user
  |         `-- types
  |-- docs
  |    `-- phing_guide
  `-- test
       |-- classes
       `-- etc
```

A.4. Program Exit Codes

Phing is script-safe - means that you can execute Phing and Configure within a automated script context. To check back the success of a Phing call it returns an exit code that can be captured by your calling script. The following list gives you details on the used exit codes and their meaning.

Table A.3: *Program Exit Codes*

Exitcode	Description
`-2`	Environment not properly defined
`-1`	Parameter or configuration error occurred
`0`	Successful execution (build succeeded), no errors (there may be warnings)
`1`	Unsuccessful execution (build failed), errors occurred

A.5. The LGPL License

Source http://www.gnu.org/licenses/lgpl.txt

```
    GNU LESSER GENERAL PUBLIC LICENSE
         Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]

       Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

  To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
```

```
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library.  Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

  Finally, software patents pose a constant threat to the existence of
any free program.  We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder.  Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License.  This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License.  We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library.  The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom.  The Lesser General
Public License permits more lax criteria for linking other code with
the library.

  We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License.  It also provides other free software developers Less
of an advantage over competing non-free programs.  These disadvantages
are the reason we use the ordinary General Public License for many
libraries.  However, the Lesser license provides advantages in certain
special circumstances.

  For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it becomes
a de-facto standard.  To achieve this, non-free programs must be
allowed to use the library.  A more frequent case is that a free
library does the same job as widely used non-free libraries.  In this
case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

  In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software.  For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

  Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

  The precise terms and conditions for copying, distribution and
modification follow.  Pay close attention to the difference between a
"work based on the library" and a "work that uses the library".  The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

    GNU LESSER GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
```

```
  0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms.  A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language.  (Hereinafter, translation is
included without limitation in the term "modification".)

  "Source code" for a work means the preferred form of the work for
making modifications to it.  For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control compilation
and installation of the library.

  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
and what the program that uses the Library does.

  1. You may copy and distribute verbatim copies of the Library's
complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

  You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

  2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application.  Therefore, Subsection 2d requires that any
```

```
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library.  To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License.  (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.)  Do not make any other change in
these notices.

  Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
distribute the source code, even though third parties are not
compelled to copy the source along with the object code.

  5. A program that contains no derivative of any portion of the
Library, but is designed to work with the Library by being compiled or
linked with it, is called a "work that uses the Library".  Such a
work, in isolation, is not a derivative work of the Library, and
therefore falls outside the scope of this License.

  However, linking a "work that uses the Library" with the Library
creates an executable that is a derivative of the Library (because it
contains portions of the Library), rather than a "work that uses the
library".  The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.

  When a "work that uses the Library" uses material from a header file
that is part of the Library, the object code for the work may be a
```

```
derivative work of the Library even though the source code is not.
Whether this is true is especially significant if the work can be
linked without the Library, or if the work is itself a library.  The
threshold for this to be true is not precisely defined by law.

  If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work.  (Executables containing this object code plus portions of the
Library will still fall under Section 6.)

  Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

  6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.

  You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License.  You must supply a copy of this License.  If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License.  Also, you must do one
of these things:

    a) Accompany the work with the complete corresponding
    machine-readable source code for the Library including whatever
    changes were used in the work (which must be distributed under
    Sections 1 and 2 above); and, if the work is an executable linked
    with the Library, with the complete machine-readable "work that
    uses the Library", as object code and/or source code, so that the
    user can modify the Library and then relink to produce a modified
    executable containing the modified Library.  (It is understood
    that the user who changes the contents of definitions files in the
    Library will not necessarily be able to recompile the application
    to use the modified definitions.)

    b) Use a suitable shared library mechanism for linking with the
    Library.  A suitable mechanism is one that (1) uses at run time a
    copy of the library already present on the user's computer system,
    rather than copying library functions into the executable, and (2)
    will operate properly with a modified version of the library, if
    the user installs one, as long as the modified version is
    interface-compatible with the version that the work was made with.

    c) Accompany the work with a written offer, valid for at
    least three years, to give the same user the materials
    specified in Subsection 6a, above, for a charge no more
    than the cost of performing this distribution.

    d) If distribution of the work is made by offering access to copy
    from a designated place, offer equivalent access to copy the above
    specified materials from the same place.

    e) Verify that the user has already received a copy of these
    materials or that you have already sent this user a copy.

  For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it.  However, as a special exception,
the materials to be distributed need not include anything that is
```

```
normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.

  It may happen that this requirement contradicts the license
restrictions of other proprietary libraries that do not normally
accompany the operating system.  Such a contradiction means you cannot
use both them and the Library together in an executable that you
distribute.

  7. You may place library facilities that are a work based on the
Library side-by-side in a single library together with other library
facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
permitted, and provided that you do these two things:

    a) Accompany the combined library with a copy of the same work
    based on the Library, uncombined with any other library
    facilities.  This must be distributed under the terms of the
    Sections above.

    b) Give prominent notice with the combined library of the fact
    that part of it is a work based on the Library, and explaining
    where to find the accompanying uncombined form of the same work.

  8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License.  Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License.  However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

  9. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Library or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

  10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties with
this License.

  11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all.  For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.
```

```
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus
excluded.  In such case, this License incorporates the limitation as if
written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number.  If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation.  If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

  14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission.  For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this.  Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

       NO WARRANTY

  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU.  SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.
```

```
       END OF TERMS AND CONDITIONS

           How to Apply These Terms to Your New Libraries

  If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change.  You can do so by permitting
redistribution under these terms (or, alternatively, under the terms of the
ordinary General Public License).

  To apply these terms, attach the following notices to the library.  It is
safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

    &lt;one line to give the library's name and a brief idea of what it does.>
    Copyright (C) &lt;year>  &lt;name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James Random Hacker.

  &lt;signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!
```

A.6. The GFDL License

Source http://www.gnu.org/licenses/fdl-1.3.txt

```
                GNU Free Documentation License
                 Version 1.3, 3 November 2008


 Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
     <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other
functional and useful document "free" in the sense of freedom: to
```

```
assure everyone the effective freedom to copy and redistribute it,
with or without modifying it, either commercially or noncommercially.
Secondarily, this License preserves for the author and publisher a way
to get credit for their work, while not being considered responsible
for modifications made by others.

This License is a kind of "copyleft", which means that derivative
works of the document must themselves be free in the same sense.  It
complements the GNU General Public License, which is a copyleft
license designed for free software.

We have designed this License in order to use it for manuals for free
software, because free software needs free documentation: a free
program should come with manuals providing the same freedoms that the
software does.  But this License is not limited to software manuals;
it can be used for any textual work, regardless of subject matter or
whether it is published as a printed book.  We recommend this License
principally for works whose purpose is instruction or reference.


1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that
contains a notice placed by the copyright holder saying it can be
distributed under the terms of this License.  Such a notice grants a
world-wide, royalty-free license, unlimited in duration, to use that
work under the conditions stated herein.  The "Document", below,
refers to any such manual or work.  Any member of the public is a
licensee, and is addressed as "you".  You accept the license if you
copy, modify or distribute the work in a way requiring permission
under copyright law.

A "Modified Version" of the Document means any work containing the
Document or a portion of it, either copied verbatim, or with
modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of
the Document that deals exclusively with the relationship of the
publishers or authors of the Document to the Document's overall
subject (or to related matters) and contains nothing that could fall
directly within that overall subject.  (Thus, if the Document is in
part a textbook of mathematics, a Secondary Section may not explain
any mathematics.)  The relationship could be a matter of historical
connection with the subject or with related matters, or of legal,
commercial, philosophical, ethical or political position regarding
them.

The "Invariant Sections" are certain Secondary Sections whose titles
are designated, as being those of Invariant Sections, in the notice
that says that the Document is released under this License.  If a
section does not fit the above definition of Secondary then it is not
allowed to be designated as Invariant.  The Document may contain zero
Invariant Sections.  If the Document does not identify any Invariant
Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed,
as Front-Cover Texts or Back-Cover Texts, in the notice that says that
the Document is released under this License.  A Front-Cover Text may
be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy,
represented in a format whose specification is available to the
general public, that is suitable for revising the document
straightforwardly with generic text editors or (for images composed of
pixels) generic paint programs or (for drawings) some widely available
drawing editor, and that is suitable for input to text formatters or
for automatic translation to a variety of formats suitable for input
to text formatters.  A copy made in an otherwise Transparent file
```

```
format whose markup, or absence of markup, has been arranged to thwart
or discourage subsequent modification by readers is not Transparent.
An image format is not Transparent if used for any substantial amount
of text.  A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain
ASCII without markup, Texinfo input format, LaTeX input format, SGML
or XML using a publicly available DTD, and standard-conforming simple
HTML, PostScript or PDF designed for human modification.  Examples of
transparent image formats include PNG, XCF and JPG.  Opaque formats
include proprietary formats that can be read and edited only by
proprietary word processors, SGML or XML for which the DTD and/or
processing tools are not generally available, and the
machine-generated HTML, PostScript or PDF produced by some word
processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself,
plus such following pages as are needed to hold, legibly, the material
this License requires to appear in the title page.  For works in
formats which do not have any title page as such, "Title Page" means
the text near the most prominent appearance of the work's title,
preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of
the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose
title either is precisely XYZ or contains XYZ in parentheses following
text that translates XYZ in another language.  (Here XYZ stands for a
specific section name mentioned below, such as "Acknowledgements",
"Dedications", "Endorsements", or "History".)  To "Preserve the Title"
of such a section when you modify the Document means that it remains a
section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which
states that this License applies to the Document.  These Warranty
Disclaimers are considered to be included by reference in this
License, but only as regards disclaiming warranties: any other
implication that these Warranty Disclaimers may have is void and has
no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either
commercially or noncommercially, provided that this License, the
copyright notices, and the license notice saying this License applies
to the Document are reproduced in all copies, and that you add no
other conditions whatsoever to those of this License.  You may not use
technical measures to obstruct or control the reading or further
copying of the copies you make or distribute.  However, you may accept
compensation in exchange for copies.  If you distribute a large enough
number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and
you may publicly display copies.


3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have
printed covers) of the Document, numbering more than 100, and the
Document's license notice requires Cover Texts, you must enclose the
copies in covers that carry, clearly and legibly, all these Cover
Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on
the back cover.  Both covers must also clearly and legibly identify
you as the publisher of these copies.  The front cover must present
the full title with all words of the title equally prominent and
visible.  You may add other material on the covers in addition.
```

```
Copying with changes limited to the covers, as long as they preserve
the title of the Document and satisfy these conditions, can be treated
as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit
legibly, you should put the first ones listed (as many as fit
reasonably) on the actual cover, and continue the rest onto adjacent
pages.

If you publish or distribute Opaque copies of the Document numbering
more than 100, you must either include a machine-readable Transparent
copy along with each Opaque copy, or state in or with each Opaque copy
a computer-network location from which the general network-using
public has access to download using public-standard network protocols
a complete Transparent copy of the Document, free of added material.
If you use the latter option, you must take reasonably prudent steps,
when you begin distribution of Opaque copies in quantity, to ensure
that this Transparent copy will remain thus accessible at the stated
location until at least one year after the last time you distribute an
Opaque copy (directly or through your agents or retailers) of that
edition to the public.

It is requested, but not required, that you contact the authors of the
Document well before redistributing any large number of copies, to
give them a chance to provide you with an updated version of the
Document.


4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under
the conditions of sections 2 and 3 above, provided that you release
the Modified Version under precisely this License, with the Modified
Version filling the role of the Document, thus licensing distribution
and modification of the Modified Version to whoever possesses a copy
of it.  In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct
   from that of the Document, and from those of previous versions
   (which should, if there were any, be listed in the History section
   of the Document).  You may use the same title as a previous version
   if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities
   responsible for authorship of the modifications in the Modified
   Version, together with at least five of the principal authors of the
   Document (all of its principal authors, if it has fewer than five),
   unless they release you from this requirement.
C. State on the Title page the name of the publisher of the
   Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications
   adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice
   giving the public permission to use the Modified Version under the
   terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections
   and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add
   to it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page.  If
   there is no section Entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
```

```
   it was based on.  These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles.  Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements".  Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant.  To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version.  Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity.  If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.


5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and
multiple identical Invariant Sections may be replaced with a single
copy.  If there are multiple Invariant Sections with the same name but
different contents, make the title of each such section unique by
adding at the end of it, in parentheses, the name of the original
author or publisher of that section if known, or else a unique number.
Make the same adjustment to the section titles in the list of
Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History"
in the various original documents, forming one section Entitled
"History"; likewise combine any sections Entitled "Acknowledgements",
and any sections Entitled "Dedications".  You must delete all sections
Entitled "Endorsements".
```

```
6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other
documents released under this License, and replace the individual
copies of this License in the various documents with a single copy
that is included in the collection, provided that you follow the rules
of this License for verbatim copying of each of the documents in all
other respects.

You may extract a single document from such a collection, and
distribute it individually under this License, provided you insert a
copy of this License into the extracted document, and follow this
License in all other respects regarding verbatim copying of that
document.


7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate
and independent documents or works, in or on a volume of a storage or
distribution medium, is called an "aggregate" if the copyright
resulting from the compilation is not used to limit the legal rights
of the compilation's users beyond what the individual works permit.
When the Document is included in an aggregate, this License does not
apply to the other works in the aggregate which are not themselves
derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these
copies of the Document, then if the Document is less than one half of
the entire aggregate, the Document's Cover Texts may be placed on
covers that bracket the Document within the aggregate, or the
electronic equivalent of covers if the Document is in electronic form.
Otherwise they must appear on printed covers that bracket the whole
aggregate.


8. TRANSLATION

Translation is considered a kind of modification, so you may
distribute translations of the Document under the terms of section 4.
Replacing Invariant Sections with translations requires special
permission from their copyright holders, but you may include
translations of some or all Invariant Sections in addition to the
original versions of these Invariant Sections.  You may include a
translation of this License, and all the license notices in the
Document, and any Warranty Disclaimers, provided that you also include
the original English version of this License and the original versions
of those notices and disclaimers.  In case of a disagreement between
the translation and the original version of this License or a notice
or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements",
"Dedications", or "History", the requirement (section 4) to Preserve
its Title (section 1) will typically require changing the actual
title.


9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense, or distribute it is void, and
will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license
from a particular copyright holder is reinstated (a) provisionally,
```

unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

```
ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of
the License in the document and put the following copyright and
license notices just after the title page:

    Copyright (c)  YEAR  YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.3
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other
combination of the three, merge those two alternatives to suit the
situation.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```

Appendix B. Core tasks

This appendix contains a reference of all core tasks, i.e. all tasks that are needed to build a basic project.

This reference lists the tasks alphabetically by the name of the classes that implement the tasks. So if you are searching for the reference to the <copy> tag, for example, you will want to look at the reference of CopyTask.

B.1. AdhocTaskdefTask

The AdhocTaskdefTask allows you to define a task within your build file.

Note that you should use <![CDATA[...]]> so that you don't have to quote entities within your <adhoc-task></adhoc-task> tags.

Table B.1: *Attributes*

Name	Type	Description	Default	Required
name	String	Name of XML tag that will represent this task.	n/a	Yes

B.1.1. Examples

```
<target name="main"
        description="==>test AdhocTask ">

  <adhoc-task name="foo"><![CDATA[
   class FooTest extends Task {
    private $bar;

    function setBar($bar) {
     $this->bar = $bar;
    }

    function main() {
     $this->log("In FooTest: " . $this->bar);
    }
   }
  ]]></adhoc-task>

  <foo bar="B.L.I.N.G"/>
</target>
```

B.2. AdhocTypedefTask

The AdhocTypedefTask allows you to define a datatype within your build file.

Note that you should use <![CDATA[...]]> so that you don't have to quote entities within your <adhoc-type></adhoc-type> tags.

Table B.2: *Attributes*

Name	Type	Description	Default	Required
name	String	Name of XML tag that will represent this datatype..	n/a	Yes

B.2.1. Example

```
<target name="main"
        description="==>test AdhocType">

  <adhoc-type name="dsn"><![CDATA[
   class CreoleDSN extends DataType {
    private $url;

    function setUrl($url) {
     $this->url = $url;
    }

    function getUrl() {
      return $this->url;
    }
   }
  ]]></adhoc-type>

     <!-- creole-sql task doesn't exist; just an example -->
     <creole-sql file="test.sql">
       <dsn url="mysql://root@localhost/test"/>
  </creole-sql>

</target>
```

B.3. AppendTask

The Append Task appends text or contents of files to a specified file.

In the example above, AppendTask is reading a filename from `book/PhingGuide.book`, processing the file contents with XSLT, and then appending the result to the file located at `${process.outputfile}`. This is a real example from the build file used to generate this book!

Table B.3: *Attributes*

Name	Type	Description	Default	Required
`destFile`	`File`	Path of file to which text should be appended.	n/a	Yes
`file`	`File`	Path to file that should be appended to dest-File.	n/a	No
`text`	`String`	Some literal text to append to file.	n/a	No

B.3.1. Examples

```
<append destFile = "${process.outputfile}">
  <filterchain>
    <xsltfilter style = "${process.stylesheet}">
      <param name = "mode" expression = "${process.xslt.mode}"/>
    </xsltfilter>
  </filterchain>
  <filelist dir = "book/" listfile = "book/PhingGuide.book"/>
</append>
```

B.3.2. Supported Nested Tags

- `filelist`

- `fileset`
- `filterchain`

B.4. ApplyTask

Applies a system command on each resource of the specified resource collection.

When the os attribute is specified, then the command is only executed when run on one of the specified operating systems.

The files of a number of Resource Collections – including but not restricted to FileSets or FileLists – are passed as arguments to the system command.

Table B.4: *Attributes*

Name	Type	Description	Default	Required	Alias
`executable`	`String`	The command to execute without any command line arguments.	n/a	Yes	
`dir`	`String`	The directory the command is to be executed in.	n/a	No	
`output`	`String`	Where to direct stdout.	n/a	No	
`error`	`String`	Where to direct stderr.	n/a	No	
`os`	`String`	Only execute if the Appendix A, *Fact Sheet* property contains specified text.	n/a	No	
`escape`	`Boolean`	Escape shell metacharacters before execution. Setting this to true will enable the escape precaution.	`false`	No	
`passthru`	`Boolean`	Whether to use PHP's passthru() function instead of exec().	`false`	No	
`spawn`	`Boolean`	Whether to spawn unix programs to the background, redirecting stdout.	`false`	No	
`returnProperty`	`String`	Property name to set return value to from the execution.	n/a	No	
`outputProperty`	`String`	Property name to set output value to from the execution.	n/a	No	
`checkreturn`	`Boolean`	Whether to check the return code of the execution, throws a BuildException when returncode != 0.	`false`	No	failonerror
`append`	`Boolean`	Whether output (and error) should be appended to or overwrite an existing file. If you set parallel to false, you will probably want to set this one to true.	`false`	No	
`parallel`	`Boolean`	Run the command only once, appending all files as arguments. If false, command will be executed once for every file.	`false`	No	
`addsourcefile`	`Boolean`	Whether source file name(s) should be added to the end of command-line auto-	`true`	No	

Name	Type	Description	Default	Required	Alias
		matically. If you need to place it somewhere different, use a nested <srcfile> element between your <arg> elements to mark the insertion point.			
spawn	Boolean	Whether the command(s) to be executed are spawned as background process. If a command is spawned, its output will not be logged by Phing.	false	No	
relative	Boolean	Whether the filenames should be passed on the command line as relative pathnames (relative to the base directory of the corresponding fileset/list for source files).	false	No	
forwardslash	Boolean	Whether the file names should be passed with forward slashes even if the operating system requires other file separator.	false	No	
maxparallel	Boolean	Limit the amount of parallelism by passing at most this many sourcefiles at once. Set it to <= 0 for unlimited.	false	No	

B.4.1. Examples

```
<!-- Invokes somecommand arg1 SOURCEFILENAME arg2 for each file in /tmp -->
<apply executable = "somecommand" parallel = "false">
  <arg value = "arg1"/>
  <srcfile/>
  <arg value = "arg2"/>
  <fileset dir = "/tmp"/>
</apply>


<!-- List all the .conf files of "/etc" to the "out.log" file. -->
<apply executable = "ls" output = "/tmp/out.log" append = "true" >
  <arg value = "-alh" />
  <fileset dir = "/etc" >
    <include name = "*.conf" />
  </fileset>
</apply>
```

B.4.2. Supported Nested Tags

- arg

Table B.5: *Attributes*

Name	Type	Description	Default	Required
value	String	A single command-line argument; can contain space characters.	n/a	One of these

Name	Type	Description	Default	Required
`file`	`String`	The name of a file as a single command-line argument; will be replaced with the absolute filename of the file.	n/a	
`path`	`String`	A string that will be treated as a path-like string as a single command-line argument; you can use ; or : as path separators and Ant will convert it to the platform's local conventions.	n/a	
`line`	`String`	A space-delimited list of command-line arguments.	n/a	

- `fileset`
- `filelist`
- `srcfile`

B.5. AvailableTask

Available Task tests if a resource/file is set and sets a certain property to a certain value if it exists.

Here, `AvailableTask` first checks for the existance of either file or directory named `test.txt` in `/tmp`. Then, it checks for the directory `foo` in `/home` and then for the file or directory `bar` in `/home/foo`. If `/tmp/test.txt` is found, the property `test_txt_exists` is set to `"Yes"`, if `/home/foo` is found and a directory, `properties.yetanother` is set to `"true"` (default). If `/home/foo/bar` exists, `AvailableTask` will set `foo.bar` to `"Well, yes"`. And last it checks if extension `foo` is loaded, so the property `foo.ext.loaded` is set to `"true"` (default). **k**

NB: the Available task can also be used as a Chapter 5, *Project components*.

Table B.6: *Attributes*

Name	Type	Description	Default	Required
`property`	`String`	Name of the property that is to be set.	n/a	Yes
`value`	`String`	The value the property is to be set to.	`"true"`	No
`file`	`String`	File/directory to check existence.	n/a	Yes (or `resource` or `extension`)
`resource`	`String`	Path of the resource to look for.	n/a	Yes (or `file` or `extension`)
`extension`	`String`	Name of the extension to look for.	n/a	Yes (or `file` or `resource`)
`type`	`String` (file|dir)	Determines if `AvailableTask` should look for a file or a directory at the position set by `file`. If empty, it checks for either file or directory.	n/a	No
`filepath`	`String`	The path to use when looking up `file`.	n/a	No
`followSymlinks`	`Boolean`	Whether to dereference symbolic links when looking up `file`.	`false`	No

B.5.1. Examples

```
<available file = "/tmp/test.txt" property = "test_txt_exists" value = "Yes"/>

<available file = "/home/foo" type = "dir" property = "properties.yetanother" />

<available file = "/home/foo/bar" property = "foo.bar" value = "Well, yes" />
```

B.6. ChmodTask

Sets the mode of a file or directory.

For more informations, see chmod [http://php.net/chmod] in the PHP Manual.

Table B.7: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The name of the file or directory. You either have to specify this attribute, or use a fileset.	n/a	Yes
`mode`	`String`	The new mode (octal) for the file. Specified in octal, even if the first digit is not a '0'.	n/a	Yes
`quiet`	`Boolean`	Set quiet mode, which suppresses warnings if `chmod()` fails	`false`	No
`failonerror`	`Boolean`	This flag means 'note errors to the output, but keep going'	`true`	No
`verbose`	`Boolean`	Give more information in error message in case of a failure	`true`	No

B.6.1. Examples

```
<chmod file = "test.txt" mode = "0755" />
<chmod file = "/home/test" mode = "0775" />
<chmod file = "/home/test/mine.txt" mode = "0500" verbose = "true" />
```

B.6.2. Supported Nested Tags

- `fileset`

B.7. ChownTask

Changes the owner of a file or directory.

Table B.8: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The name of the file or directory. You either have to specify this attribute, or use a fileset.	n/a	Yes

Name	Type	Description	Default	Required
`user`	`String`	The new owner of the file. Can contain a user-name and a groupname, separated by a dot.	n/a	No
`group`	`String`	The new group owner of the file.	n/a	No
`quiet`	`Boolean`	Set quiet mode, which suppresses warnings if `chmod()` fails	`false`	No
`failonerror`	`Boolean`	This flag means 'note errors to the output, but keep going'	`true`	No
`verbose`	`Boolean`	Give more information in error message in case of a failure	`true`	No

B.7.1. Examples

```
<chown file = "my-file.txt" user = "foo" />
<chown file = "my-file.txt" user = "username.groupname" />
<chown file = "/home/test/my-directory" user = "bar" />
<chown file = "/home/test/my-file.txt" user = "foo" verbose = "true" failonerror = "false" />
```

B.7.2. Supported Nested Tags

- `fileset`

B.8. ConditionTask

Sets a property if a certain condition holds true - this is a generalization of Section B.5, “AvailableTask ” and Section B.36, “UpToDateTask ”.

If the condition holds true, the property value is set to true by default; otherwise, the property is not set. You can set the value to something other than the default by specifying the `value` attribute.

Conditions are specified as nested elements, you must specify exactly one condition - see the Chapter 5, *Project components* for a complete list of nested elements.

Table B.9: *Attributes*

Name	Type	Description	Default	Required
`property`	`String`	The name of the property to set.	n/a	Yes
`value`	`String`	The value to set the property to. Defaults to "true".	`true`	No

B.8.1. Examples

```
<condition property = "isMacOrWindows">
    <or>
      <os family = "mac"/>
      <os family = "windows"/>
    </or>
```

```
</condition>
```

B.8.2. Supported Nested Tags

- `or`
- `and`

B.9. CopyTask

Copies files or directories. 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 explictly overwrite existing files.

`CopyTask` does not allow self copying, i.e. copying a file to the same name for security reasons.

Table B.10: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The source file.	Yes	
`tofile`	`String`	The destination the file is to be written to. `tofile` specifies a `full` filename. If you only want to specify a directory to copy to, use `todir`. Either this or the `todir` attribute is required.	n/a	Yes (or `todir`)
`todir`	`String`	The directory the file is to be copied to. The file will have the same name of the source file. If you want to specify a different name, use `tofile`. The directory must exist.	n/a	Yes (or `tofile`)
`overwrite`	`Boolean`	Overwrite existing files even if the destination files are newer.	`false`	No
tstamp or preservelastmodified	`Boolean`	If set to `true`, the new file will have the same mtime as the old one.	`false`	No
preservemode or preservepermissions	`Boolean`	If set to `true`, the new file (and directory) will have the same permissions as the old one. The `mode` specified for directory creation will be ignored.	`true`	No
`includeemptydirs`	`Boolean`	If set to `true`, also empty directories are copied.	`true`	No
`mode`	`Integer`	Mode (octal) to create directories with.	From umask	No
`haltonerror`	`Boolean`	If set to `true`, halts the build when errors are encountered.	`true`	No

Note

No automatic exapnsion of symbolic links

By default, `CopyTask` does not expand / dereference symbolic links, and will simply copy the link itself. To enable dereferencing, set `expandsymboliclinks` to `true` in the `<fileset>` tag.

B.9.1. Examples

On the one hand, CopyTask can be used to copy file by file:

```
<copy file = "somefile.txt" tofile = "/tmp/anotherfile.bak" overwrite = "true"/>
```

Additionally, `CopyTask` supports Filesets, i.e. you can easily include/exclude one or more files. For more information, see Appendix D, *Core Types* -- pay particular attention to the `defaultexcludes` attribute. Appendix F, *Core mappers* and Appendix E, *Core filters* are also supported by `CopyTask`, so you can do almost everything that needs processing the content of the files or the filename.

```
<copy todir = "/tmp/backup" >
  <fileset dir = ".">
    <include name = "**/*.txt" />
    <include name = "**/*.doc" />
    <include name = "**/*.swx" />
  </fileset>
  <filelist dir = "." files = "test.html"/>
</copy>
```

```
<copy todir = "build" >
  <fileset defaultexcludes = "false" expandsymboliclinks = "true" dir = ".">
    <include name = "**/*.php" />
  </fileset>
</copy>
```

B.9.2. Supported Nested Tags

- `fileset`
- `filelist`
- `filterchain`
- `mapper`

B.10. CvsTask

Allows rudimentary interfacing with the CVS versioning system.

As you would expect, this lets you do pretty much anything with CVS. The CvsTask calls ExecTask which actually does the command execution.

Table B.11: *Attributes*

Name	Type	Description	Default	Required
`cvsRoot`	`String`	The root directory on the CVS server	n/a	No
`CvsRsh`	`String`	Path to the rsh to execute	n/a	No

Name	Type	Description	Default	Required
port	Integer	Port ion the server to use	0	No
passfile	String file-name	Name of file with CVS passwords	n/a	No
dest	String	The directory where checked out files should be placed	n/a	Yes
modules	String	The package/module to operate upon	n/a	Yes
tag	String	The tag of the package/module to operate upon	n/a	No
date	String	Use the most recent revision no later than the given date	n/a	No
quiet	Boolean	Quiet run	false	No
noexec	Boolean	If true, only report changes don't actually do anything	false	No
failonerror	Boolean	Stop the build process if the command returns any errors	false	No
compression	Boolean	If true, turns on compression using default (3) compression level	false	No
compression-level	Integer	Specifies compression level 1-9	false	No
output	String file-name	File to which output should be written	n/a	No
error	String file-name	File to which error output should be written	n/a	No
command	String	Optional command line to be given to the CVS task	null	No

B.10.1. Examples

```
<cvs cvsRoot = ":pserver:guest@cvs.tigris.org:/cvs"
      module = "phing" dest = "${ws.dir}"/>
```

or, using the optional command line arguments:

```
<cvs output = "patch" command = "-q diff -u -N" />
```

B.11. CvsPassTask

This lets you create your own cvs password file -- i.e. this is necessary if you want to provide a password in your build file.

If no password file is specified the default location `.cvspass` in users home directory is used.

Table B.12: *Attributes*

Name	Type	Description	Default	Required
`cvsRoot`	`String`	The root directory on the CVS server	n/a	Yes
`passFile`	`String`	Password file to add password to	n/a	No
`password`	`String`	Password to add to file	n/a	Yes

B.11.1. Examples

```
<!-- create a password file -->
<cvspass cvsRoot = ":pserver:guest@cvs.tigris.org:/cvs" passfile = "cvspass" password = "guest"

<!-- use the just-created password file -->
<cvs cvsRoot = ":pserver:guest@cvs.tigris.org:/cvs" module = "phing" passFile = "cvspass" />
```

B.12. DeleteTask

Deletes a file or directory, or set of files defined by a fileset. See Appendix D, *Core Types* for information on Filesets.

Table B.13: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The file that is to be deleted. You either have to specify this attribute, `dir`, or use a fileset.	n/a	Yes (or `dir`)
`dir`	`String`	The directory that is to be deleted. You either have to specify this attribute, `file`, or use a fileset.	n/a	Yes (or `file`)
`verbose`	`Boolean`	Used to force listing of all names of deleted files.	n/a	No
`quiet`	`Boolean`	If 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 verbose	n/a	No
`failonerror`	`Boolean`	If this attribute is set to `true`, DeleteTask will verbose on errors but the build process will not be stopped.	`false`	No
`includeemptydirs`	`Boolean`	Determines if empty directories are also to be deleted.	`false`	No

B.12.1. Examples

```
<!-- Delete a specific file -->
```

```
<delete file = "/tmp/foo.bar" />

<!-- Delete a directory -->
<delete dir = "/tmp/darl" includeemptydirs = "true" verbose = "true" failonerror = "true" />

<!-- Delete using a fileset -->
<delete>
  <fileset dir = "/tmp">
    <include name = "*.bar" />
  </fileset>
</delete>
```

B.12.2. Supported Nested Tags

- `fileset`

B.13. EchoTask

Echoes a message to the current loggers and listeners which means standard out unless overridden. A level can be specified, which controls at what logging level the message is filtered at.

The task can also echo to a file, in which case the option to append rather than overwrite the file is available, and the level option is ignored

Additionally, the task can echo the contents of a nested fileset element.

Table B.14: *Attributes*

Name	Type	Description	Default	Required
`msg`	`String`	The string that is to be send to the output.	n/a	Yes
`message`	`String`	Alias for `msg`.	n/a	Yes
`file`	`String`	The file to write the message to.	n/a	No
`append`	`Boolean`	Append to an existing file?	`false`	No
`level`	`String`	Control the level at which this message is reported. One of `error`, `warning`, `info`, `verbose`, `debug`.	`info`	No

B.13.1. Examples

```
<echo msg = "Phing rocks!" />

<echo message = "Binarycloud, too." />

<echo>And don't forget Propel.</echo>

<echo file = "test.txt" append = "false">This is a test message</echo>
```

B.13.2. Supported Nested Tags

- `fileset`

B.14. ExecTask

Executes a shell command. You can use this to quickly add a new command to Phing. However, if you want to use this regularly, you should think about writing a Task for it.

Table B.15: *Attributes*

Name	Type	Description	Default	Required
`command`	`String`	The command that is to be executed.	n/a	One of the two
`executable`	`String`	The command to execute without any command line arguments.	n/a	
`dir`	`String`	The directory the command is to be executed in.	n/a	No
`output`	`String`	Where to direct stdout.	n/a	No
`error`	`String`	Where to direct stderr.	Redirected to stdout, unless `passthru` is set to true.	No
`os`	`String`	Only execute if the Appendix A, *Fact Sheet* property contains specified text.	n/a	No
`escape`	`Boolean`	By default, we escape shell metacharacters before executing. Setting this to false will disable this precaution.	`true`	No
`passthru`	`Boolean`	Whether to use PHP's passthru() function instead of exec().	`false`	No
`logoutput`	`Boolean`	Whether to log returned output as MSG_INFO instead of MSG_VERBOSE.	`false`	No
`spawn`	`Boolean`	Whether to spawn unix programs to the background, redirecting stdout.	`false`	No
`returnProperty`	`String`	Property name to set return value to from exec() call.	n/a	No
`outputProperty`	`String`	Property name to set output value to from exec() call.	n/a	No
`checkreturn`	`Boolean`	Whether to check the return code of the program, throws a BuildException when return-code != 0.	`false`	No
`level`	`String`	Control the level at which status messages are reported. One of `error`, `warning`, `info`, `verbose`, `debug`.	`verbose`	No

B.14.1. Examples

```
<!-- List the contents of "/home". -->
<exec command = "ls -l" dir = "/home" />

<!-- Start the make process in "/usr/src/php-4.0". -->
```

```
<exec command = "make" dir = "/usr/src/php-4.0" />

<!-- List the contents of "/tmp" out to a file. -->
<exec command = "ls -l /tmp > foo.out" escape = "false" />
```

B.14.2. Supported Nested Tags

- `arg`

Table B.16: *Attributes*

Name	Type	Description	Default	Required
`value`	`String`	A single command-line argument; can contain space characters. To pass an empty argument, enclose two double quotes in single quotes ("""").	n/a	One of these
`file`	`String`	The name of a file as a single command-line argument; will be replaced with the absolute filename of the file.	n/a	
`path`	`String`	A string that will be treated as a path-like string as a single command-line argument; you can use ; or : as path separators and Ant will convert it to the platform's local conventions.	n/a	
`line`	`String`	A space-delimited list of command-line arguments.	n/a	

B.15. FailTask

Causes the current build script execution to fail and the script to exit with an (optional) error message.

Table B.17: *Attributes*

Name	Type	Description	Default	Required
`message`	`String`	The message to display (reason for script abort).	"No Message"	No
`msg`	`String`	Alias for message	"No Message"	No
`if`	`String`	Name of property that must be set for script to exit.	n/a	No
`unless`	`String`	Name of property that must `not` be set in order for script to exit.	n/a	No

B.15.1. Examples

```
<!-- Exit w/ message -->
<fail message = "Failed for some reason!" />

<!--  Exit if ${errorprop} is defined -->
<fail if = "errorprop" message = "Detected error!" />
```

```
<!-- Exit unless ${dontfail} prop is defined. -->
<fail unless = "dontfail" message = "Detected error!" />
```

B.16. ForeachTask

The `foreach` task iterates over a list, a list of `filesets`, or both. If both, `list` and `filesets`, are specified, the list will be evaluated first. Nested `filesets` are evaluated in the order they appear in the task.

Table B.18: *Attributes*

Name	Type	Description	Default	Required
`list`	`String`	The list of values to process, with the delimiter character, indicated by the "delimiter" attribute, separating each value.	n/a	No
`target`	`String`	The target to call for each token, passing the token as the parameter with the name indicated by the "param" attribute.	n/a	Yes
`param`	`String`	The name of the parameter to pass the tokens in as to the target.	n/a	Yes
`absparam`	`String`	The name of the absolute pathparameter to pass the tokens in as to the target (used while processing nested filesets).	n/a	No
`delimiter`	`String`	The delimiter string that separates the values, in the "list" parameter. The default is ",".		No

B.16.1. Examples

```
<!-- loop through languages, and call buildlang task with setted param -->
<property name = "languages" value = "en,fr,de" />
<foreach list = "${languages}" param = "lang" target = "buildlang" />

<!-- loop through files, and call subtask task with set param and absparam -->
<foreach param = "filename" absparam = "absfilename" target = "subtask">
  <fileset dir = ".">
    <include name = "*.php"/>
  </fileset>
</foreach>
```

B.16.2. Supported Nested Tags

- `fileset`
- `mapper`

B.17. IfTask

Perform some tasks based on whether a given condition holds true or not.

This task doesn't have any attributes, the condition to test is specified by a nested element - see the Chapter 5, *Project components* for a complete list of nested elements.

Just like the `<condition>` task, only a single condition can be specified - you combine them using `<and>` or `<or>` conditions.

In addition to the condition, you can specify three different child elements, `<elseif>` , `<then>` and `<else>` . All three subelements are optional. Both `<then>` and `<else>` must not be used more than once inside the if task. Both are containers for Phing tasks.

The `<elseif>` behaves exactly like an `<if>` except that it cannot contain the `<else>` element inside of it. You may specify as may of these as you like, and the order they are specified is the order they are evaluated in. If the condition on the `<if>` is false, then the first `<elseif>` who's conditional evaluates to true will be executed. The `<else>` will be executed only if the `<if>` and all `<elseif>` conditions are false.

B.17.1. Examples

```
<if>
 <equals arg1 = "${foo}" arg2 = "bar" />
 <then>
   <echo message = "The value of property foo is bar" />
 </then>
 <else>
   <echo message = "The value of property foo is not bar" />
 </else>
</if>
```

```
<if>
 <equals arg1 = "${foo}" arg2 = "bar" />
 <then>
   <echo message = "The value of property foo is 'bar'" />
 </then>

 <elseif>
  <equals arg1 = "${foo}" arg2 = "foo" />
  <then>
   <echo message = "The value of property foo is 'foo'" />
  </then>
 </elseif>

 <else>
   <echo message = "The value of property foo is not 'foo' or 'bar'" />
 </else>
</if>
```

B.18. ImportTask

Imports another build file into the current project.

On execution it will read another Phing file into the same Project. Functionally it is nearly the same as copy and pasting the imported file onto the end of the importing file.

Table B.19: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The file to import.	n/a	Yes

Name	Type	Description	Default	Required
`optional`	`Boolean`	If true, do not stop the build if the file does not exist.	`false`	No

B.18.1. Target Overriding

If a target in the main file is also present in at least one of the imported files, the one from the main file takes precedence.

So if I import for example a `docs/build.xml` file named `builddocs`, that contains a "`docs`" target, I can redefine it in my main buildfile and that is the one that will be called. This makes it easy to keep the same target name, so that the overriding target is still called by any other targets--in either the main or imported buildfile(s)--for which it is a dependency, with a different implementation. The target from `docs/build.xml` is made available by the name "`builddocs.docs`". This enables the new implementation to call the old target, thus enhancing it with tasks called before or after it.

B.18.2. Special Properties

Imported files are treated as they are present in the main buildfile. This makes it easy to understand, but it makes it impossible for them to reference files and resources relative to their path. Because of this, for every imported file, Phing adds a property that contains the path to the imported buildfile. With this path, the imported buildfile can keep resources and be able to reference them relative to its position.

So if I import for example a `docs/build.xml` file named `builddocs`, I can get its path as `phing.file.builddocs`, similarly to the `phing.file` property of the main buildfile. Additionally, the directory will be stored in `phing.dir.builddocs`.

Note that "builddocs" is not the filename, but the name attribute present in the imported project tag.

If import file does not have a name attribute, the `phing.file.projectname` and `phing.dir.projectname` properties will not be set.

B.18.3. Resolving Files Against the Imported File

Suppose your main build file called `importing.xml` imports a build file `imported.xml` , located anywhere on the file system, and `imported.xml` reads a set of properties from `imported.properties` :

```
<!-- importing.xml -->
<project name = "importing" basedir = "." default = "...">
  <import file = "${path_to_imported}/imported.xml"/>
</project>

<!-- imported.xml -->
<project name = "imported" basedir = "." default = "...">
  <property file = "imported.properties"/>
</project>
```

This snippet however will resolve `imported.properties` against the basedir of `importing.xml` , because the basedir of `imported.xml` is ignored by Phing. The right way to use `imported.properties` is:

```
<!-- imported.xml -->
<project name = "imported" basedir = "." default = "...">
  <property file = "${phing.file.imported}/imported.properties"/>
</project>
```

As explained above `${phing.file.imported}` stores the full path of the build script, that defines the project called *imported*, (in short it stores the path to `imported.xml`) and `${phing.dir.imported}` stores its directory. This technique also allows `imported.xml` to be used as a standalone file (without being imported in other project).

B.18.4. Examples

```
<import file = "path/to/build.xml"/>
<import file = "path/to/build.xml" optional = "true"/>
```

B.19. IncludePathTask

Sets the PHP `include_path` configuration option for the duration of this phing run.

Table B.20: *Attributes*

Name	Type	Description	Default	Required
`classpath`	`String`	the new include path[s]	n/a	Yes
`classPathRef`	`String`	Reference to a previously defined Path type	n/a	No

B.19.1. Examples

```
<includepath classpath = "new/path/here" />
<includepath classpath = "path1:path2" />
```

```
<path id = "project.class.path">
  <pathelement dir = "lib/"/>
  <pathelement dir = "ext/"/>
</paentry>
<includepath classpathref = "project.class.path" />
```

B.20. InputTask

The `InputTask` can be used to interactively set property values based on input from the console (or other Reader).

Table B.21: *Attributes*

Name	Type	Description	Default	Required
`propertyName`	`String`	The name of the property to set.	n/a	No
`defaultValue`	`String`	The default value to be set if no new value is provided.	n/a	Yes
`message`	`String`	Prompt text (same as CDATA).	n/a	Yes
`promptChar`	`String`	The prompt character to follow prompt text.	n/a	No
`validArgs`	`String`	Comma-separated list of valid choices the user must supply. If used, one of these options must be chosen.	n/a	No

B.20.1. Examples

```
<!-- Getting string input -->
<echo>HTML pages installing to: ${documentRoot}</echo>
<echo>PHP classes installing to: ${servletDirectory}</echo>

<input propertyname = "documentRoot">Web application document root</input>
<input propertyname = "servletDirectory"
  defaultValue = "/usr/servlets" promptChar = "?">PHP classes install dir</input>

<echo>HTML pages installed to ${documentRoot}</echo>
<echo>PHP classes installed to ${servletDirectory}</echo>

<!-- Having the user choose from a set of valid choices -->
<echo>Choose a valid option:</echo>

<input propertyname = "optionsChoice" validargs = "foo,bar,bob">
  Which item would you like to use
</input>
```

B.21. LoadFileTask

The `LoadFileTask` loads the contents of a (text) file into a single property.

Table B.22: *Attributes*

Name	Type	Description	Default	Required
`property`	`String`	The name of the property to set.	n/a	Yes
file (or `srcFile`)	`String`	The file to load.	n/a	Yes

B.21.1. Examples

```
<loadfile property = "version" file = "version.txt"/>
```

B.21.2. Supported Nested Tags:

- `filterchain`

B.22. MkdirTask

Creates a directory, including any necessary but non-existent parent directories. Does nothing if the directory already exists.

Table B.23: *Attributes*

Name	Type	Description	Default	Required
`dir`	`String`	The directory that is to be created.	n/a	Yes
`mode`	`Integer`	The mode to create the directory with.	From umask	No

B.22.1. Examples

```
<!-- Create a temp directory -->
<mkdir dir = "/tmp/foo" />

<!-- Using mkdir with a property -->
<mkdir dir = "${dirs.install}/tmp" />
```

B.23. MoveTask

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 if the file or directory has been copied to the destination successfully.

B.23.1. Examples

```
<!-- The following will move the file "somefile.txt" to "/tmp" and
    change its filename to "anotherfile.bak". It will overwrite
    an existing file. -->
<move file = "somefile.txt" tofile = "/tmp/anotherfile.bak" overwrite = "true"/>

<!-- This will move the "/tmp" directory to "/home/default/tmp",
    preserving the directory name. So the final name is
    "/home/default/tmp/tmp". Empty directories are also copied -->
<move file = "/tmp" todir = "/home/default/tmp" includeemptydirs = "true" />
```

B.23.2. Attributes and Nested Elements

For further documentation, see Section B.9, “CopyTask ”, since MoveTask only is a child of CopyTask and inherits all attributes.

B.24. PhingTask

This task calls another build file. You may specify the target that is to be called within the build file. Additionally, the <phing> Tag may contain <property> Tags (see Section B.27, “PropertyTask ”).

Table B.24: *Attributes*

Name	Type	Description	Default	Required
inheritAll	Boolean	If true, pass all properties to the new phing project.	true	No
inheritRefs	Boolean	If true, pass all references to the new phing project.	false	No
dir	String	The directory to use as a base directory for the new phing project. Default is the current project's basedir, unless inheritall has been set to false, in which case it doesn't have	n/a	No

Name	Type	Description	Default	Required
		a default value. This will override the basedir setting of the called project.		
phingFile	String	The build file to use. Defaults to "build.xml". This file is expected to be a filename relative to the dir attribute given.	n/a	Yes
target	String	The target of the new Phing project to execute. Default is the new project's default target.	n/a	No
haltonfailure	Boolean	If true, fail the build process when the called build fails	false	No

B.24.1. Examples

```
<!-- Call target "xslttest" from buildfile "alternativebuildfile.xml" -->
 <phing phingfile = "alternativebuild.xml" inheritRefs = "true" target = "xslttest" />

<!-- Do a more complex call -->
<phing phingfile = "somebuild.xml" target = "sometarget">
  <property name = "foo" value = "bar" />
  <property name = "anotherone" value = "32" />
</phing>
```

B.24.2. Supported Nested Tags

- fileset

B.24.3. Base directory of the new project

The base directory of the new project is set dependant on the dir and the inheritAll attribute. This is important to keep in mind or else you might run into bugs in your build.xml's. The following table shows when which value is used:

Table B.25: *How attributes are used*

dir	inheritAll	new project's basedir
value provided	true	value of dir attribute
value provided	false	value of dir attribute
omitted	true	basedir of calling task (the build file containing the <phing> call.
omitted	false	basedir attribute of the <project> element of the new project

B.25. PhingCallTask

The PhingCallTask calls a target within the same Phing project.

A <phingcall> tag may contain <property> tags that define new properties. These properties are only set if properties of the same name have not been set outside the "phingcall" tag.

Table B.26: *Attributes*

Name	Type/ Values	Description	Default	Required
`target`	`String`	The name of the target in the same project thatn/a is to be called.		Yes

Note

Local scope.

Every `<phingcall>` tag creates a new local scope. Thus, any properties or other variables set inside that scope will cease to exist (or revert to their previous value) once the `<phingcall>` tag completes.

B.25.1. Examples

```
<target name = "foo">
    <phingcall target = "bar">
        <property name = "property1" value = "aaaaa" />
        <property name = "foo" value = "baz" />
    </phingcall>
</target>
```

In the example above, the properties `property1` and `foo` are defined and only accessible inside the called target.

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

B.25.2. Supported Nested Tags

- `property`
- `param` (alias for property)

B.26. PhpEvalTask

With the `PhpEvalTask`, you can set a property to the results of evaluating a PHP expression or the result returned by a function/method call.

Table B.27: *Attributes*

Name	Type	Description	Default	Required
`function`	`String`	The name of the Property.	n/a	One of these is required.
`expression`	`String`	The expression to evaluate.	n/a	
`class`	`String`	The static class which contains function.	n/a	No
`returnProperty`	`String`	The name of the property to set with resultn/a of expression or function call. *Note:* if this at-		No

Name	Type	Description	Default	Required
		tribute is set, the expression must return a value.		
`level`	`String`	Control the level at which phplint reports status messages. One of `error`, `warning`, `info`, `verbose`, `debug`.	`info`	No

B.26.1. Examples

```
<php function = "crypt" returnProperty = "enc_passwd">
  <param value = "${auth.root_passwd}"/>
</php>
```

```
<php expression = "3 + 4" returnProperty = "sum"/>
```

```
<php expression = "echo 'test';">
```

B.26.2. Supported Nested Tags

- `param`

B.27. PropertyTask

With `PropertyTask`, you can define `user` properties in your build file.

Table B.28: *Attributes*

Name	Type	Description	Default	Required
`name`	`String`	The name of the Property.	n/a	Yes (unless using `file` or `environment`)
`value`	`String`	The value of the Property.	n/a	Yes (unless using `file` or `environment`)
`environment`	`String`	Loads properties from the environment with the specified value as prefix. Thus if you specify environment="myenv" you will be able to access OS-specific environment variables via property names "myenv.PATH" or "myenv.TERM".	n/a	No
`file`	`String`	Path to properties file.	n/a	No
`override`	`Boolean`	Whether to force override of existing value.	`false`	No
`prefix`	`String`	Used when properites are loaded from file. Prefix is applied to properties loaded from specified file. A "." is appended to the prefix if not specified.	n/a	No

Name	Type	Description	Default	Required
`refid`	`String`	A reference to a previously defined propeprty	n/a	No
`fallback`	`String`	If a reference cannot be found within the current project scope this attribute specifies a fallback project scope.	n/a	No

Note

Important note about scope: when the `<property>` tag is called inside a `<phingcall>` tag, any properties are set in a new local scope. Thus, any properties or other variables set inside that scope will cease to exist (or revert to their previous value) once the parent `<phingcall>` tag completes.

B.27.1. Examples

```
<property name = "strings.test" value = "Harr harr, more power!" />
<echo message = "${strings.test}" />

<property name = "foo.bar" value = "Yet another property..." />
<echo message = "${foo.bar}" />

<property file = "build.properties" />

<property environment = "env" />

<property name = "newproperty" value = "Hello">
  <filterchain>
    <replaceregexp>
      <regexp pattern = "Hello" replace = "World" ignoreCase = "true"/>
    </replaceregexp>
  <filterchain>
</property>
```

B.27.2. Supported Nested Tags:

- `filterchain`

B.28. PropertyPromptTask

`PropertyPromptTask` is a simple task to read in user input into a property. If you need something more advanced, see the Section B.20, “InputTask ”.

Table B.29: *Attributes*

Name	Type	Description	Default	Required
`propertyName`	`String`	The name of the Property to set.	n/a	Yes
`promptText`	`String`	The text to use for the prompt.	n/a	Yes
`promptCharacter`	`String`	The character to use after the prompt.	?	No
`defaultValue`	`String`	A default value to use (if user just hits enter).	n/a	No

Name	Type	Description	Default	Required
`useExisting-Value`	`String`	Whether existing property should be used if available. (This will result in user only being prompted if the propertyName property is not already set.)	`false`	No

B.28.1. Examples

```
<propertyprompt propertyName = "someprop" defaultValue = "/var/www"
  promptText = "Enter your web root" />
<echo>${someprop}</echo>
```

B.29. ReflexiveTask

The `ReflexiveTask` performs operations on files. It is essentially a convenient way to transform (using filter chains) files without copying them.

Table B.30: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	A single file to be processed.	n/a	Yes (unless <fileset> provided)

B.29.1. Examples

```
<reflexive>
  <fileset dir = ".">
    <include pattern = "*.html">
  </fileset>
  <filterchain>
    <replaceregexp>
      <regexp pattern = "\r(\n)" replace = "\1"/>
    </replaceregexp>
  </filterchain>
</reflexive>
```

B.29.2. Supported Nested Tags:

- `fileset`
- `filterchain`

B.30. ResolvePathTask

The `ResolvePathTask` turns a relative path into an absolute path, with respect to specified directory or the project basedir (if no dir attribute specified).

This task is useful for turning a user-defined relative path into an absolute path in cases where buildfiles will be called in different directories. Without this task, buildfiles lower in the directory tree would misinterpret the user-defined relative paths.

Table B.31: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The file or directory path to resolve.	n/a	Yes
`dir`	`File`	The base directory to use when resolving "file".	project.basedir	No
`propertyName`	`String`	The name of the property to set with resolved (absolute) path.	n/a	Yes
`level`	`String`	Control the level at which status messages are reported. One of `error`, `warning`, `info`, `verbose`, `debug`.	`verbose`	No

B.30.1. Examples

```
<property name = "relative_path" value = "./dirname"/>

<resolvepath propertyName = "absolute_path" file = "${relative_path}"/>

<echo>Resolved [absolute] path: ${absolute_path}</echo>
```

B.31. TaskdefTask

With the `TaskdefTask` you can import a user task into your buildfile.

Table B.32: *Attributes*

Name	Type	Description	Default	Required
`classname`	`String`	The path to the class that defines the TaskClass.	n/a	Yes, unless the `file` attribute has been specified.
`name`	`String`	The name the task is available as after importing. If you specify "validate", for example, you can access the task imported here with `<validate>`.	n/a	Yes, unless the `file` attribute has been specified.
`file`	`String`	Name of the file to load definitions from.	n/a	No
`classpath`	`String`	The classpath to use when including classes. This is added to PHP's include_path.	n/a	No
`classpathref`	`String`	Reference to classpath to use when including classes. This is added to PHP's include_path.	n/a	No

B.31.1. Examples

```
<!-- Includes the Task named "ValidateHTMLTask" and makes it available by
     <validatehtml> -->
<taskdef classname = "user.tasks.ValidateHTMLTask" name = "validatehtml" />
```

```
<!-- Includes the Task "RebootTask" from "user/sometasks" somewhere inside
     the $PHP_CLASSPATH -->
<taskdef classname = "user.sometasks.RebootTask" name = "reboot" />

<!-- Includes all tasks from the property file. Each line in the property
file defines a task in the format: name=path.to.Task -->
<taskdef file = "/path/to/mytasks.properties" />
```

NB: Taskdef now supports the PEAR-style naming convention to define and load tasks:

```
<taskdef name = "sampletask" classname = "Dir_Subdir_SampleTask"/>
```

will load class `Dir_Subdir_SampleTask` from file `Dir/Subdir/SampleTask.php`.

B.31.2. Supported Nested Tags

- `classpath`

B.32. TouchTask

The `TouchTask` works like the Unix `touch` command: It sets the `modtime` of a file to a specific time. Default is the current time.

Table B.33: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The file which time is to be changed.	n/a	Yes, or nested `<fileset>` tag
`datetime`	`DateTime`	The date and time the mtime of the file is to be set to. The format is "MM/DD/YYYY HH:MM AM or PM"	now	No
`millis`	`Integer`	The number of milliseconds since Midnight Jan 1 1970 (Unix epoche).	now	No

B.32.1. Examples

```
<touch file = "README.txt" millis = "102134111" />

<touch file = "COPYING.lib" datetime = "10/10/1999 09:31 AM" />
```

B.32.2. Supported Nested Tags

- `fileset`

B.33. TryCatchTask

This task is a wrapper task that lets you run tasks(s) when another set of tasks fails, mirroring PHP's `try/catch` functionality (with the addition of `finally` block)

The tasks inside of the `try` block will always be run. If one of them throws a BuildException, the following things can happen:

- If there is no `catch` block, the exception will be passed to Phing.
- If the `property` attribute has been set a property of that name will contain the message of the exception.
- If there is a `catch` block, the nested tasks will be run.

If a `finally` block is present, the nested tasks will be run regardless of whether the tasks in the `try` block have thrown an exception or not.

This task was inspired by http://ant-contrib.sourceforge.net/tasks/tasks/trycatch.html.

Table B.34: *Attributes*

Name	Type	Description	Default	Required
`property`	`String`	Name of a property that will receive the message of the exception that has been caught (if any)	n/a	No

B.33.1. Examples

```
<trycatch property="foo">
    <try>
        <fail>Tada!</fail>
    </try>

    <catch>
        <echo>In catch.</echo>
    </catch>

    <finally>
        <echo>In finally.</echo>
    </finally>
</trycatch>

<echo>As property: ${foo}</echo>
```

B.34. TstampTask

Sets the `DSTAMP`, `TSTAMP`, and `TODAY` properties in the current project. By default, the `DSTAMP` property is in the format `"%Y%m%d"`, `TSTAMP` is in the format `"%H%M"`, and `TODAY` is in the format `"%B %d %Y"`. Use the nested `<format>` element to specify a different format.

These properties can be used in the build-file, for instance, to create time-stamped filenames, or used to replace placeholder tags inside documents to indicate, for example, the release date. The best place for this task is probably in an initialization target.

Table B.35: *Attributes*

Name	Type	Description	Default	Required
`prefix`	`String`	Prefix used for all properties set.	n/a	No

B.34.1. Examples

```
<tstamp/>
```

sets the standard DSTAMP, TSTAMP, and TODAY properties according to the default formats.

```
<tstamp>
  <format property = "DATE" pattern = "%c" locale = "nl_NL"/>
</tstamp>
```

sets the standard properties as well as the property DATE with the date/time pattern "%c" using the Dutch locale.

```
<tstamp prefix = "start"/>
```

sets three properties with the standard formats, prefixed with "start.": start.DSTAMP, start.TSTAMP, and start.TODAY.

B.34.2. Supported Nested Tags

- `format`

  The `Tstamp` task supports a `<format>` 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 PHP `strftime()` function.

Table B.36: *Attributes*

Name	Type	Description	Default	Required
`property`	`String`	The property to receive the date/time string in the given pattern.	n/a	Yes
`classname`	`String`	The date/time pattern to be used. The values are as defined by the PHP strftime() function.	n/a	Yes
`locale`	`String`	The locale used to create date/time string. For more information see the PHP setlocale() function.	n/a	No

B.35. TypedefTask

With the `TypedefTask` you can import a user type into your buildfile.

Table B.37: *Attributes*

Name	Type	Description	Default	Required
`classname`	`String`	The path to the class that defines the type class.	n/a	Yes
`name`	`String`	The name the type is available as after importing. If you specify "cproject", for example, you can access the type imported here with `<cproject>`.	n/a	Yes
`classpath`	`String`	The classpath to use when including classes. This is added to PHP's include_path.	n/a	No

Name	Type	Description	Default	Required
`classpathref`	`String`	Reference to classpath to use when including classes. This is added to PHP's include_path.	n/a	No

B.35.1. Examples

```
<!--
Includes the Type named "CustomProject" and makes it available by
<cproject>
-->
<typedef classname = "user.types.CustomProject" name = "cproject" />
```

B.35.2. Supported Nested Tags

- `classpath`

B.36. UpToDateTask

`UpToDateTask` tests if a file is newer than another file or files and sets a property if it is. This is a common way to avoid, possibly time consuming, creation of a target if none of the files/resources it depends on have changed.

Table B.38: *Attributes*

Name	Type	Description	Default	Required
`property`	`String`	Name of the property that is to be set	n/a	Yes
`value`	`String`	The value the property is to be set to	`true`	No
`srcfile`	`String`	The file to check against target file(s)	n/a	Yes (or nested `fileset`)
`targetfile`	`String`	The file for which we want to determine the status	n/a	Yes (or nested `mapper`)

B.36.1. Examples

```
<uptodate property = "propelBuild.notRequired" targetfile = "${deploy}/propelClasses.tgz" >
  <fileset dir = "${src}/propel">
    <include="**/*.php"/>
  </fileset>
</uptodate>
```

The above example sets the property `propelBuild.notRequired` to true if the `${deploy}/propelClasses.tgz` file is more up-to-date than any of the PHP class files in the `${src}/propel` directory.

```
<target name = "CompileTarget">
  <uptodate property = "target.uptodate" targetfile = "main">
    <fileset refid = "sources"/>
  </uptodate>
```

```
  <if>
    <not><isset property = "target.uptodate"/></not>
   <then>
      <!-- Some commands to update the target ... -->
   </then>
  </if>
</target>
```

The above example shows a common use when doing a "compile" type target where a single target depends on other source files. In this case the commands to update the target (whatever they are) are only run if any of the source files are more up to date than the target.

B.36.2. Supported Nested Tags

- `filelist`
- `fileset`
- `mapper`

B.37. WaitForTask

Wait for a condition to become true or a timeout, whichever comes first.

Table B.39: *Attributes*

Name	Type	Description	Default	Required
`MaxWait`	`Integer`	Set the maximum length of time to wait in units	3min	Yes
`WaitUnit`	`String`	Set the max wait time unit. Must be one of "`week`","`day`", "`hour`", "`minute`", "`second`", "`millisecond`"	millisec-ond	No
`CheckEvery`	`Integer`	Set the time between each check	500ms	Yes
`CheckEveryUnit`	`String`	Set the check every time unit. Must be one of "`week`","`day`", "`hour`", "`minute`", "`second`", "`millisecond`"	millisec-onds	No
`TimeoutProper-ty`	`String`	Name of the property to set after a timeout.	null	No

B.37.1. Examples

Wait for a maximum of ten seconds for the file "`ready`" to appear.

```
<waitfor maxwaitunit = "second" maxwait = "10">
  <available file = "ready"/>
</waitfor>
```

B.37.2. Supported Nested Tags

All conditionals including `and`, `or`, `not` etc.

B.38. XsltTask

With `XsltTask`, you can run a XSLT transformation on an XML file. Actually, `XsltTask` extends `CopyTask`, so you can use all the elements allowed there.

`XsltTask` is implemented by means of the `XsltFlter` and hence relies on PHP5 XSLT support via (`libxslt`) which must be available in php5. The `XsltTask` is equivalent to running command line `xsltproc` since that is a frontend for libxslt.

Table B.40: *Attributes*

Name	Type	Description	Default	Required
`style`	`String`	The path where the Xslt file is located	n/a	Yes
`resolvedocumentexternals`	`Boolean`	Whether to resolve entities in the XML document. (see this link [http://www.php.net/manual/en/class.domdocument.php#domdocument.props.resolveexternals] for details)	`false`	No
`resolvestyleshеetexternals`	`Boolean`	Whether to resolve entities in the stylesheet.	`false`	No

`Note:` You can also use all the attributes available for Section B.9, “CopyTask ”.

B.38.1. Examples

```
<!-- Transform docbook with an imaginary XSLT file -->
<xslt todir = "/srv/docs/phing" style = "dbk2html.xslt" >
  <fileset dir = ".">
    <include name = "**/*.xml" />
  </fileset>
</xslt>
```

B.38.2. Supported Nested Elements

- `mapper`
- `filterchain`
- `param`

`Note:` You can use all the elements also available for Section B.9, “CopyTask ”.

Additionally, you can use <param> tags with a `name` and a `expression` (or `value` alias) attribute. These parameters are then available from within the xsl style sheet.

Appendix C. Optional tasks

This appendix contains a reference of all optional tasks, i.e. tasks that are not directly needed for building projects, but can assist in various aspects of development and deployment.

This reference lists the tasks alphabetically by the name of the classes that implement the tasks. So if you are searching for the reference to the <phplint> tag, for example, you will want to look at the reference of PhpLintTask.

C.1. ApiGenTask

This task runs ApiGen [http://apigen.org/], a tool for creating professional API documentation from PHP source code, similar to discontinued phpDocumentor/phpDoc.

Table C.1: *Attributes*

Name	Type	Description	Default	Required
executable	String	ApiGen executable name.	apigen	No
config	String	Config file name.	n/a	Source and destination are required - either set explicitly or using a config file. Attribute values set explicitly have precedence over values from a config file.
source	String	List of source files or directories.	n/a	
destination	String	Destination directory.	n/a	
exclude	String	List of masks (case sensitive) to exclude files or directories from processing.	n/a	No
skipdocpath	String	List of masks (case sensitive) to exclude elements from documentation generating.	n/a	No
skipdocprefix	String	Name prefix (case sensitive) to exclude elements from documentation generating.	n/a	No
charset	String	Character set of source files.	UTF-8	No
main	String	Main project name prefix.	n/a	No
title	String	Title of generated documentation.	n/a	No
baseurl	String	Documentation base URL.	n/a	No
googlecseid	String	Google Custom Search ID.	n/a	No
googlecselabel	String	Google Custom Search label.	n/a	No
googleanalytics	String	Google Analytics tracking code.	n/a	No
templateconfig	String	Template config file name.	n/a	If not set the default template is used.

Name	Type	Description	Default	Required
`allowedhtml`	`String`	List of HTML tags allowed in the documentation.	b, i, a, ul, ol, li, p, br, var, samp, kbd, tt	No
`accesslevels`	`String`	Element access levels. Documentation only for methods and properties with the given access level will be generated.	public, protected	No
`internal`	`Boolean`	Whether to generate documentation for elements marked as internal and internal documentation parts or not.	No	No
`php`	`Boolean`	Whether to generate documentation for PHP internal classes or not.	Yes	No
`tree`	`Boolean`	Whether to generate tree view of classes, interfaces, traits and exceptions or not.	Yes	No
`deprecated`	`Boolean`	Whether to generate documentation for deprecated elements or not.	No	No
`todo`	`Boolean`	Whether to generate documentation of tasks or not.	No	No
`sourcecode`	`Boolean`	Whether to generate highlighted source code files or not.	Yes	No
`download`	`Boolean`	Whether to generate a link to download documentation as a ZIP archive or not.	No	No
`report`	`String`	File name for checkstyle report of poorly documented elements.	n/a	No
`wipeout`	`Boolean`	Whether to wipe out the destination directory first or not.	Yes	No
`quiet`	`Boolean`	Whether to enable scaning and generating messages or not.	No	No
`updatecheck`	`Boolean`	Whether to check for ApiGen updates or not.	No	No
`debug`	`Boolean`	Whether to enable the debug mode or not.	No	No

C.1.1. Example

```
<apigen
  source = "classes"
  destination = "api"
  exclude = "*/tests/*"
  title = "My Project API Documentation"
  deprecated = "true"
  todo = "true"/>
```

C.2. CoverageMergerTask

The CoverageMergerTask merges code coverage information from external sources with an existing code coverage database.

The format of the code coverage files is expected to be identical to:

```
file_put_contents('/www/live/testcases/coverage.data', serialize(xdebug_get_code_coverage));
```

C.2.1. Example

```
<coverage-merger>
  <fileset dir = "/www/live/testcases">
    <include name = "**/*.data"/>
  </fileset>
</coverage-merger>
```

C.2.2. Supported Nested Tags

- `fileset`

C.3. CoverageReportTask

The CoverageReportTask formats a coverage database into a framed HTML report using XSLT. The report can optionally make use of the **Ge**neric **S**yntax **Hi**ghlighting library, `GeSHi` (See GeSHi Homepage [http://qbnz.com/highlighter/]) library to mark up source code. The path to the library (if not in the default path) can be specified as an attribute.

Table C.2: *Attributes*

Name	Type	Description	Default	Required
`outfile`	`String`	The location for the intermediate XML file.	coverage.dat	Yes
`classpath`	`String`	Additional classpath to locate source referenced in the report	n/a	No
`geshipath`	`String`	Path to GeSHi highlighting library	n/a	No/Yes* If syntax highlighting si to be enabled
`geshilanguagespath`	`String`	Language to use with GeSHi	n/a	No

C.3.1. Example

```
<coverage-report outfile = "reports/coverage.xml">
  <report todir = "reports/coverage" styledir = "/home/phing/etc"/>
</coverage-report>
```

C.3.2. Supported Nested Tags

- report

Table C.3: *Attributes*

Name	Type	Description	Default	Required
`styledir`	`String`	The directory where the stylesheets are located.	n/a	Yes

Name	Type	Description	Default	Required
`todir`	`String`	The directory where the files resulting from the transformation should be written to.		Yes
`title`	`String`	Title of the project (used in the generated document(s))		No
`usesorttable`	`Boolean`	Whether to use the sorttable JavaScript library (see http://www.kryogenix.org/code/browser/sorttable/)	`false`	No

C.4. CoverageSetupTask

The CoverageSetupTask prepares a database which can be used to gather code coverage information for unit tests.

Table C.4: *Attributes*

Name	Type	Description	Default	Required
`database`	`String`	The location for the coverage database.	coverage.db	Yes

C.4.1. Example

```
<coverage-setup database = "./reports/coverage.db">
  <fileset dir = "classes">
    <include name = "**/*.php"/>
  </fileset>
</coverage-setup>
<phpunit codecoverage = "true">
  <batchtest>
    <fileset dir = "src">
      <include name = "*Test.php"/>
    </fileset>
  </batchtest>
</phpunit>
```

C.4.2. Supported Nested Tags

- `classpath`
- `fileset`
- `filelist`

C.5. CoverageThresholdTask

This task validates the code coverage database and will stop the build cycle if any class or method or entire project's coverage is lower than the specified threshold.

Table C.5: *Attributes*

Name	Type	Description	Default	Required
`database`	`String`	The location of the coverage database. (This is optional if `CoverageSetupTask` has run before)	n/a	No
`perProject`	`Integer`	The minimum code coverage for the entire project.	25	No
`perClass`	`Integer`	The minimum code coverage for any class.	25	No
`perMethod`	`Integer`	The minimum code coverage for any method.	25	No
`verbose`	`Boolean`	Whether to enable detailed logging or not.	`false`	No

C.5.1. Example

```
<coverage-threshold database = "./reports/coverage.db"/>
```

C.5.2. Supported Nested Tags

- `classpath`
- excludes

Validates an optional code coverage database against the default thresholds.

```
<coverage-threshold
    perProject = "50"
    perClass = "60"
    perMethod = "70"/>
```

Validates the code coverage database (from CoverageSetupTask) against the specified thresholds.

```
<coverage-threshold
    perProject = "50"
    perClass = "60"
    perMethod = "70"/>
    <excludes>
        <file>**/*Processor.php</file>
        <class>Model_Filter_Windows</class>
        <method>Model_System::execute()</method>
    </excludes>
```

Validates the code coverage database (from CoverageSetupTask) against the specified thresholds and excludes the given file, class and method from threshold validation. The filename is relative to the project basedir. A Method can be named either "Model_System::execute()" or "Model_System::execute". The method name is considered only for the given class "Model_System".

C.6. DbDeployTask

The `DbDeployTask` creates .sql files for making revisions to a database, based on dbdeploy conventions centering around a changelog table in the database. See rules for using dbdeploy [http://dbdeploy.com/documentation/getting-started/rules-for-using-dbdeploy/] for more information. You will need a changelog table like so:

Table C.6: *Attributes*

Name	Type	Description	Default	Required
`url`	`String`	PDO connection url	n/a	Yes
`userid`	`String`	DB userid to use for accessing the changelog table	none	As required by db
`password`	`String`	DB password to use for accessing the changelog table	none	As required by db
`dir`	`String`	Directory containing dbdeploy delta scripts	none	Yes
`outputfile`	`String`	Filename in which deployment SQL will be generated	dbdeploy_deploy.sql	No
`undooutputfile`	`String`	Filename in which undo SQL will be generated	dbdeploy_undo.sql	No
`deltaset`	`String`	deltaset to check within db	Main	No
`lastchange-toapply`	`Integer`	Highest-numbered delta script to apply to db	999	No
`appliedBy`	`String`	Value of the 'applied_by' column for each entry in the changelog table	dbdeploy	No
`checkall`	`Boolean`	False means dbdeploy will only apply patches that have a higher number than the last patch-number that was applied True means dbde-ploy will apply all changes that aren't applied already (in ascending order)	false	No

C.6.1. Example

```
CREATE TABLE changelog (
  change_number BIGINT NOT NULL,
  delta_set VARCHAR(10) NOT NULL,
  start_dt TIMESTAMP NOT NULL,
  complete_dt TIMESTAMP NULL,
  applied_by VARCHAR(100) NOT NULL,
  description VARCHAR(500) NOT NULL
)
```

```
<dbdeploy
  url = "sqlite:${project.basedir}/data/db.sqlite"
  userid = "dbdeploy"
  password = "dbdeploy"
  dir = "${project.basedir}/data/dbdeploy/deltas"
/>
```

The above example uses a sqlite database and delta scripts located in dbdeploy/deltas in the project base dir.

C.7. DocBloxTask

This task is now deprecated in favor of the Section C.52, “DocBloxTask”.

This task runs DocBlox [http://www.docblox-project.org/], a PHP 5.3-compatible API documentation tool.

Table C.7: *Attributes*

Name	Type	Description	Default	Required
`title`	`String`	Title of the project.	n/a	No
`destdir`	`String`	Destination directory for output files.	n/a	Yes
`quiet`	`Boolean`	Suppress DocBlox chatter.	`true`	No

C.7.1. Example

```
<docblox title = "API Documentation"
  destdir = "apidocs">
   <fileset dir = "./classes">
      <include name = "**/*.php" />
   </fileset>
</docblox>
```

C.7.2. Supported Nested Tags

- fileset - Files that should be included for parsing

C.8. ExportPropertiesTask

Exports all defined properties to a specified file.

Table C.8: *Attributes*

Name	Type	Description	Default	Required
`targetfile`	`String`	Target file for saved properties	n/a	Yes
`disallowed-PropertyPre-fixes`	`String`	Exclude properties starting with these prefixes (separated by `,`	`'host.','ph-ing.','os.','php.','line.','env.','user.'`	No

C.8.1. Example

```
<exportproperties targetfile = "output.props" />
```

C.9. FileHashTask

Calculates either MD5 or SHA1 hash value of a file and stores the value as a hex string in a property.

Table C.9: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Filename	n/a	Yes

Name	Type	Description	Default	Required
`hashtype`	`Integer`	Specifies what hash algorithm to use. 0=MD5,1=SHA1	0	No
`propertyname`	`String`	Name of property where the hash value is stored	filehash-value	No

C.9.1. Example

```
<filehash file = "${builddir}/${tarball}.tar.${compression}" />
<echo msg = "Hashvalue is; ${filehashvalue}" />
```

C.10. FileSizeTask

Stores the size of a specified file in a property. The file size is returned in bytes.

Table C.10: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Filename	n/a	Yes
`propertyname`	`String`	Name of property where the file size is stored	filesize	No

C.10.1. Example

```
<filesize file = "${builddir}/${tarball}.tar.${compression}" />
<php expression = "floor(${filesize}/1024)" returnProperty = "ksize" />
<php expression = "floor(${filesize}/1024/1024)" returnProperty = "msize" />
<echo msg = "Filesize is: ${ksize} kB"/>
<echo msg = "Filesize is: ${msize} MB"/>
```

C.11. FileSyncTask

Syncs files or directories using the rsync command. Syncing can be done on the same server or from/to a remote server.

Table C.11: *Attributes*

Name	Type	Description	Default	Required
`rsyncPath`	`String`	Path to rsync command	/usr/bin/rsync	Yes
`sourceDir`	`String`	Source directory (with user&host for remote sources)	n/a	Yes
`destinationDir`	`String`	Destination directory (with user&host for remote destinations)	n/a	Yes
`excludeFile`	`String`	Excluded patterns file	n/a	No
`backupDir`	`String`	Creates a backup so users can rollback to an existing restore point	n/a	No

Name	Type	Description	Default	Required
`options`	`String`	Any options that rsync supports, removes the default options	-rpKz	No
`verbose`	`Boolean`	This option increases the amount of information you are given during the transfer	True	No
`dryRun`	`Boolean`	This option makes rsync perform a trial run that doesn't make any changes	False	No
`itemizeChanges`	`Boolean`	This option requests a simple itemized list of the changes that are being made to each file, including attribute changes	False	No
`checksum`	`Boolean`	This option will cause rsync to skip files based on checksum, not mod-time & size	False	No
`delete`	`Boolean`	This option deletes files that don't exist on sender after transfer including `force` and `ignore-errors`	False	No
`identityFile`	`String`	Identity file for ssh authentication of a remote transfer	n/a	No

C.11.1. Example

```
<filesync sourcedir = "/var/www/development/project1" destinationdir = "/var/www/project1" />;

<filesync
sourcedir = "/var/www/development/project1"
destinationdir = "user@server:/var/www/project1"
dryrun = "true"
itemizechanges = "true"
verbose = "true"
checksum = "true" />;
```

C.12. FtpDeployTask

Deploys a set of files to a remote FTP server.

Table C.12: *Attributes*

Name	Type	Description	Default	Required
`host`	`String`	The hostname of the remote server.	none	Yes
`port`	`Integer`	The port of the remote server.	21	No
`username`	`String`	The username to use when logging in to the remote server.	none	Yes
`password`	`String`	The password to use when logging in to the remote server	none	Yes
`dir`	`String`	Directory on the remote server.	none	No
`mode`	`String`	The transfer mode to use, either `ascii` or `binary`.	bi-nary	No
`clearfirst`	`Boolean`	Delete all files in the remote directory before uploading	`false`	No

Name	Type	Description	Default	Required
passive	Boolean	Open connection in passive mode	false	No
dirmode	mixed	Permissions of the uploaded files, can either be 'inherit' or it can be a octal value without the leading zero. Settings the dirmode to 'inherit' will cause the uploaded files to have the same permissions as on the filesystem.	false	No
filemode	mixed	This option does the same as dirmode, except it only affects regular files.	false	No
depends	boolean	If depends is set to true, the task will only update files with a local modification timestamp that is newer than the corresponding timestamp on the server.	false	No
level	String	Control the level at which the task reports status messages. One of error, warning, info, verbose, debug.	verbose	No

C.12.1. Example

```
<ftpdeploy
  host = "${ftp.host}"
  port = "${ftp.port}"
  username = "${ftp.username}"
  password = "${ftp.password}"
  dir = "${ftp.dir}"
  passive = "false"
  mode = "${ftp.mode}">
  <fileset dir = ".">
    <include name = "**"/>
    <exclude name = "phing"/>
    <exclude name = "build.xml"/>
    <exclude name = "images/**.png"/>
    <exclude name = "images/**.gif"/>
    <exclude name = "images/**.jpg"/>
  </fileset>
</ftpdeploy>
```

C.12.2. Supported Nested Tags

- fileset

  The files to deploy

C.13. GitInitTask

Create an empty git repository or reinitialize an existing one.

Table C.13: *Attributes*

Name	Type	Description	Default	Required
gitPath	String	Path to Git binary	/usr/bin/git	No

Name	Type	Description	Default	Required
`repository`	`String`	Path to Git repository	n/a	Yes
`bare`	`Boolean`	Create bare repository. See --bare option of git-init [http://www.kernel.org/pub/software/scm/git/docs/git-init.html].	`false`	No

C.13.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- Initialize normal repository -->
<gitinit repository = "${repo.dir.resolved}" />

<!-- Initialize bare repository -->
<gitinit bare = "true" repository = "${repo.dir.resolved}" />
```

C.14. GitCloneTask

Clone a repository into a new directory.

Table C.14: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	The (possibly remote) repository to clone from.	n/a	Yes
`targetPath`	`String`	The name of a new directory to clone into. Cloning into an existing directory is only allowed if the directory is empty.	n/a	Yes
`bare`	`Boolean`	Create bare repository. See --bare option of git-clone [http://www.kernel.org/pub/software/scm/git/docs/git-clone.html].	`false`	No

C.14.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- Clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<!-- Clone bare repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}"
    bare = "true" />
```

C.15. GitGcTask

Cleanup unnecessary files and optimize the local repository.

Table C.15: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	The repository to cleanup.	n/a	Yes
`aggressive`	`Boolean`	This option will cause git gc to more aggressively optimize the repository at the expense of taking much more time. See --aggressive option of git-gc [http://www.kernel.org/pub/software/scm/git/docs/git-gc.html].	`false`	No
`auto`	`Boolean`	With this option, git gc checks whether any housekeeping is required; if not, it exits without performing any work. See --auto option of git-gc [http://www.kernel.org/pub/software/scm/git/docs/git-gc.html].	`false`	No
`noprune`	`Boolean`	Do not prune any loose objects. See --no-prune option of git-gc [http://www.kernel.org/pub/software/scm/git/docs/git-gc.html].	`false`	No
`prune`	`String`	Prune loose objects older than date. See --prune option of git-gc [http://www.kernel.org/pub/software/scm/git/docs/git-gc.html].	2.weeks.ago	No

C.15.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- Clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<!-- Cleanup repository-->
<gitgc
    repository = "${repo.dir.resolved}"
    aggressive = "true"
    prune = "1.week.ago" />
```

C.16. GitBranchTask

Create, move or delete repository branches. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-branch.html] (branch listing functionality is omitted in current implementation).

Table C.16: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No

Name	Type	Description	Default	Required
`repository`	`String`	Path to Git repository	n/a	Yes
`branchname`	`String`	The name of the branch to create or delete.	n/a	Yes
`newbranch`	`String`	The new name for an existing branch.	n/a	Yes, if branch move invoked
`startpoint`	`String`	The new branch head will point to this commit. It may be given as a branch name, a commit-id, or a tag. If this option is omitted, the current HEAD will be used instead. See <start-point> argument of git-branch [http://www.kernel.org/pub/software/scm/git/docs/git-branch.html].		No
`setupstream`	`String`	If specified branch does not exist yet or if --force has been given, acts exactly like --track. Otherwise sets up configuration like --track would when creating the branch, except that where branch points to is not changed. See --set-upstream option of git-branch [http://www.kernel.org/pub/software/scm/git/docs/git-branch.html].		No
`track`	`Boolean`	See --track option of git-branch [http://www.kernel.org/pub/software/scm/git/docs/git-branch.html].	`false`	No
`notrack`	`Boolean`	See --no-track option of git-branch [http://www.kernel.org/pub/software/scm/git/docs/git-branch.html].	`false`	No
`force`	`Boolean`	Reset <branchname> to <startpoint> if <branchname> exists already. Without -f git branch refuses to change an existing branch.	`false`	No
`move`	`Boolean`	Move/rename a branch and the corresponding reflog.	`false`	No
`forcemove`	`Boolean`	Move/rename a branch even if the new branch name already exists.	`false`	No
`delete`	`Boolean`	Delete a branch. The branch must be fully merged in its upstream branch, or in HEAD if no upstream was set with --track or --set-upstream.	`false`	No
`forcedelete`	`Boolean`	Delete a branch irrespective of its merged status.	`false`	No

C.16.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- Initialize normal repository -->
<gitinit repository = "${repo.dir.resolved}" />

<!-- Create branch "sample-branch" tracking current HEAD -->
<gitbranch
```

```
    repository = "${repo.dir.resolved}"
    branchname = "sample-branch" />

<!--
Create branch "sample-branch" tracking origin/master
Note that you can omit both startpoint and track attributes in this case
-->
<gitbranch
    repository = "${repo.dir.resolved}"
    branchname = "sample-branch"
    startpoint = "origin/master"
    track = "true" />

<!-- Delete fully merged branch "sample-branch" -->
<gitbranch
    repository = "${repo.dir.resolved}"
    branchname = "sample-branch"
    delete = "true" />

<!-- Force delete even unmerged branch "sample-branch" -->
<gitbranch
    repository = "${repo.dir.resolved}"
    branchname = "sample-branch"
    forcedelete = "true" />

<!-- Renabe "branch1" to "branch2" -->
<gitbranch
    repository = "${repo.dir.resolved}"
    branchname = "branch1"
    newbranch = "branch2"
    move = "true" />
```

C.17. GitFetchTask

Download objects and refs from another repository. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].

Table C.17: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	Path to Git repository	n/a	Yes
`source`	`String`	The "remote" repository that is the source of a fetch or pull operation. See <repository> in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	origin	No
`refspec`	`String`	See <refspec> in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].		No
`group`	`String`	A name referring to a list of repositories as the value of remotes.<group> in the configuration file. See <group> in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].		No
`quiet`	`Boolean`	Silence any internally used git commands. Progress is not reported to	`false`	No

Name	Type	Description	Default	Required
		the standard error stream. See --quiet in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].		
`all`	`Boolean`	Fetch all remotes. See --all in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	`false`	No
`keep`	`Boolean`	Keep downloaded pack. See --keep in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	`false`	No
`prune`	`Boolean`	After fetching, remove any remote tracking branches which no longer exist on the remote. See --prune in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	`false`	No
`tags`	`Boolean`	See --tags in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	`false`	No
`notags`	`Boolean`	See --no-tags in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	`false`	No
`force`	`Boolean`	When git fetch is used with <rbranch>:<lbranch> refspec, it refuses to update the local branch <lbranch> unless the remote branch <rbranch> it fetches is a descendant of <lbranch>. This option overrides that check. See --force in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	`false`	No

C.17.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- Initialize normal repository -->
<gitinit repository = "${repo.dir.resolved}" />

<!-- Fetch objects from all remotes -->
<gitfetch
    repository = "${repo.dir.resolved}" all = "true" />

<!-- Fetch from origin/master to "refspec-branch" local branch -->
<gitfetch
    repository = "${repo.dir.resolved}"
    source = "origin"
    refspec = "master:refspec-branch"
    quiet = "true" />
```

C.18. GitCheckoutTask

Checkout a branch or paths to the working tree. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].

Table C.18: *Attributes*

Name	Type	Description	Default	Required
gitPath	String	Path to Git binary	/usr/bin/git	No
repository	String	Path to Git repository	n/a	Yes
branchname	String	Branch to checkout. See <branch> in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].	origin	No
startpoint	String	The name of a commit at which to start the new branch; Defaults to HEAD. See <start_point> in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].		No
create	Boolean	Create a new branch named <branchname> and start it at <startpoint>	false	No
forcecreate	Boolean	Creates the branch <branchname> and start it at <startpoint>; if it already exists, then reset it to <startpoint>. This is equivalent to running "git branch" with "-f".	false	No
merge	Boolean	See --merge in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].	false	No
track	Boolean	See --track in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].	false	No
notrack	Boolean	See --no-track in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].	false	No
quiet	Boolean	Quiet, suppress feedback messages. See --quiet in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].	false	No
force	Boolean	When switching branches, proceed even if the index or the working tree differs from HEAD. This is used to throw away local changes. See --force in git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].	false	No

C.18.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<!-- create and switch to "mybranch" branch -->
<gitcheckout
    repository = "${repo.dir.resolved}"
    branchname = "mybranch" quiet = "true" create = "true" />

<!-- get back to "master" branch -->
```

```
<gitcheckout
    repository = "${repo.dir.resolved}"
    branchname = "master" quiet = "true" />

<!-- create (force) already created branch -->
<gitcheckout
    repository = "${repo.dir.resolved}"
    branchname = "mybranch" quiet = "true"
    forceCreate = "true" />
```

C.19. GitCommitTask

Record changes to the repository. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-commit.html].

Table C.19: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	Path to Git repository	n/a	Yes
`message`	`String`	Commit message	n/a	No
`allFiles`	`Boolean`	Whether to automatically stage files that have been modified and deleted (see --all in git-commit [http://www.kernel.org/pub/software/scm/git/docs/git-commit.html])	n/a	No

C.19.1. Example

```
<!-- commit all modified / deleted files -->;
<gitcommit
    repository = "/path/to/repo"
    message = "Commit message" allFiles = "true" />;
```

C.20. GitMergeTask

Join two or more development histories together. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].

Table C.20: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	Path to Git repository	n/a	Yes

Name	Type	Description	Default	Required
remote	String	Space separated list of branches to merge into current HEAD. See <commit> in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	n/a	No
message	String	Commit message to be used for the merge commit (in case one is created). See <msg> in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	n/a	No
fastForwardCommit	Boolean	If set false (default), will not generate a merge commit if the merge resolved as a fast-forward, only update the branch pointer. If set true, will generate a merge commit even if the merge resolved as a fast-forward. See --ff/--no-ff options in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	false	No
strategy	String	Merge strategy. One of "resolve", "recursive", "octopus", "ours", or "subtree". See <strategy> in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	n/a	No
strategyOption	String	Pass merge strategy specific option through to the merge strategy. See <strategy-option> in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	n/a	No
commit	Boolean	See --commit in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	false	No
nocommit	Boolean	See --no-commit in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	false	No
quiet	Boolean	Quiet, suppress feedback messages. See --quiet in git-merge [http://www.kernel.org/pub/software/scm/git/docs/git-merge.html].	false	No

C.20.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<!-- create couple of test branches -->
<gitbranch
    repository = "${repo.dir.resolved}"
    branchname = "merge-test-1" startpoint = "origin/master" />
<gitbranch
    repository = "${repo.dir.resolved}"
    branchname = "merge-test-2" startpoint = "origin/master" />

<!-- Merge those branches back into master -->
```

```
<gitmerge
    repository = "${repo.dir.resolved}"
    remote = "merge-test-1 merge-test-2"
    message = "merging repos" commit = "true" />
```

C.21. GitPullTask

Fetch from and merge with another repository or a local branch. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].

Table C.21: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	Path to Git repository	n/a	Yes
`all`	`Boolean`	Fetch all remotes	`false`	No
`source`	`String`	The "remote" repository that is the source of a fetch or pull operation. See <repository> in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	origin	Yes, if allRemotes set to false
`refspec`	`String`	See <refspec> in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	n/a	No
`strategy`	`String`	Merge strategy. One of "resolve", "recursive", "octopus", "ours", or "subtree". See <strategy> in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	n/a	No
`strategyOption`	`String`	Pass merge strategy specific option through to the merge strategy. See <strategy-option> in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	n/a	No
`rebase`	`Boolean`	See --rebase in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	`false`	No
`norebase`	`Boolean`	See --no-rebase in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	`false`	No
`tags`	`Boolean`	Enable tag references following. See --tags in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	`false`	No
`notags`	`Boolean`	Disable tag references following. See --no-tags in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	`false`	No
`keepFiles`	`Boolean`	See --keep in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	`false`	No
`append`	`Boolean`	See --append in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	`false`	No

Name	Type	Description	Default	Required
quiet	Boolean	Quiet, suppress feedback messages. See --quiet in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	false	No
force	Boolean	Force update. See --force in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].	false	No

C.21.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<!-- pull from all remotes -->
<gitpull
    repository = "${repo.dir.resolved}" all = "true" />

<!-- pull remote origin/foobranch and rebase when merging -->
<gitpull
    repository = "${repo.dir.resolved}"
    source = "origin" refspec = "foobranch"
    strategy = "recursive" keep = "true"
    force = "true" quiet = "true" rebase = "true" />
```

C.22. GitPushTask

Update remote refs along with associated objects. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].

Table C.22: *Attributes*

Name	Type	Description	Default	Required
gitPath	String	Path to Git binary	/usr/bin/git	No
repository	String	Path to Git repository	n/a	Yes
all	Boolean	Push all references	false	No
destination	String	The "remote" repository that is destination of a push operation. See <repository> in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	origin	Yes, if allRemotes set to false
refspec	String	See <refspec> in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	n/a	No
mirror	Boolean	See --mirror in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	false	No

Name	Type	Description	Default	Required
`delete`	`Boolean`	Delete "remote" reference. Same as pre-fixing the refspec with colon. See --delete in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	`false`	No
`tags`	`Boolean`	Push all references under refs/tags. See --tags in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	`false`	No
`quiet`	`Boolean`	Quiet, suppress feedback messages. See --quiet in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	`false`	No
`force`	`Boolean`	Force update. See --force in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	`false`	No

C.22.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<!-- push branch "master" into "foobranch" on "origin" remote -->
<gitpush
    repository = "${repo.dir.resolved}"
    refspec = "master:foobranch" tags = "true" />

<!-- create new branch "newbranch" on "origin" remote -->
<gitpush
    repository = "${repo.dir.resolved}"
    refspec = "master:newbranch" quiet = "true" />

<!-- delete "newbranch" branch from "origin" remote -->
<gitpush
    repository = "${repo.dir.resolved}"
    delete = "true"
    refspec = "newbranch" quiet = "true" />
```

C.23. GitTagTask

Create, list, delete or verify a tag object signed with GPG. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html].

Table C.23: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	Path to Git repository	n/a	Yes

Name	Type	Description	Default	Required
message	String	Use given tag message. See -m of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	n/a	No
name	String	Tag name	n/a	Yes
commit	String	<commit> argument to git-tag	n/a	No
object	String	<object> argument to git-tag	n/a	No
pattern	String	<pattern> argument to git-tag	n/a	No
outputProperty	String	Property name to set with output value from git-tag	n/a	No
file	String	Take tag message from given file. See -F of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	n/a	No
annotate	Boolean	Make unsigned, annotated tag object. See -a of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No
force	Boolean	Replace existing tag with given name. See -f of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No
delete	Boolean	Delete existing tags with given names. See -d of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No
list	Boolean	List tags with names matching given pattern. See -l of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No
num	Integer	Specifies how many lines from the annotation, if any, are printed when using -l. See -n of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	n/a	No
contains	String	Only list tags containing specified commit. See --contains of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	n/a	No
sign	Boolean	Make GPG-signed tag. See -s of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No
keySign	String	Make GPG-signed tag, using given key. See -u of git-tag of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	n/a	No
verify	Boolean	Verify GPG signature of given tag names. See -v of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No

C.23.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />
```

```
<!-- clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<gittag repository = "${repo.dir.resolved}" name = "ver1.0" />
<!-- Force duplicate tag creation -->
<gittag
    repository = "${repo.dir.resolved}"
    name = "ver1.0" force = "true"/>
<!-- Create tag with annotation and message -->
<gittag
    repository = "${repo.dir.resolved}"
    name = "ver1.0"
    annotate = "true" message = "Version 1.0 tag"/>
<!-- Delete tag -->
<gittag
    repository = "${repo.dir.resolved}"
        name = "ver2.0" delete = "true" />
<!-- List tags matching to pattern "marked" into "tags" variable -->
<gittag repository = "${repo.dir.resolved}"
    list = "true"
    outputProperty = "tags"
    pattern = "marked" />
```

C.24. GitLogTask

Show commit logs. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-log.html].

Table C.24: *Attributes*

Name	Type	Description	Default	Required
`gitPath`	`String`	Path to Git binary	/usr/bin/git	No
`repository`	`String`	Path to Git repository	n/a	Yes
`paths`	`String`	<paentry> arguments to git-log. Accepts one or more paths delimited by PATH_SEPARATOR	n/a	No
`outputProperty`	`String`	Property name to set with output value from git-log	n/a	No
`format`	`String`	Commit format. See --format of git-log. Can be one of `oneline`, `short`, `medium`, `full`, `fuller`, `email`, `raw` and `format:<string>`	medium	No
`date`	`String`	Date format. See --date of git-log.	n/a	No
`since`	`String`	<since> argument to git-log.	n/a	No
`until`	`String`	<until> argument to git-log.	HEAD	No
`stat`	`String`	Generate a diffstat. See --stat of git-log	n/a	No
`nameStatus`	`Boolean`	Names + status of changed files. See --name-status of git-log.	`false`	No
`maxCount`	`Integer`	Number of commits to show. See -<n>|-n|--max-count of git-log.	n/a	No

Name	Type	Description	Default	Required
noMerges	Boolean	Don't show commits with more than one parent. See --no-merges of git-log.	false	No

C.24.1. Example

```
<property name = "repo.dir" value = "./relative/path/to/repo" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />

<!-- clone repository -->
<gitclone
    repository = "git://github.com/path/to/repo/repo.git"
    targetPath = "${repo.dir.resolved}" />

<gitlog
    paths = "${repo.dir.resolved}"
    format = "oneline"
    maxCount = "2"
    stat = "true"
    noMerges = "false"
    since = "Sun Jan 23 23:55:42 2011 +0300"
    until = "Mon Jan 24 09:59:33 2011 +0300"
    outputProperty = "logs"
    repository = "${repo.dir.resolved}" />
```

C.25. GrowlNotifyTask

When you have a long process and want to be notified when it is finished, without to stay focused on the console windows. Then use the GrowlNotify task.

This task requires the PEAR Net_Growl [http://pear.php.net/package/Net_Growl] package installed (version 2.6.0).

Features

- Compatible Windows and Mac/OSX
- Do not forget notification with sticky option
- Define priority of messages
- Send notification on private or public network

Table C.25: *Attributes*

Name	Type	Description	Default	Required
name	String	Name of application to be register	Growl for Phing	No
sticky	Boolean	Indicates if the notification should be sticky on desktop	false	No
message	String	Text of notification. Use \n to specify a linen break	n/a	Yes
title	String	Title of notification	GrowlNotify	No

Name	Type	Description	Default	Required
`notification`	`String`	The notification name/type	General Notification	No
`appicon`	`String`	• absolute url (http://domain/image.png) • absolute file path (c:\temp\image.png) • relative file path (.\folder\image.png)	n/a	No
`host`	`String`	The host address where to send the notification	127.0.0.1	No
`password`	`String`	The password required to send notifications over network	n/a	No
`priority`	`String`	The notification priority. Valid values are : • low • moderate • normal • high • emergency	normal	No
`protocol`	`String`	The protocol used to send the notification. May be either gntp or udp.	gntp	No
`icon`	`String`	The icon to show for the notification. Must be a valid file type (png, jpg, gif, ico). Can be any of the following: • absolute url (http://domain/image.png) • absolute file path (c:\temp\image.png) • relative file path (.\folder\image.png)	embeded growl icon v2	No

C.25.1. Examples

Send a single notification on a remote host

Both sender and Growl client (Mac or Windows) should share the same password.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "phing-GrowlNotifyTask" basedir = "." default = "notification">

    <taskdef name = "growlnotify" classname = "phing.tasks.ext.GrowlNotifyTask" />

    <target name = "notification"
        description = "display a single message with growl gntp over network"
    >
        <growlnotify message = "Deployment of project LAMBDA is finished."
            host = "192.168.1.2"
            password = "seCretPa$$word"
        />
    </target>
```

```
</project>
```

Send a single notification with UDP protocol

When you don't have a Macintosh, OS compatible with Growl GNTP, you should use the basic UDP protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "phing-GrowlNotifyTask" basedir = "." default = "notification">

    <taskdef name = "growlnotify" classname = "phing.tasks.ext.GrowlNotifyTask" />

    <target name = "notification"
        description = "display a single message with growl udp over network"
    >
        <growlnotify message = "Notify my MAC that does not accept GNTP."
            host = "192.168.1.2"
            password = "seCretPa$$word"
            protocol = "udp"
        />
    </target>

</project>
```

Send an important notification

If you want to send a notification that is so important that you don't want to missed it, even if you are away from your computer. Use the sticky attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "phing-GrowlNotifyTask" basedir = "." default = "notification">

    <taskdef name = "growlnotify" classname = "phing.tasks.ext.GrowlNotifyTask" />

    <target name = "notification"
        description = "display a sticky message on desktop"
    >
        <growlnotify message = "Project LAMDBA, unit tests FAILED."
            priority = "high"
            sticky = "true"
        />
    </target>

</project>
```

Use your icons to identify an application

You may customize the Growl notification system, with different icons and more.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "phing-GrowlNotifyTask" basedir = "." default = "notification">

    <taskdef name = "growlnotify" classname = "phing.tasks.ext.GrowlNotifyTask" />

    <target name = "notification"
        description = "display a custom icon message"
    >
        <growlnotify message = "Have a look on my beautiful message!"
            name = "phing Notifier"
            title = "phing notification"
            priority = "low"
            sticky = "false"
            appicon = "../images/my_icon.png"
        />
    </target>
```

```
</project>
```

C.26. HttpGetTask

This task will download a file through HTTP GET and save it to a specified directory. You need an installed version of HTTP_Request2 to use this task.

Table C.26: *Attributes*

Name	Type	Description	Default	Required
`url`	`String`	The request URL	n/a	Yes
`dir`	`String`	The directory to save the file	n/a	Yes
`filename`	`String`	The filename for the downloaded file	The file-name part of the URL	No
`followRedirects`	`Boolean`	Whether to follow HTTP redirects	false	No
`sslVerifyPeer`	`Boolean`	Whether to verify SSL certificates	true	No

C.26.1. Example

```
<httpget url = "http://buildserver.com/builds/latest.stable.tar.bz2" dir = "/usr/local/lib"/>
```

C.27. HttpRequestTask

This task will make an HTTP request to the provided URL and match the response against the provided regular expression. If an regular expression is provided and doesn't match the build will fail. You need an installed version of HTTP_Request2 to use this task.

Table C.27: *Attributes*

Name	Type	Description	Default	Required
`url`	`String`	The request URL	n/a	Yes
`responseRegex`	`String`	The regular expression for matching the response	n/a	No
`authUser`	`String`	The authentication user name	n/a	No
`authPassword`	`String`	The authentication password	n/a	No
`authScheme`	`String`	The authentication scheme	basic	No
`verbose`	`Boolean`	Whether to enable detailed logging	`false`	No
`observerEvents`	`String`	Comma-separated list of events to log when `verbose` is set to `true`	connect, sentHeaders, sentBodyPart, received-	No

Name	Type	Description	Default	Required
			Headers, received-Body, disconnect	
`method`	`String`	The HTTP method of the request, currently only GET or POST supported	`GET`	No

C.27.1. Example

```
<http-request url = "http://my-production.example.com/check-deployment.php"/>
```

Just perform a HTTP request to the given URL.

```
<http-request
   url = "http://my-production.example.com/check-deployment.php"
   responseRegex = "/Heartbeat/"
   verbose"true"
   observerEvents = "connect, disconnect"/>
```

Perform a HTTP request to the given URL and matching the response against the given regex pattern. Enable detailed logging and log only the specified events.

```
<http-request url = "http://my-production.example.com/check-deployment.php">
   <config name = "adapter" value = "HTTP_Request2_Adapter_Curl"/>
   <header name = "user-agent" value = "Phing HttpRequestTask"/>
 </http-request>
```

Perform a HTTP request to the given URL. Setting request adapter to curl instead of socket. Setting an additional header.

```
<http-request
            url = "http://my-production.example.com/check-deployment.php"
            verbose"true"
            method = "POST">
              <postparameter name = "param1" value = "value1" />
              <postparameter name = "param2" value = "value2" />
            </http-request>
```

Perform an HTTP POST request to the given URL. Setting POST request parameters to emulate form submission.

C.27.2. Supported Nested Tags

- config

  Holds additional config data. See HTTP_Request2 for supported values.

  Table C.28: *Attributes*

Name	Type	Description	Default	Required
`name`	`String`	Config parameter name	n/a	Yes
`value`	Mixed	Config value	n/a	Yes

- header

Holds additional header `name` and `value`.

Table C.29: *Attributes*

Name	Type	Description	Default	Required
`name`	`String`	Header name	n/a	Yes
`value`	`String`	Header value	n/a	Yes

C.28. IoncubeEncoderTask

The `IoncubeEncoderTask` executes the ionCube [http://www.ioncube.com] encoder (for either PHP4 or PHP5 projects).

For more information on the meaning of the various options please consult the ionCube user guide [http://www.ioncube.com/USER-GUIDE.pdf].

Table C.30: *Attributes*

Name	Type	Description	Default	Required
`allowedserver`	`String`	Restricts the encoded files to particular servers and/or domains. Consult the IonCude documentation for more information.	none	No
`binary`	`Boolean`	Whether to save encoded files in binary format (default is ASCII format)	`false`	No
`copy`	`String`	Specifies files or directories to exclude from being encoded or encrypted and copy them to the target directory (separated by space).	none	No
`encode`	`String`	Specifies additional file patterns, files or directories to encode, or to reverse the effect of `copy`	none	No
`encrypt`	`String`	Specify files or directories (space separated list) that are to be encrypted.	none	No
`expirein`	`String`	Sets a period in seconds (s), minutes (m), hours (h) or days (d) after which the files expire. Accepts: `500s` or `55m` or `24h` or `7d`	none	No
`expireon`	`String`	Sets a YYYY-MM-DD date to expire the files.	none	No
`fromdir`	`String`	Path containing source files	none	Yes
`ignore`	`String`	Set files and directories to ignore entirely and exclude from the target directory (separated by space).	none	Yes
`ioncubepath`	`String`	Path to the ionCube binaries	/usr/local/ioncube	No
`keep`	`String`	Set files and directories not to be ignored (separated by space).	none	No
`licensepath`	`String`	Path to the license file that will be used by the encoded files	none	No

Name	Type	Description	Default	Required
nodoccomments	String	Omits documents comments (/** ... */) from the encoded files.	none	No
obfuscation-key	String	The obfuscation key must be supplied when using the obfuscate option	none	No
obfuscate	String	The Encoder can obfuscate the names of global functions, the names of local variables in global functions, and line numbers. Use either all or any of functions, locals or linenos separated by a space.	none	No
optimize	String	Controls the optimization of the encoded files, accepts either more or max	none	No
passphrase	String	The passphrase to use when encoding with a license file	none	No
phpversion	Integer	The PHP version to use	5	No
targetoption	String	Option to use when target directory exists, accepts replace, merge, update and rename	none	No
todir	String	Path to save encoded files to	none	Yes
withoutruntimeloadersupport	Boolean	Whether to disable support for runtime initialization of the ionCube Loader	false	No
noshortopentags	Boolean	Whether to disable support for short PHP tags	false	No
callbackfile	String	Path to callback file (.php)	n/a	No
obfuscationexclusionsfile	String	Path to obfuscation exclusions file	n/a	No
ignoredeprecatedwarnings	Boolean	Whether to ignore deprecated warnings	false	No
ignorestrictwarnings	Boolean	Whether to ignore strict warnings	false	No
allowencodingintosource	Boolean	Whether to allow encoding into the source tree	false	No
messageifnoloader	String	A valid PHP expression to customize the "non loader installed" message	n/a	No
actionifnoloader	String	A valid PHP expression to replace the "non loader installed" action	n/a	No
showcommandline	Boolean	whether to show command line before it is executed	false	No

C.28.1. Example

```
<ioncubeencoder
   binary = "true"
   copy = "*.ini config/*"
   encode = "*.inc licenses/license.key"
   encrypt = "*.tpl *.xml"
```

```
   fromdir = "files"
   ignore = "*.bak RCS/ *~ docs/"
   ioncubepath = "/usr/local/ioncube"
   keep = "docs/README"
   licensepath = "mylicense.txt"
   optimize = "max"
   passphrase = "mypassphrase"
   phpversion = "4"
   noshortopentags = "false"
   targetoption = "replace"
   todir = "encoded"
   withoutruntimeloadersupport = "true"
   callbackfile = "errhandler.php"
   obfuscationexlusionsfile = "obfex.txt">
  <comment>A project encoded with the ionCube encoder.</comment>
</ioncubeencoder>
```

C.28.2. Supported Nested Tags

- comment

  Custom text that is added to the start of each encoded file.

C.29. IoncubeLicenseTask

The `IoncubeLicenseTask` executes the ionCube [http://www.ioncube.com] make_license program.

For more information on the meaning of the various options please consult the ionCube user guide [http://www.ioncube.com/USER-GUIDE.pdf].

Table C.31: *Attributes*

Name	Type	Description	Default	Required
`ioncubepath`	`String`	Path to the ionCube binaries	/usr/local/ioncube	No
`licensepath`	`String`	Path to the license file that will be generated	none	No
`passphrase`	`String`	The passphrase to use when generating the license file	none	No
`allowedserver`	`String`	Restricts the license to particular servers and/or domains. Consult the IonCude documentation for more information.	none	No
`expirein`	`String`	Sets a period in seconds (s), minutes (m), hours (h) or days (d) after which the license expires. Accepts: 500s or 55m or 24h or 7d.	none	No
`expireon`	`String`	Sets a YYYY-MM-DD date to expire the license.	none	No

C.29.1. Example

```
<ioncubelicense
   ioncubepath = "/usr/local/ioncube"
```

```
    licensepath = "mylicense.txt"
    passphrase = "mypassphrase"
    allowedserver = "00:06:4F:01:8F:2C"
    expireon = "2010-09-01"
    expirein = "7d">
  <comment>A license file made with the ionCube encoder.</comment>
</ioncubelicense>
```

C.29.2. Supported Nested Tags

- comment

  Custom text that is added to the start of each encoded file.

C.30. JslLintTask

The `JslLintTask` uses the Javascript Lint [http://www.javascriptlint.com] program to check the sytax on one or more JavaScript source code files.

NB: the Javascript lint program must be in the system path!

Table C.32: *Attributes*

Name	Type	Description	Default	Required
`executable`	`String`	Path to JSL executable	jsl	No
`file`	`String`	Path to source file	n/a	No
`haltonfailure`	`Boolean`	Stop the build process if the linting process encounters an error.	`false`	No
`haltonwarning`	`Boolean`	Stop the build process if the linting process encounters a warning.	`false`	No
`showwarnings`	`Boolean`	Sets the flag if warnings should be shown.	`true`	No
`cachefile`	`String`	If set, enables writing of last-modified times to `cachefile`, to speed up processing of files that rarely change	none	No
`conffile`	`String`	Path to JSL config file	none	No
`tofile`	`String`	File to write list of 'bad files' to.	n/a	No

C.30.1. Example

```
<jsllint file = "path/to/source.php"/>
```

Checking syntax of one particular source file.

```
<jsllint>
  <fileset dir = "src">
    <include name = "**/*.php"/>
  </fileset>
</jsllint>
```

Check syntax of a fileset of source files.

C.30.2. Supported Nested Tags

- `fileset`

C.31. JsMinTask

The `JsMinTask` minifies JavaScript files using JsMin [http://code.google.com/p/jsmin-php/]. JsMin is bundled with Phing and does not need to be installed separately.

For more information on minifying JavaScript files see Douglas Crockford's introduction to minifying JavaScript files [http://www.crockford.com/javascript/jsmin.html].

Table C.33: *Attributes*

Name	Type	Description	Default	Required
`targetDir`	`String`	Path where to store minified JavaScript files	none	Yes
`suffix`	`String`	Suffix to append to the filenames.	-min	No
`failonerror`	`Boolean`	Whether an error while minifying a JavaScript file should stop the build or not	`false`	No

C.31.1. Example

```
<jsMin targetDir = "docroot/script/minified" failOnError = "false">
  <fileset dir = "docroot/script">
    <include name = "**/*.js"/>
  </fileset>
</jsMin>
```

C.31.2. Supported Nested Tags

- `fileset`

  JavaScript files to be minified.

C.32. LiquibaseChangeLogTask

The `LiquibaseChangeLogTask` writes the Change Log XML to copy the current state of the database to the given changeLogFile.

Table C.34: *Attributes*

Name	Type	Description	Default	Required
`jar`	`String`	Location of the Liquibase jar file.	n/a	Yes
`classpath`	`String`	Additional classpath entries.	n/a	Yes
`changeLogFile`	`String`	Location of the changelog file in which the changes get written or read from.	n/a	Yes

Name	Type	Description	Default	Required
username	String	The username needed to connect to the database.	n/a	Yes
password	String	The password needed to connect to the database.	n/a	Yes
url	String	The JDBC Url representing the databasen/a datasource, e.g jdbc:mysql://localhost/mydatabase		Yes

C.32.1. Example

```
<liquibase-changelog
  jar = "/usr/local/lib/liquibase/liquibase.jar"
  classpathref = "/usr/local/lib/liquibase/lib/mysql-connector-java-5.1.15-bin.jar"
  changelogFile = "./changelogTest.xml"
  username = "liquibase"
  password = "liquibase"
  url = "jdbc:mysql://localhost/mydatabase"
/>
```

C.33. LiquibaseDbDocTask

The LiquibaseDbDocTask generates a Javadoc-like documentation based on current database and the given changelog file.

Table C.35: *Attributes*

Name	Type	Description	Default	Required
jar	String	Location of the Liquibase jar file.	n/a	Yes
classpath	String	Additional classpath entries.	n/a	Yes
changeLogFile	String	Location of the changelog file in which then/a changes get written or read from.		Yes
username	String	The username needed to connect to the database.	n/a	Yes
password	String	The password needed to connect to the database.	n/a	Yes
url	String	The JDBC URL representing the databasen/a data source, e.g jdbc:mysql://localhost/mydatabase		Yes
outputDir	String	Absolute path where the documentation getsn/a written to. If the given directory does not exist, it get`s created automatically.		Yes

C.33.1. Example

```
<liquibase-dbdoc
```

```
  jar = "/usr/local/lib/liquibase/liquibase.jar"
  classpathref = "/usr/local/lib/liquibase/lib/mysql-connector-java-5.1.15-bin.jar"
  changelogFile = "./changelogTest.xml"
  username = "liquibase"
  password = "liquibase"
  url = "jdbc:mysql://localhost/mydatabase"
  outputDir = "/tmp/generateddocs"
/>
```

C.34. LiquibaseDiffTask

The `LiquibaseDiffTask` creates a diff between two databases. Will output the changes needed to convert the reference database to the state of the database.

Table C.36: *Attributes*

Name	Type	Description	Default	Required
`jar`	`String`	Location of the Liquibase jar file.	n/a	Yes
`classpath`	`String`	Additional classpath entries.	n/a	Yes
`changeLogFile`	`String`	Location of the changelog file in which the changes get written or read from.	n/a	Yes
`username`	`String`	The username needed to connect to the database.	n/a	Yes
`password`	`String`	The password needed to connect to the database.	n/a	Yes
`url`	`String`	The JDBC Url representing the databasen/a datasource, e.g `jdbc:mysql://localhost/mydatabase`		Yes
`referenceUsername`	`String`	The username needed to connect to the reference database.	n/a	Yes
`referencePassword`	`String`	The password needed to connect to the reference database.	n/a	Yes
`referenceUrl`	`String`	The JDBC Url representing the database reference datasource, e.g `jdbc:mysql://localhost/refdatabase`	n/a	Yes

C.34.1. Example

```
<liquibase-diff
  jar = "/usr/local/lib/liquibase/liquibase.jar"
  classpathref = "/usr/local/lib/liquibase/lib/mysql-connector-java-5.1.15-bin.jar"
  changelogFile = "./changelogTest.xml"
  username = "liquibase"
  password = "liquibase"
  url = "jdbc:mysql://localhost/mydatabase"
  referenceUsername = "liquibase"
  referencePassword = "liquibase"
  referenceUrl = "jdbc:mysql://localhost/refdatabase"
/>
```

C.35. LiquibaseRollbackTask

The `LiquibaseRollbackTask` rolls back the database to the state is was when the tag was applied.

Table C.37: *Attributes*

Name	Type	Description	Default	Required
`jar`	`String`	Location of the Liquibase jar file.	n/a	Yes
`classpath`	`String`	Additional classpath entries.	n/a	Yes
`changeLogFile`	`String`	Location of the changelog file in which the changes get written or read from.	n/a	Yes
`username`	`String`	The username needed to connect to the database.	n/a	Yes
`password`	`String`	The password needed to connect to the database.	n/a	Yes
`url`	`String`	The JDBC Url representing the databasen/a datasource, e.g `jdbc:mysql://localhost/mydatabase`		Yes
`rollbackTag`	`String`	The name of the tag to roll the database back to.	n/a	Yes

C.35.1. Example

```
<liquibase-rollback
  jar = "/usr/local/lib/liquibase/liquibase.jar"
  classpathref = "/usr/local/lib/liquibase/lib/mysql-connector-java-5.1.15-bin.jar"
  changelogFile = "./changelogTest.xml"
  username = "liquibase"
  password = "liquibase"
  url = "jdbc:mysql://localhost/mydatabase"
  rollbackTag = "tag_0_1"
/>
```

C.36. LiquibaseTagTask

The `LiquibaseTagTask` tags the current database state for future rollback.

Table C.38: *Attributes*

Name	Type	Description	Default	Required
`jar`	`String`	Location of the Liquibase jar file.	n/a	Yes
`classpath`	`String`	Additional classpath entries.	n/a	Yes
`changeLogFile`	`String`	Location of the changelog file in which the changes get written or read from.	n/a	Yes
`username`	`String`	The username needed to connect to the database.	n/a	Yes
`password`	`String`	The password needed to connect to the database.	n/a	Yes

Name	Type	Description	Default	Required
`url`	`String`	The JDBC Url representing the database datasource, e.g `jdbc:mysql://localhost/mydatabase`	n/a	Yes
`tag`	`String`	The name of the tag to apply.	n/a	Yes

C.36.1. Example

```
<liquibase-tag
  jar = "/usr/local/lib/liquibase/liquibase.jar"
  classpathref = "/usr/local/lib/liquibase/lib/mysql-connector-java-5.1.15-bin.jar"
  changelogFile = "./changelogTest.xml"
  username = "liquibase"
  password = "liquibase"
  url = "jdbc:mysql://localhost/mydatabase"
  tag = "tag_0_1"
/>
```

C.37. LiquibaseUpdateTask

The `LiquibaseUpdateTask` applies the latest changes from the changelog file to the definied database.

Table C.39: *Attributes*

Name	Type	Description	Default	Required
`jar`	`String`	Location of the Liquibase jar file.	n/a	Yes
`classpath`	`String`	Additional classpath entries.	n/a	Yes
`changeLogFile`	`String`	Location of the changelog file in which the changes get written or read from.	n/a	Yes
`username`	`String`	The username needed to connect to the database.	n/a	Yes
`password`	`String`	The password needed to connect to the database.	n/a	Yes
`url`	`String`	The JDBC Url representing the database datasource, e.g `jdbc:mysql://localhost/mydatabase`	n/a	Yes

C.37.1. Example

```
<liquibase-update
  jar = "/usr/local/lib/liquibase/liquibase.jar"
  classpathref = "/usr/local/lib/liquibase/lib/mysql-connector-java-5.1.15-bin.jar"
  changelogFile = "./changelogTest.xml"
  username = "liquibase"
  password = "liquibase"
  url = "jdbc:mysql://localhost/mydatabase"
/>
```

C.38. MailTask

A task to send email.

Table C.40: Attributes

Name	Type	Description	Default	Required
`from`	`String`	Email address of sender.	none	Yes
`tolist`	`String`	Comma-separated list of recipients.	none	Yes
`message`	`String`	Message to send in the body of the email.	none	No
`subject`	`String`	Email subject line.	none	No

C.38.1. Example

```
<mail tolist = "user@example.org" subject = "build complete">The build process is a success
```

C.38.2. Supported Nested Tags

- `fileset`

  Files to be attached (not implemented at this time).

C.39. ParallelTask

Executes nested tasks in parallel.

Warning

This task is highly experimental, and will only work on *nix machines that have the PHP `pcntl` extension installed.

Table C.41: Attributes

Name	Type	Description	Default	Required
`threadCount`	`Integer`	Maximum number of threads / processes to use.	n/a	No

C.39.1. Example

```
<parallel threadCount = "4">
    <echo>Job 1</echo>
    <echo>Job 2</echo>
    <echo>Job 3</echo>
    <echo>Job 4</echo>
</trycatch>\u0000
```

C.40. PatchTask

The `PatchTask` uses the patch [http://savannah.gnu.org/projects/patch] program to apply diff file to originals.

NB: the `patch` program must be in the system path!

Table C.42: *Attributes*

Name	Type	Description	Default	Required
`patchfile`	`String`	File that includes the diff output	n/a	Yes
`originalfile`	`String`	File to patch. If not specified Task tries to guess it from the diff file	none	No
`destfile`	`String`	File to send the output to instead of patching the file in place	none	No
`backups`	`Boolean`	Keep backups of the unpatched files	`false`	No
`quiet`	`Boolean`	Work silently unless an error occurs	`false`	No
`reverse`	`Boolean`	Assume patch was created with old and new files swapped	`false`	No
`ignorewhite-space`	`Boolean`	Ignore whitespace differences	`false`	No
`strip`	`Integer`	Strip the smallest prefix containing specified number of leading slashes from filenames	none	No
`dir`	`String`	The directory in which to run the patch com-mand	none	No
`haltonfailure`	`Boolean`	Stop the build process if the patching process encounters an error.	`false`	No

C.40.1. Example

```
<patch
  patchfile = "/path/to/patches/file.ext.patch"
  dir = "/path/to/original"
/>
```

Apply "file.ext.path" to original file locataed in "/path/to/original" folder.

C.41. PDOSQLExecTask

The `PDOSQLExecTask` executes SQL statements using PDO.

Note

The combination of large SQL files and `delimitertype` set to `normal` can trigger segmentation faults with large files.

Table C.43: *Attributes*

Name	Type	Description	Default	Required
`url`	`String`	PDO connection URL (DSN)	none	Yes
`userid`	`String`	Username for connection (if it cannot be specified in URL)	none	No
`password`	`String`	The password to use for the connection (if it cannot be specified in URL)	none	No
`src`	File	A single source file of SQL statements to execute.	none	No
`onerror`	`String`	The action to perform on error (continue, stop, or abort)	abort	No
`delimiter`	`String`	The delimeter to separate SQL statements; (e.g. "GO" in MSSQL)		No
`delimitertype`	`String`	The delimiter type ("normal", "row" or "none"). Normal means that any occurence 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. None disables all delimiter detection.	none	No
`autocommit`	`Boolean`	Whether to autocommit every single statement.	`false`	No
`encoding`	`String`	Encoding to use for read SQL files	none	No

C.41.1. Example

```
<pdosqlexec url = "pgsql:host=localhost dbname=test">
  <fileset dir = "sqlfiles">
   <include name = "*.sql"/>
   </fileset>
</pdosqlexec>
```

```
<pdosqlexec url = "mysql:host=localhost;dbname=test" userid = "username" password = "password">
  <transaction src = "path/to/sqlfile.sql"/>
  <formatter type = "plain" outfile = "path/to/output.txt"/>
  </pdosqlexec>
```

Note

Because of backwards compatiblity, the PDOSQLExecTask can also be called using the `'pdo'` statement.

```
<pdo url = "pgsql:host=localhost dbname=test">
  <fileset dir = "sqlfiles">
   <include name = "*.sql"/>
  </fileset>

  <!-- xml formatter -->
  <formatter type = "xml" output = "output.xml"/>

  <!-- custom formatter -->
```

```
  <formatter classname = "path.to.CustomFormatterClass">
    <param name = "someClassAttrib" value = "some-value"/>
  </formatter>

  <!-- No output file + usefile=false means it goes to phing log -->
  <formatter type = "plain" usefile = "false" />
</pdo>
```

C.41.2. Supported Nested Tags

- `transaction`

  Wrapper for a single transaction. Transactions allow several files or blocks of statements to be executed using the same PDO connection and commit operation in between.

  Table C.44: *Attributes*

Name	Type	Description	Default	Required
`tsrcfile`	`String`	File with statements to be run as one transaction	n/a	No

- `fileset`

  Files containing SQL statements.

- `filelist`

  Files containing SQL statements.

- `formatter`

  The results of any queries that are executed can be printed in different formats. Output will always be sent to a file, unless you set the `usefile` attribute to `false`. The path to the output file can be specified by the `outfile` attribute; there is a default filename that will be returned by the formatter if no output file is specified.

  There are three predefined formatters - one prints the query results in XML format, the other emits plain text. Custom formatters that extend phing.tasks.pdo.PDOResultFormatter can be specified.

  Table C.45: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	Use a predefined formatter (either `xml` or `plain`).	n/a	One of these attributes is required.
`classname`	`String`	Name of a custom formatter class (must extend phing.tasks.ext.pdo.PDOResultFormatter).	n/a	
`usefile`	`Boolean`	Boolean that determines whether output should be sent to a file.	`true`	No
`outfile`	File	Path to file in which to store result.	Depends on formatter	No
`showheaders`	`Boolean`	(only applies to plain formatter) Whether to show column headers.	`false`	No
`coldelim`	`String`	(only applies to plain formatter) The column delimiter.	,	No

Name	Type	Description	Default	Required
`rowdelim`	`String`	(only applies to plain formatter) The row de-limiter.	\n	No
`encoding`	`String`	(only applies to XML formatter) The xml doc-ument encoding.	(PHP de-fault)	No
`formatoutput`	`Boolean`	(only applies to XML formatter) Whether to format XML output.	`true`	No

C.42. PearPackageTask

With the PearPackageTask, you can create a package.xml which can be installed using the PEAR installer. Use this in conjunction with the Section C.75, “TarTask” to completely script the building of a PEAR package.

Note

Note that this task creates a `version 1` package.xml file.

This task uses the `PEAR_PackageFileManager` class. In order to be maximally flexible, the majority of options are set generically (using `<option>` tag) and are set using `PEAR_PackageFileManager::setOptions()`. Use the `<mapping>` tag to represent complex values (which are turned into associative arrays and also set using `setOptions()` method).

Table C.46: *Attributes*

Name	Type	Description	Default	Required
`name`	`String`	The name of the PEAR package.	n/a	Yes
`dir`	`String`	The base directory of files to add to package.	n/a	Yes
`destFile`	`String`	The file to create.	package.xml in base di-rectory	No

C.42.1. Example

```
<pearpkg name = "phing" dir = "${build.src.dir}" destFile = "${build.base.dir}/package.xml">
<fileset dir = ".">
  <include name = "**"/>
</fileset>
<option name = "notes">Sample release notes here.</option>
<option name = "description">Package description</option>
<option name = "summary">Short description</option>
<option name = "version" value = "2.0.0b1"/>
<option name = "state" value = "beta"/>
 <mapping name = "maintainers">
  <element>
   <element key = "handle" value = "hlellelid"/>
```

```
    <element key = "name" value = "Hans"/>
    <element key = "email" value = "hans@xmpl.org"/>
    <element key = "role" value = "lead"/>
   </element>
</mapping>
</pearpkg>
```

C.42.2. Supported Nested Tags

- `fileset`
- `option`
- `mapping`

  The <mapping> tag represents a complex data type. You can use nested <mapping> (and nested <element> with <element> tags) to represent the full complexity of the structure. Bear in mind that what you are creating will be mapped to an associative array that will be passed in via `PEAR_PackageFileMaintainer::setOptions()` .

  ```
  <mapping name = "option_name">
      <element key = "key_name" value = "key_val"/>
      <element key = "key_name" value = "key_val"/>
  </mapping>
  ```

- `role` See PEAR_PackageFileManager::addRole [http://pear.php.net/manual/en/package.pear.pear-packagefilemanager.pear-packagefilemanager.addrole.php] for more information.

Table C.47: *Available options*

Name	Type	Description	Default	Required
`extension`	`String`	The file extension	n/a	Yes
`role`	`String`	The file extension	n/a	Yes

C.43. PearPackage2Task

With the PearPackage2Task, you can create a `version 2` package.xml which can be installed using the PEAR installer. Use this in conjunction with the TarTask [#TarTask] to completely script the building of a PEAR package.

This task uses the PEAR_PackageFileManager2 class. In order to be maximally flexible, the majority of options are set generically (using `<option>` tag) and are set using PEAR_PackageFileManager::setOptions(). Use the `<mapping>` tag to represent complex values.

Note that Travis Swicegood has created a more complete implementation of this functionality which can be found here: pear.domain51.com [http://pear.domain51.com/].

Table C.48: *Attributes*

Name	Type	Description	Default	Required
`name`	`String`	The name of the PEAR package.	n/a	Yes

Name	Type	Description	Default	Required
dir	String	The base directory of files to add to package.	n/a	Yes

C.43.1. Example

```
<pearpkg2 name = "phing" dir = "${build.src.dir}">
   <option name = "outputdirectory" value = "./build"/>
   <option name = "packagefile" value = "package2.xml"/>
   <option name = "packagedirectory" value = "./${build.dist.dir}"/>
   <option name = "baseinstalldir" value = "${pkg.prefix}"/>
   <option name = "channel" value = "my.pear-channel.com"/>
   <option name = "summary" value = "${pkg.summary}"/>
   <option name = "description" value = "${pkg.description}"/>
   <option name = "apiversion" value = "${pkg.version}"/>
   <option name = "apistability" value = "beta"/>
   <option name = "releaseversion" value = "${pkg.version}"/>
   <option name = "releasestability" value = "beta"/>
   <option name = "license" value = "none"/>
   <option name = "phpdep" value = "5.0.0"/>
   <option name = "pearinstallerdep" value = "1.4.6"/>
   <option name = "packagetype" value = "php"/>
   <option name = "notes" value = "${pkg.relnotes}"/>
   <mapping name = "maintainers">
    <element>
     <element key = "handle" value = "hlellelid"/>
     <element key = "name" value = "Hans"/>
     <element key = "email" value = "hans@xmpl.org"/>
     <element key = "role" value = "lead"/>
    </element>
   </mapping>
</pearpkg2>
```

C.43.2. Supported Nested Tags

- `fileset`
- `option`

Table C.49: *Available options*

Name	Type	Description	Default	Required
summary	String		n/a	Yes
description	String		n/a	Yes
license	String		n/a	Yes
channel	String	Channel name (not alias!). Must be registered (`pear channel-discover channel`) on the machine, where the build will be.	n/a	Yes
apiversion	String		n/a	Yes
releaseversion	String		n/a	Yes
releasestability	String	One from: snapshot, devel, alpha, beta or stable.	n/a	Yes
apistability	String	One from: devel, alpha, beta or stable.	n/a	Yes
note	String		n/a	Yes

Name	Type	Description	Default	Required
packagetype	String		n/a	Yes
phpdep	String		n/a	Yes
pearin-stallerdep	String		n/a	Yes

- mapping

  The <mapping> tag represents a complex data type. You can use nested <mapping> (and nested <element> with <element> tags) to represent the full complexity of the structure. Bear in mind that what you are creating will be mapped to an associative array that will be passed in via PEAR_PackageFileMaintainer::setOptions() .

```
<mapping name = "option_name">
    <element key = "key_name" value = "key_val"/>
    <element key = "key_name" value = "key_val"/>
</mapping>
```

  Available mappings and they structures:

  - deps (optional) see PEAR_PackageFileManager::addDependency() [http://pear.php.net/manual/en/package.pear.pear-packagefilemanager.pear-packagefilemanager.adddependency.php] for more info

Table C.50: *Parameters*

Name	Type	Description	Default	Required
channel	String	Channel name, from package is.	n/a	Yes
name	String	Package name in channel.	n/a	Yes
version	String	Minimal version.	n/a	Yes
max	String	Maximum version.	Same as version.	No
recommended	String	Recommended version.	Same as version.	No

  - extdeps (optional) see PEAR_PackageFileManager::addDependency() [http://pear.php.net/manual/en/package.pear.pear-packagefilemanager.pear-packagefilemanager.adddependency.php] for more info

Table C.51: *Parameters*

Name	Type	Description	Default	Required
name	String	Package name.	n/a	Yes
version	String	Minimal version.	n/a	Yes
max	String	Maximum version.	Same as version.	No
recommended	String	Recommended version.	Same as version.	No

  - maintainers (required at least one) see PEAR_PackageFileManager::addMaintainer() [http://pear.php.net/manual/en/package.pear.pear-packagefilemanager.pear-packagefilemanager.addmaintainer.php] for more info

Table C.52: *Parameters*

Name	Type	Description	Default	Required
`handle`	`String`	User identifier in channel.	n/a	Yes
`name`	`String`	Real name.	n/a	Yes
`email`	`String`		n/a	Yes
`role`	`String`	One from: lead, developer, contributor orn/a helper.		Yes

- `replacements` (optional) see PEAR_PackageFileManager::addReplacement() [http://pear.php.net/manual/en/package.pear.pear-packagefilemanager.pear-packagefilemanager.addreplacement.php] for more info

Table C.53: *Parameters*

Name	Type	Description	Default	Required
`path`	`String`	Relative path of file.	n/a	Yes
`type`	`String`	Variable type, either php-const, pear-confign/a or package-info.		Yes
`from`	`String`	Text to replace in the source file.	n/a	Yes
`to`	`String`	Variable name to use for replacement.	n/a	Yes

- `role` See PEAR_PackageFileManager::addRole [http://pear.php.net/manual/en/package.pear.pear-packagefilemanager.pear-packagefilemanager.addrole.php] for more information.

  Available options:

Table C.54: *Parameters*

Name	Type	Description	Default	Required
`extension`	`String`	The file extension	n/a	Yes
`role`	`String`	The file extension	n/a	Yes

C.44. PharPackageTask

Phar [http://www.php.net/manual/en/book.phar.php] packages generating with Phing. This task require PECL's Phar [http://pecl.php.net/package/phar] extension to be installed on your system. Phar is built-in in PHP from 5.3 version.

Table C.55: *Attributes*

Name	Type	Description	Default	Required
`basedir`	`String`	Base directory, which will be deleted fromn/a each included file (from path). Paths with deleted `basedir` part are local paths in package.		Yes
`destfile`	`String`	Destination (output) file. Will be recreated, ifn/a exists!		Yes

Name	Type	Description	Default	Required
`compression`	`String`	Compression type (gzip, bzip2, none) to apply to the packed files.	none	No
`webstub`	`String`	Relative path within the phar package to run, if accessed through a web browser.	n/a	No
`clistub`	`String`	Relative path within the phar package to run, if accessed on the command line.	n/a	No
`stub`	`String`	A path to a php file that contains a custom stub	n/a	No
`alias`	`String`	An alias to assign to the phar package	n/a	No
`signature`	`String`	Signature algorithm (md5, sha1, sha256, sha512), used for this package.	sha1	No

C.44.1. Example

Sample build command:

```
<pharpackage
  destfile = "./build/package.phar"
  basedir = "./">
  <fileset dir = "./classes">
    <include name = "**/**" />
  </fileset>
  <metadata>
    <element name = "version" value = "1.0" />
    <element name = "authors">
      <element name = "John Doe">
        <element name = "e-mail" value = "john@example.com" />
      </element>
    </element>
  </metadata>
</pharpackage>
```

C.44.2. Supported Nested Tags

- `fileset`
- metadata

C.45. PhkPackageTask

This task runs PHK_Creator.phk to build PHK-package. Learn more about build process in PHK Builder's Guide [http://phk.tekwire.net/joomla/support/doc/builders_guide.htm].

Table C.56: *Attributes*

Name	Type	Description	Default	Required
`phkcreatorpath`	`String`	Path to PHK_Creator.phk.	n/a	Yes
`inputdirectory`	`String`	Path to directory, that will be packed.	n/a	Yes

Name	Type	Description	Default	Required
`outputfile`	`String`	Output PHK-file. Directory, where file will be stored, must exist!	n/a	Yes
`compress`	`String`	Compression type (gzip, bzip2, none) to apply to the packed files.	none	No
`strip`	`Boolean`	When true, PHP source file(s) are stripped (filtered through php_strip_whitespace()) before being stored into the archive.	`false`	No
`name`	`String`	The package's name (Information only).	n/a	No
`webrunscript`	`String`	The script to run in web direct access mode. Subfile path.	n/a	No
`crccheck`	`Boolean`	If true, a CRC check will be forced every time the package is mounted.	`false`	No

C.45.1. Example

Sample build command:

```
<phkpackage
    phkcreatorpath = "/path/to/PHK_Creator.phk"
    inputdirectory = "src"
    outputfile = "build/sample-project.phk"
    compress = "gzip"
    strip = "true"
    name = "Sample Project"
    webrunscript = "index.php">
    <webaccess>
        <paentry>/</paentry>
    </webaccess>
</phkpackage>
```

C.45.2. Supported Nested Tags

- `webaccess`

  Collection of `path` tags (see example below), that will be visible outside package in web mode.

C.46. PhpCodeSnifferTask

This task runs PHP_CodeSniffer [http://pear.php.net/package/PHP_CodeSniffer] to detect violations of a defined set of coding standards.

Table C.57: *Attributes*

Name	Type	Description	Default	Required
`standard`	`String`	The name of the standard to check for.	Generic	No
`format`	`String`	The output format. The `default` format is specified in the task itself. Additionally all	default	No Ignored if nested `for-`

Name	Type	Description	Default	Required
		report formats of PHP_CodeSniffer can be choosen (ex. `checkstyle`, `full`, `summary`, ...).		`matter` elements are supplied.
`showSniffs`	`Boolean`	Print the list of used sniffs.	`false`	No
`showWarnings`	`Boolean`	Print warnings.	`true`	No
`showSources`	`Boolean`	Flag that determines whether to show sources or not.	`true`	No
`docGenerator`	`String`	The name of the doc generator (HTML, Text).	n/a	No
`docFile`	`String`	Path to write output file to. If not set documentation will be written to STDOUT when `docGenerator` is set.	n/a	No
`file`	`String`	The file or folder to check (usually the nested tag `fileset` is used instead).	`false`	Either this attribute or the nested tag `fileset` is required.
`sniffs`	`String`	The list of allowed sniffs (separated by space, comma or semicolon). The sniffs must be part of the choosen standard.	n/a	No
`verbosity`	`Integer`	The verbosity level of CodeSniffer where level 1 prints progress information and level 2 prints developer debug information.	0	No
`encoding`	`String`	The encoding of the files to check	iso-8859-1	No
`tabWidth`	`Integer`	Replaces tabs with the given number of spaces. If zero no replacing is done.	0	No
`reportWidth`	`Integer`	The max. width for the report.	80	No
`allowedFileExtensions`	`String`	The allowed file extensions / file types to process. Separated by space, comma or semicolon.	php, inc, js, css	No
`ignorePatterns`	`String`	The patterns to ignore files and folders (separated by space, comma or semicolon) when a directory is specified in the `file` attribute.	n/a	No
`noSubdirectories`	`Boolean`	Do not recurse into subdirectories when a directory is specified in the `file` attribute.	`false`	No
`haltonerror`	`Boolean`	Stop the build process if errors occurred during the run.	`false`	No
`haltonwarning`	`Boolean`	Stop the build process if warnings occurred during the run.	`false`	No
`skipversioncheck`	`Boolean`	Skips the version check when the task starts.	`false`	No

C.46.1. Examples

Checks all files in the directory `file` matching the allowed file extension with the PEAR standard and prints the `summary` report without warnings.

```
<phpcodesniffer
  standard = "PEAR"
  format = "summary"
  file = "/path/to/source-files"
  allowedFileExtensions = "php php5 inc"/>
```

Checks all matching files in the `fileset` with the `Zend` standard, sets the `zend_ca_path` configuration which may be required by one of the sniffs, prints a list of used sniffs and prints the `default` report with warnings and the `checkstyle` report to `/path/to/checkstyle.xml`.

```
<phpcodesniffer
  standard = "Zend"
  showSniffs = "true"
  showWarnings = "true">
  <fileset dir = "/path/to/source-files">
    <include name = "**/*.php"/>
  </fileset>
  <config name = "zend_ca_path" value = "/path/to/ZendStudio/bin/ZendCodeAnalyzer"/>
  <formatter type = "default" usefile = "false"/>
  <formatter type = "checkstyle" outfile = "/path/to/checkstyle.xml"/>
</phpcodesniffer>
```

Checks all files in the directory `file` with the PEAR standard and prints the `checkstyle` report without warnings. It also generates the documentation for the selected coding standard and writes it to the given file.

```
<phpcodesniffer
  standard = "PEAR"
  file = "/path/to/source-files"
  docGenerator = "HTML"
  docFile = "/path/to/doc.html">
  <formatter type = "checkstyle" outfile = "/path/to/checkstyle.xml"/>
</phpcodesniffer>
```

C.46.2. Supported Nested Tags

- `fileset`

  Either this nested tag or the attribute `file` is required.

- `config`

  The configuration parameters which are usually loaded from the CodeSniffer.conf can be set.

Table C.58: *Attributes*

Name	Type	Description	Default	Required
`name`	`String`	Name of the configuration parameter.	n/a	Yes
`value`	`String`	Value of the configuration parameter.	n/a	Yes

- `formatter`

  The results of the tests can be printed in different formats. Output will always be sent to a file, unless you set the `usefile` attribute to `false`.

Table C.59: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	The output format. Accepts the same values as the `format` attribute (`default`, `xml`,	n/a	Yes

Name	Type	Description	Default	Required
		`checkstyle`, `csv`, `report`, `summary` & `doc`).		
`usefile`	`Boolean`	Boolean that determines whether output should be sent to a file.	`true`	No
`outfile`	`String`	Path to write output file to.	n/a	Yes, if `usefile` is `true`.

C.47. PHPCPDTask

This task runs phpcpd [http://github.com/sebastianbergmann/phpcpd/], a Copy/Paste Detector (CPD) for PHP Code. You need an installed version of this software to use this task.

Table C.60: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Path to source file or path	n/a	Only when there are *no* nested `fileset` elements
`minTokens`	`Integer`	Sets the minimum number of identical tokens (default: 70)	70	No
`minLines`	`Integer`	Sets the minimum number of identical lines (default: 5)	5	No
`format`	`String`	The format for the report when no nested for-matter is used.	default	No

C.47.1. Examples

```
<phpcpd file = "path/to/source.php"/>
```

Checking for copy/paste code in one particular source file. Sending Default-Report to STDOUT.

```
<phpcpd file = "path/to/source">
   <formatter type = "pmd" outfile = "reports/pmd-cpd.xml"/>
 </phpcpd>
```

Checking for copy/paste code in files of the given path.

```
<phpcpd>
   <fileset dir = "${builddir}" id = "filestocpd">
     <include name = "apps/**/*.php" />
     <include name = "lib/de/**/*.php" />
     <include name = "lib/task/**/*.php" />
     <include name = "lib/services/**/*.php" />
     <include name = "lib/form/**/*.php" />
     <include name = "lib/model/**/*.php" />
   </fileset>
   <formatter type = "pmd" outfile = "reports/pmd-cpd.xml"/>
 </phpcpd>
```

C.47.2. Supported Nested Tags

- `fileset`

  This nested tag is required when the `file` attribute is not set.

- `formatter`

  The results of the copy/paste scan can be printed in different formats. Output will always be sent to a file, unless you set the `usefile` attribute to `false`.

Table C.61: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	The output format. Accepts the same values as the `format` attribute (`default`, `pmd`).	n/a	Yes
`useFile`	`Boolean`	Flag that determines whether output should be sent to a file or not.	`true`	No
`outfile`	`String`	Path to write output file to.	n/a	Yes

C.48. PHPLocTask

This task runs phploc [http://github.com/sebastianbergmann/phploc/], a tool for measuring the size of PHP projects. You need an installed version of this tool (installable via PEAR) to use this task.

Table C.62: *Attributes*

Name	Type	Description	Default	Required
`reportType`	`String`	The type of the report. Available types are cli|csv|txt|xml.	cli	No
`reportName`	`String`	The name of the report type without a file extension.	phploc-report	No
`reportDirectory`	`String`	The directory to write the report file to.	false	Yes, when report type csv, txt or xml is defined.
`countTests`	`Boolean`	Flag to count the projects tests or not.	false	No
`file`	`String`	The name of the file to check.	n/a	Yes, when no nested fileset is defined.
`suffixes`	`String`	A comma-separated list of file suffixes to check.	php	No

C.48.1. Examples

```
<target name = "-measure-and-log" description = "Measures and logs the size of the project" hidde
                  <tstamp>
                    <format property = "check.date.time" pattern = "%Y%m%d-%H%M%S" locale = "en_U
                  </tstamp>
                  <phploc reportType = "txt" reportName = "${check.date.time}-report"
                          reportDirectory = "phploc-reports">
```

```
                    <fileset dir = ".">
                      <include name = "**/*.php" />
                      <include name = "*.php" />
                    </fileset>
                  </phploc>
                </target>
```

Checks the size of the project living in ${project.basedir} and writes the result as a txt report to ${project.basedir}/phploc-reports/${check.date.time}-report.txt.

```
<target name = "project-size-and-tests" description = "Measures the size of the project and cou
                  <phploc countTests = "true">
                    <fileset dir = ".">
                      <include name = "**/*.php" />
                      <include name = "*.php" />
                    </fileset>
                  </phploc>
                </target>
```

Checks the size of the project living in ${project.basedir}, counts the project tests and writes/logs the result to the CLI.

C.48.2. Supported Nested Tags

- `fileset`

C.49. PHPMDTask

This task runs phpmd [http://phpmd.org], a Project Mess Detector (PMD) for PHP Code. You need an installed version of this software to use this task.

Table C.63: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Path to source file or path	n/a	Only when there are *no* nested `fileset` elements
`rulesets`	`String`	Sets the rulesets used for analyzing the source code	codesize, unused-code	No
`minimumPriority`	`Integer`	The minimum priority for rules to load.	5	No
`allowedFileExtensions`	`String`	Comma-separated list of valid file extensions (without dot) for analyzed files.	php	No
`ignorePatterns`	`String`	Comma-separated list of directory patterns to ignore.	`.git`, `.svn`, CVS, `.bzr`, `.hg`	No
`format`	`String`	The format for the report when no nested for-matter is used.	text	No

C.49.1. Example

```
<phpmd file = "path/to/source.php"/>
```

Checking syntax of one particular source file. Sending Text-Report to STDOUT.

```
<phpmd file = "path/to/source">
   <formatter type = "html" outfile = "reports/pmd.html"/>
 </phpmd>
```

Checking syntax of source files in the given path.

```
<phpmd>
   <fileset dir = "${builddir}">
     <include name = "apps/**/*.php" />
     <include name = "lib/de/**/*.php" />
   </fileset>
   <formatter type = "xml" outfile = "reports/pmd.xml"/>
 </phpmd>
```

Checking syntax of source files in the fileset pathes.

C.49.2. Supported Nested Tags

- `fileset`

  This nested tag is required when the `file` attribute is not set.

- `formatter`

  The results of the analysis can be printed in different formats. Output will always be sent to STDOUT, unless you set the `usefile` attribute to `true` and set an filename in the `outfile` attribute.

Table C.64: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	The output format. Accepts the same values as the `format` attribute (`xml`, `html`, `text`).	n/a	Yes
`usefile`	`Boolean`	Boolean that determines whether output should be sent to a file.	`true`	No
`outfile`	`String`	Path to write output file to.	n/a	Yes

C.50. PhpDependTask

This task runs PHP_Depend [http://pdepend.org], a software analyzer and metric tool for PHP Code. You need an installed version of this software to use this task.

Table C.65: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Path to source file or path	n/a	Only when there are *no* nested `file-set` elements

Name	Type	Description	Default	Required
`configFile`	`String`	Path to PHP_Depend configuration file	n/a	No
`allowedFileExtensions`	`String`	Comma-separated list of valid file extensions (without dot) for analyzed files.	`php,php5`	No
`excludeDirectories`	`String`	Comma-separated list of directory patterns to ignore.	`.git`, `.svn`, `CVS`	No
`excludePackages`	`String`	Comma-separated list of packages to ignore.	n/a	No
`withoutAnnotations`	`Boolean`	Should the parse ignore doc comment annotations?	`false`	No
`supportBadDocumentation`	`Boolean`	Should PHP_Depend treat `+global` as a regular project package?	`false`	No
`debug`	`Boolean`	Enable debug output?	`false`	No
`haltonerror`	`Boolean`	Stop the build process if errors occurred during the run.	`false`	No

C.50.1. Example

```
<phpdepend file = "path/to/source">
   <logger type = "phpunit-xml" outfile = "reports/metrics.xml"/>
 </phpdepend>
```

Running code analysis for source files in the given path.

```
<phpdepend>
   <fileset dir = "${builddir}">
     <include name = "apps/**/*.php" />
     <include name = "lib/de/**/*.php" />
   </fileset>
   <logger type = "jdepend-xml" outfile = "reports/jdepend.xml"/>
   <analyzer type = "coderank-mode" value = "method"/>
 </phpdepend>
```

Running code analysis for source files in the fileset pathes with CodeRank strategy `method`.

C.50.2. Supported Nested Tags

- `fileset`

  This nested tag is required when the `file` attribute is not set.

- `logger`

  The results of the analysis can be parsed by differed loggers. At least one logger is required. Output will always be sent to a file.

Table C.66: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	The name of the logger. Valid loggers are: `jdepend-chart`, `jdepend-xml`,	n/a	Yes

Name	Type	Description	Default	Required
		`overview-pyramid`, `phpunit-xml` and `summary-xml`.		
`outfile`	`String`	Path to write output file to.	n/a	Yes

- `analyzer`

  Some additional analyzers can be added to the runner.

Table C.67: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	The name of the analyzer. Valid analyzers are: `coderank-mode`.	n/a	Yes
`value`	`String`	The value for the analyzer.	n/a	Yes

C.51. PhpDocumentorTask

This task is now deprecated in favor of the Section C.52, “DocBloxTask”.

This task runs phpDocumentor [http://www.phpdoc.org/], an auto-documentation tool for PHP similar to Javadoc.

Table C.68: *Attributes*

Name	Type	Description	Default	Required
`title`	`String`	Title for browser window / package index.	n/a	No
`destdir`	`String`	Destination directory for output files.	n/a	Yes
`target`	`String`	Alias of `destdir` ("target" is config param used by PhpDocumentor)	n/a	
`output`	`String`	Output format (such as `HTML:Smarty:PHP`).	n/a	Yes
`sourcecode`	`Boolean`	Generate syntax-highlighted sourcecode file for each file parsed?	`false`	No
`examplesdir`	`String`	Path to directory in which to look for exampl documentation.	n/a	No
`parseprivate`	`Boolean`	Parse @internal and elements marked pri-vate.	`false`	No
`javadocdesc`	`Boolean`	JavaDoc-compliant description parsing. Use on/off, default off (more flexibility)	`false`	No
`quiet`	`Boolean`	Suppress output to STDOUT.	`false`	No
`packageoutput`	`String`	Output documentation only for selected pack-ages. Use a comma-delimited list	n/a	No
`ignoretags`	`String`	Comma-separated list of tags to ignore (@package, @subpackage, @access and @ignore may not be ignored).	n/a	No
`defaultpacka-gename`	`String`	name to use for the default package. If not specified, uses 'default'	n/a	No

Name	Type	Description	Default	Required
defaultcategoryname	String	name to use for the default category. If not specified, uses 'default'	n/a	No
pear	Boolean	Treat parse dirs as PEAR repository? (package is directory, _members are @access private)	false	No
templatebase	String	Set base dirctory of all templates for this parse.	n/a	No
undocumentedelements	Boolean	Control whether or not warnings will be shown for undocumented elements. Useful for identifying classes and methods that haven't yet been documented.	false	No
customtags	Boolean	Custom tags, will be recognized and put in tags[] instead of unknowntags[].	false	No
ignore	String	List of files to ignore, separated by ','.	n/a	No

C.51.1. Example

```
<phpdoc title = "API Documentation"
  destdir = "apidocs"
  sourcecode = "false"
  output = "HTML:Smarty:PHP">
   <fileset dir = "./classes">
      <include name = "**/*.php" />
   </fileset>
   <projdocfileset dir = ".">
      <include name = "README" />
      <include name = "INSTALL" />
      <include name = "CHANGELOG" />
   </projdocfileset>
</phpdoc>
```

C.51.2. Supported Nested Tags

- `fileset` - Files that should be included for parsing
- `projdocfileset` - Files that should be treated as README/INSTALL/CHANGELOG files

C.52. DocBloxTask

This task runs phpDocumentor 2 [http://www.phpdoc.org/], a PHP 5.3-compatible API documentation tool. This project is the result of the merge of the phpDocumentor and DocBlox projects.

Table C.69: *Attributes*

Name	Type	Description	Default	Required
title	String	Title of the project.	n/a	No
destdir	String	Destination directory for output files.	n/a	Yes
quiet	Boolean	DEPRECATED	n/a	No

Name	Type	Description	Default	Required
`template`	`String`	Name of the documentation template to use.	`respon-sive-twig`	No

C.52.1. Example

```
<phpdoc2 title = "API Documentation"
  destdir = "apidocs"
  template = "responsive-twig">
   <fileset dir = "./classes">
      <include name = "**/*.php" />
   </fileset>
</phpdoc2>
```

C.52.2. Supported Nested Tags

- fileset - Files that should be included for parsing

C.53. PhpDocumentorExternalTask

This task is now deprecated in favor of the Section C.52, “DocBloxTask”.

This is the same as the Section C.51, “PhpDocumentorTask” but uses the command line application. Use this as a fallback in case you're running into troubles when using the phpDocumentor-library with the PhpDocumentorTask directly, e.g. when you're using Smarty and have Smarty in your library path too.

This task supports everything the PhpDocumentorTask supports, differences are documented below.

Table C.70: *Parameters*

Name	Type	Description	Default	Required
`programpath`	`String`	Path to the phpdoc executable (relative or absolute).	n/a	No
`sourcepath`	`String`	A directory to scan for parsable files. Supports multiple directories separated with a comma.	n/a	Yes, if no `<fileset>` is given

Table C.71: *Unsupported Parameters*

Name	Description
`configdir`	Currently not supported. The attribute will be ignored and a warning messag will be generated. The build continues (to ease when changing an existing phpdoc task) however this may have unexpected side effects.

C.53.1. Example

```
<phpdocext title = "API Documentation"
  programpath = "/usr/bin/phpdoc"
  destdir = "apidocs"
  sourcecode = "false"
```

```
  output = "HTML:Smarty:PHP">
   <fileset dir = "./classes">
      <include name = "**/*.php" />
   </fileset>
   <projdocfileset dir = ".">
      <include name = "README" />
      <include name = "INSTALL" />
      <include name = "CHANGELOG" />
   </projdocfileset>
</phpdocext>
```

C.54. PhpLintTask

The `PhpLintTask` checks syntax (lint) on one or more PHP source code files.

Table C.72: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Path to source file	n/a	No
`haltonfailure`	`Boolean`	Stop the build process if the linting process encounters an error.	`false`	No
`errorproperty`	`String`	The name of a property that will be set to contain the error string (if any).	n/a	No
`interpreter`	`String`	Path to alternative PHP interpreter	Defaults to the `${php.command}` property which is the interpreter used to execute phing itself.	No
`cachefile`	`String`	If set, enables writing of last-modified times to `cachefile`, to speed up processing of files that rarely change	none	No
`level`	`String`	Control the level at which phplint reports status messages. One of `error`, `warning`, `info`, `verbose`, `debug`.	`debug`	No
`tofile`	`String`	File to write list of 'bad files' to.	n/a	No
`deprecatedAsError`	`Boolean`	Whether to treat deprecated warnings (introduced in PHP 5.3) as errors.	`false`	No

C.54.1. Example

```
<phplint file = "path/to/source.php"/>
```

Checking syntax of one particular source file.

```
<phplint>
```

```
  <fileset dir = "src">
    <include name = "**/*.php"/>
  </fileset>
</phplint>
```

Check syntax of a fileset of source files.

C.54.2. Supported Nested Tags

- `fileset`

C.55. PHPUnitTask

This task runs testcases using the PHPUnit [http://www.phpunit.de/] framework. It is a functional port of the Ant JUnit [http://ant.apache.org/manual/OptionalTasks/junit.html] task.

NB: the identifiers `phpunit2` (PHPUnit2Task) and `phpunit3` (PHPUnit3Task) have been deprecated!

Table C.73: *Attributes*

Name	Type	Description	Default	Required
`printsummary`	`Boolean`	Print one-line statistics for each testcase.	`false`	No
`bootstrap`	`String`	The name of a bootstrap file that is run before executing the tests.	none	No
`codecoverage`	`Boolean`	Gather code coverage information while running tests (requires Xdebug).	`false`	No
`haltonerror`	`Boolean`	Stop the build process if an error occurs during the test run.	`false`	No
`haltonfailure`	`Boolean`	Stop the build process if a test fails (errors are considered failures as well).	`false`	No
`haltonincomplete`	`Boolean`	Stop the build process if any incomplete tests are encountered.	`false`	No
`haltonskipped`	`Boolean`	Stop the build process if any skipped tests are encountered.	`false`	No
`failureproperty`	`String`	Name of property to set (to true) on failure.	n/a	No
`errorproperty`	`String`	Name of property to set (to true) on error.	n/a	No
`incompleteproperty`	`String`	Name of property to set (to true) on incomplete tests.	n/a	No
`skippedproperty`	`String`	Name of property to set (to true) on skipped tests.	n/a	No
`usecustomerrorhandler`	`Boolean`	Use a custom Phing/PHPUnit error handler to process PHP errors.	`true`	No
`processisolation`	`Boolean`	Enable process isolation when executing tests.	`false`	No
`configuration`	`String`	Path to a PHPUnit configuration file (such as `phpunit.xml`). Supported elements are: `bootstrap`, `processIsolation`, `sto-`	n/a	No

Name	Type	Description	Default	Required
		`pOnFailure`, `stopOnError`, `stopOnIncomplete` and `stopOnSkipped`. Values provided **overwrite** other attributes!		

C.55.1. Supported Nested Tags

- `formatter`

  The results of the tests can be printed in different formats. Output will always be sent to a file, unless you set the `usefile` attribute to `false`. The name of the file is predetermined by the formatter and can be changed by the `outfile` attribute.

  There are four predefined formatters - `xml` and `clover` print the test results in the JUnit and Clover XML formats, the other two emit plain text. The `brief` formatter will only print detailed information for testcases that failed, while `plain` emits a short statistics line for all test cases. Custom formatters that implement phing.tasks.ext.phpunit.formatter.PHPUnitResultFormatter can be specified.

  Table C.74: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	Use a predefined formatter (either `xml`, `plain`, `clover`, or `brief`).	n/a	One of these is required.
`classname`	`String`	Name of a custom formatter class.	n/a	
`usefile`	`Boolean`	Boolean that determines whether output should be sent to a file.	`true`	No
`todir`	`String`	Directory to write the file to.	n/a	No
`outfile`	`String`	Filename of the result.	Depends on formatter	No

- `batchtest`

  Define a number of tests based on pattern matching. `batchtest` collects the included files from any number of nested <fileset>s. It then generates a lists of classes that are (in)directly defined by each PHP file.

  Table C.75: *Attributes*

Name	Type	Description	Default	Required
`exclude`	`String`	A list of classes to exclude from the pattern matching. For example, when you have two baseclasses `BaseWebTest` and `BaseMathTest`, which are included a number of testcases (and thus added to the list of testclasses), you can exclude those classes from the list by typing `exclude="BaseWebTest BaseMathTest"`.	n/a	No
`classpath`	`String`	Used to define more paths on which - besides the PHP include_path - to look for the test files.	n/a	No
`name`	`String`	The name that is used to create a testsuite from this batchtest.	Phing Batchtest	No

C.55.2. Example

```
<phpunit>
  <formatter todir = "reports" type = "xml"/>
  <batchtest>
    <fileset dir = "tests">
      <include name = "**/*Test*.php"/>
      <exclude name = "**/Abstract*.php"/>
    </fileset>
  </batchtest>
</phpunit>
```

Runs all matching testcases in the directory `tests`, writing XML results to the directory `reports`.

```
<phpunit codecoverage = "true" haltonfailure = "true" haltonerror = "true">
  <formatter type = "plain" usefile = "false"/>
  <batchtest>
    <fileset dir = "tests">
      <include name = "**/*Test*.php"/>
    </fileset>
  </batchtest>
</phpunit>
```

Runs all matching testcases in the directory `tests`, gathers code coverage information, writing plain text results to the console. The build process is aborted if a test fails.

```
<phpunit bootstrap = "src/autoload.php">
  <formatter type = "plain" usefile = "false"/>
  <batchtest>
    <fileset dir = "tests">
      <include name = "**/*Test*.php"/>
    </fileset>
  </batchtest>
</phpunit>
```

Runs all matching testcases in the directory `tests`, writing plain text results to the console. Additionally, before executing the tests, the bootstrap file `src/autoload.php` is loaded.

Important note: using a mechanism such as an "AllTests.php" file to execute testcases will bypass the Phing hooks used for reporting and counting, and could possibly lead to strange results. Instead, use one of more fileset's to provide a list of testcases to execute.

C.55.3. Supported Nested Tags

- `fileset`

C.56. PHPUnitReport

This task transforms PHPUnit xml reports to HTML using XSLT.

NB: the identifiers `phpunit2report` (PHPUnit2Report) and `phpunit3report` (PHPUnit3Report)have been deprecated!

Table C.76: *Attributes*

Name	Type	Description	Default	Required
`infile`	`String`	The filename of the XML results file to use.	testsuites.xml	No

Name	Type	Description	Default	Required
`format`	`String`	The format of the generated report. Must be `noframes` or `frames`.	noframes	No
`styledir`	`String`	The directory where the stylesheets are located. They must conform to the following conventions: frames format: the stylesheet must be named `phpunit-frames.xsl`. noframes format: the stylesheet must be named `phpunit-noframes.xsl`. If unspecified, the task will look for the stylesheet(s) in the following directories: the PHP include path, the Phing home directory and the PEAR data directory (if applicable).	n/a	No
`todir`	`String`	An existing directory where the files resulting from the transformation should be written to.		Yes
`usesorttable`	`Boolean`	Whether to use the sorttable JavaScript library (see http://www.kryogenix.org/code/browser/sorttable/)	`false`	No

C.56.1. Example

```
<phpunitreport infile = "reports/testsuites.xml"
    format = "frames"
    todir = "reports/tests"
    styledir = "/home/phing/etc"/>
```

Generates a framed report in the directory `reports/tests` using the file `reports/testsuites.xml` as input.

Important note: testclasses that are not explicitly placed in a package (by using a '@package' tag in the class-level DocBlock) are listed under the "default" package.

C.57. rSTTask

Renders rST (reStructuredText) files into different output formats.

This task requires the `python docutils` installed. They contain `rst2html`, `rst2latex`, `rst2man`, `rst2odt`, `rst2s5`, `rst2xml`.

Homepage: https://gitorious.org/phing/rsttask

Table C.77: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	rST input file to render	n/a	Yes (or fileset)
`format`	`String`	Output format: • html • latex • man	html	No

Name	Type	Description	Default	Required
		• odt • s5 • xml		
`destination`	`String`	Path to store the rendered file to. Used as directory if it ends with a `/`.	magically determined from input file	No
`uptodate`	`Boolean`	Only render if the input file is newer than the target file	`false`	No
`toolpath`	`String`	Path to the rst2* tool	determined from `format`	No
`toolparam`	`String`	Additional commandline parameters to the rst2* tool	n/a	No
`mode`	`Integer`	The mode to create directories with.	From umask	No

C.57.1. Features

- renders single files
- render nested filesets
- mappers to generate output file names based on the rst ones
- multiple output formats
- filter chains to e.g. replace variables after rendering
- custom parameters to the rst2* tool
- configurable rst tool path
- uptodate check
- automatically overwrites old files
- automatically creates target directories

C.57.2. Examples

Render a single rST file to HTML

By default, HTML is generated. If no target file is specified, the input file name is taken, and its extension replaced with the correct one for the output format.

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "single">
  <target name = "single" description = "render a single rST file to HTML">
```

```
    <rST file = "path/to/file.rst" />

  </target>
</project>
```

Render a single rST file to any supported format

The `format` attribute determines the output format:

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "single">
  <target name = "single" description = "render a single rST file to S5 HTML">

    <rST file = "path/to/file.rst" format = "s5" />

  </target>
</project>
```

Specifying the output file name

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "single">
  <target name = "single" description = "render a single rST file">

    <rST file = "path/to/file.rst" destination = "path/to/output/file.html" />

  </target>
</project>
```

Rendering multiple files

A nested `fileset` tag may be used to specify multiple files.

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "multiple">
  <target name = "multiple" description = "renders several rST files">

    <rST>
     <fileset dir = ".">
       <include name = "README.rst" />
       <include name = "docs/*.rst" />
     </fileset>
    </rST>

  </target>
</project>
```

Rendering multiple files to another directory

A nested `mapper` may be used to determine the output file names.

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "multiple">
  <target name = "multiple" description = "renders several rST files">

    <rST>
     <fileset dir = ".">
       <include name = "README.rst" />
       <include name = "docs/*.rst" />
     </fileset>
     <mapper type = "glob" from = "*.rst" to = "path/to/my/*.xhtml"/>
    </rST>

  </target>
```

```
</project>
```

Modifying files after rendering

You may have variables in your rST code that can be replaced after rendering, i.e. the version of your software.

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "filterchain">
  <target name = "filterchain" description = "renders several rST files">

    <rST>
     <fileset dir = ".">
       <include name = "README.rst" />
       <include name = "docs/*.rst" />
     </fileset>
     <filterchain>
       <replacetokens begintoken = "##" endtoken = "##">
         <token key = "VERSION" value = "1.23.0" />
       </replacetokens>
     </filterchain>
    </rST>

  </target>
</project>
```

Rendering changed files only

The `uptodate` attribute determines if only those files should be rendered that are newer than their output file.

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "multiple">
  <target name = "multiple" description = "renders several rST files">

    <rST uptodate = "true">
     <fileset dir = ".">
       <include name = "docs/*.rst" />
     </fileset>
    </rST>

  </target>
</project>
```

Specify a custom CSS file

You may pass any additional parameters to the rst conversion tools with the `toolparam` attribute.

```
<?xml version="1.0" encoding="utf-8"?>
<project name = "example" basedir = "." default = "single">
  <target name = "single" description = "render a single rST file to S5 HTML">

    <rST file = "path/to/file.rst" toolparam = "--stylesheet-path=custom.css" />

  </target>
</project>
```

C.57.3. Supported Nested Tags

- `fileset`
- `mapper`

- `filterchain`

C.58. S3PutTask

Uploads an object to Amazon S3. This task requires the PEAR package Services_Amazon_S3 [http://pear.php.net/package/Services_Amazon_S3]

Table C.78: *Attributes*

Name	Type	Description	Default	Required
`key`	`String`	Amazon S3 key	n/a	Yes (or defined before task call as: amazon.key)
`secret`	`String`	Amazon S3 secret	n/a	Yes (or defined before task call as: amazon.secret)
`bucket`	`String`	Bucket to store the object in	n/a	Yes (or defined before task call as: amazon.bucket)
`content`	`String`	Content to store in the object	n/a	Yes (or source or fileset)
`source`	`String`	Where to read content for the object from	n/a	Yes (or content or fileset)
`object`	`String`	Object name	n/a	Yes (unless fileset)
`contentType`	`String`	Content type of the object, set to `auto` if you want to autodetect the content type based on the source file extension	binary/octet-stream	No
`fileNameOnly`	`Boolean`	Whether filenames should contain paths when uploaded to a bucket	false	No

C.58.1. Example

Uploading a file

```
<s3put source = "/path/to/file.txt" object = "file.txt" bucket = "mybucket" key = "AmazonKey" s
```

You can also define "bucket, key, secret" outside of the task call:

```
<property name = "amazon.key" value = "my_key" />
<property name = "amazon.secret" value = "my_secret" />
<property name = "amazon.bucket" value = "mybucket" />

<s3put source = "/path/to/file.txt" object = "file.txt" />
```

You can also specify inline content instead of a file to upload:

```
<property name = "amazon.key" value = "my_key" />
```

```
<property name = "amazon.secret" value = "my_secret" />
<property name = "amazon.bucket" value = "mybucket" />

<s3put content = "Some content here" object = "file.txt" />
```

It also works with filesets

```
<property name = "amazon.key" value = "my_key" />
<property name = "amazon.secret" value = "my_secret" />
<property name = "amazon.bucket" value = "mybucket" />
<s3put>
 <fileset dir = "${project.basedir}">
  <include name = "**/*.jpg" />
 </fileset>
</s3put>
```

C.58.2. Supported Nested Tags

- `fileset`

C.59. S3GetTask

Downloads an object from Amazon S3. This task requires the PEAR package Services_Amazon_S3 [http://pear.php.net/package/Services_Amazon_S3]

Table C.79: *Attributes*

Name	Type	Description	Default	Required
`key`	`String`	Amazon S3 key	n/a	Yes (or defined before task call as: amazon.key)
`secret`	`String`	Amazon S3 secret	n/a	Yes (or defined before task call as: amazon.secret)
`bucket`	`String`	Bucket containing the object	n/a	Yes (or defined before task call as: amazon.bucket)
`object`	`String`	Object name	n/a	Yes
`target`	`String`	Where to store the object after download	n/a	Yes

C.59.1. Example

Downloading an object

```
<s3get object = "file.txt" target = "${project.basedir}" bucket = "mybucket" key = "AmazonKey" se
```

You can also define "bucket, key, secret" outside of the task call:

```
<property name = "amazon.key" value = "my_key" />
<property name = "amazon.secret" value = "my_secret" />
```

```
<property name = "amazon.bucket" value = "mybucket" />

<s3get object = "file.txt" target = "${project.basedir}" />
```

C.60. ScpTask

The `ScpTask` copies files to and from a remote host using scp. This task requires the PHP SSH2 extension [http://pecl.php.net/package/ssh2] to function.

Table C.80: *Attributes*

Name	Type	Description	Default	Required
`host`	`String`	Remote host	none	Yes
`port`	`Integer`	Remote port	22	No
`username`	`String`	Username to use for the connection	none	Yes
`password`	`String`	Password to use for the connection	none	No
`pubkeyfile`	`String`	Public key file (OpenSSH format) to use for the connection	none	No
`privkeyfile`	`String`	Private key file (OpenSSH format) to use for the connection	none	No
`privkey-filepassphrase`	`String`	Private key file passphrase to use for the connection	none	No
`autocreate`	`Boolean`	Whether to autocreate remote directories	`true`	No
`todir`	`String`	Directory to put file(s) in	none	No
`file`	`String`	Filename to use	none	No
`fetch`	`Boolean`	Whether to fetch (instead of copy to) the file	`false`	No
`level`	`String`	Control the level at which the task reports status messages. One of `error`, `warning`, `info`, `verbose`, `debug`.	`verbose`	No

C.60.1. Example

```
<scp username = "john" password = "smith"
host = "webserver" fetch = "true"
todir = "/home/john/backup"
file = "/www/htdocs/test.html" />
```

Fetches a single file from the remote server.

```
<scp username = "john" password = "smith"
host = "webserver"
todir = "/www/htdocs/"
file = "/home/john/dev/test.html" />
```

Copies a single file to the remote server.

```
<scp username = "john" password = "smith"
host = "webserver" todir = "/www/htdocs/project/">
    <fileset dir = "test">
```

```
        <include name = "*.html" />
    </fileset>
</scp>
```

Copies multiple files to the remote server.

C.60.2. Supported Nested Tags

- `fileset`
- `sshconfig`

  Sometimes it is necessary to set specific configuration parameters on the ssh connection when connecting to a remote server. You can set them with the sshconfig nested tag. Set the parameters to specify connection and encryption options. These are the parameters as specified by the $methods parameter of the ssh2_connect function. See ssh2_connect [http://us3.php.net/ssh2_connect] for more information

  sshconfig can also be used as project level parameter with a refid so the same parameters can be re-used across a project easily.

Table C.81: *Attributes*

Name	Type	Description	Default	Required
`kex`	`String`	List of key exchange methods to advertise, comma separated in order of preference.	n/a	No
`hostkey`	`String`	List of hostkey methods to advertise, comen/a separated in order of preference.		No
`client`	`Nested Tag`	Element containing attributes crypt, comp, and mac method preferences for messages sent from client to server. All attributes are optional.	n/a	No
`server`	`Nested Tag`	Element containing attributes crypt, comp, and mac method preferences for messages sent from server to client. All attributes are optional.	n/a	No

C.61. SshTask

The `SshTask` executes commands on a remote host using ssh. This task requires the PHP SSH2 extension [http://pecl.php.net/package/ssh2] to function.

Table C.82: *Attributes*

Name	Type	Description	Default	Required
`host`	`String`	Remote host	none	Yes
`port`	`Integer`	Remote port	22	No
`username`	`String`	Username to use for the connection	none	Yes
`password`	`String`	Password to use for the connection	none	No
`pubkeyfile`	`String`	Public key file (OpenSSH format) to use for the connection	none	No

Name	Type	Description	Default	Required
`privkeyfile`	`String`	Private key file (OpenSSH format) to use for the connection	none	No
`privkey-filepassphrase`	`String`	Private key file passphrase to use for the connection	none	No
`command`	`String`	Command to execute on the remote server	none	Yes
`property`	`String`	The name of the property to capture (any) output of the command	none	No
`display`	`Boolean`	Whether to display the output of the command	`true`	No
`pty`	`String`	The terminal type to open	none	No
`failonerror`	`Boolean`	Decides if a command chain will fail if one of the executed commands failed. Added for backward compatibility. Set to true if you execute more than one command and want the task to fail on any error.	`False`	No

C.61.1. Example

```
<ssh username = "john" password = "smith"
host = "webserver" command = "ls" />
```

Executes a single command on the remote server.

C.61.2. Supported Nested Tags

- `sshconfig`

  Sometimes it is necessary to set specific configuration parameters on the ssh connection when connecting to a remote server. You can set them with the sshconfig nested tag. Set the parameters to specify connection and encryption options. These are the parameters as specified by the $methods parameter of the ssh2_connect function. See ssh2_connect [http://us3.php.net/ssh2_connect] for more information

  sshconfig can also be used as project level parameter with a refid so the same parameters can be re-used across a project easily.

Table C.83: *Attributes*

Name	Type	Description	Default	Required
`kex`	`String`	List of key exchange methods to advertise, comma separated in order of preference.	n/a	No
`hostkey`	`String`	List of hostkey methods to advertise, comen separated in order of preference.	n/a	No
`client`	`Nested Tag`	Element containing attributes crypt, comp, and mac method preferences for messages sent from client to server. All attributes are optional.	n/a	No
`server`	`Nested Tag`	Element containing attributes crypt, comp, and mac method preferences for messages	n/a	No

Name	Type	Description	Default	Required
		sent from server to client. All attributes are optional.		

C.62. SimpleTestTask

This task runs testcases using the SimpleTest [http://www.simpletest.org/] framework.

Table C.84: *Attributes*

Name	Type	Description	Default	Required
`printsummary`	`Boolean`	Print one-line statistics for each testcase.	`false`	No
`haltonerror`	`Boolean`	Stop the build process if an error occurs during the test run.	`false`	No
`haltonfailure`	`Boolean`	Stop the build process if a test fails (errors are considered failures as well).	`false`	No
`failureproperty`	`String`	Name of property to set (to true) on failure.	n/a	No
`errorproperty`	`String`	Name of property to set (to true) on error.	n/a	No
`debug`	`Boolean`	Switch debugging on/off	`false`	No

C.62.1. Example

```
<simpletest>
  <formatter type = "plain"/>
  <fileset dir = "tests">
    <include name = "**/*Test*.php"/>
    <exclude name = "**/Abstract*.php"/>
  </fileset>
</simpletest>
```

Runs all matching testcases in the directory `tests`, writing plain text results to the console.

```
<simpletest haltonfailure = "true" haltonerror = "true">
  <formatter type = "plain" usefile = "false"/>
  <fileset dir = "tests">
    <include name = "**/*Test*.php"/>
  </fileset>
</simpletest>
```

Runs all matching testcases in the directory `tests`, writing plain text results to the console. The build process is aborted if a test fails.

C.62.2. Supported Nested Tags

- `formatter`

  The results of the tests can be printed in different formats. Output will always be sent to a file, unless you set the `usefile` attribute to `false`. The name of the file is predetermined by the formatter and can be changed by the `outfile` attribute.

There are three predefined formatters - one prints the test results in XML format, the other emits plain text. The formatter named brief will only print detailed information for testcases that failed, while plain gives a little statistics line for all test cases. Custom formatters that implement phing.tasks.ext.PHPUnitResultFormatter can be specified.

If you use the XML formatter, it may not include the same output that your tests have written as some characters are illegal in XML documents and will be dropped.

Table C.85: *Attributes*

Name	Type	Description	Default	Required
`type`	`String`	Use a predefined formatter (either `xml`, `plain`, or `summary`).	n/a	One of these is required.
`classname`	`String`	Name of a custom formatter class.	n/a	
`usefile`	`Boolean`	Boolean that determines whether output should be sent to a file.	`true`	No
`todir`	`String`	Directory to write the file to.	n/a	No
`outfile`	`String`	Filename of the result.	Depends on formatter	No

- `fileset`

C.63. SvnCheckoutTask

The `SvnCheckoutTask` checks out a Subversion repository to a local directory.

Table C.86: *Attributes*

Name	Type	Description	Default	Required
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No
`repositoryurl`	`String`	URL of SVN repository	none	Yes
`username`	`String`	A username used to connect to the SVN server	none	No
`password`	`String`	A password used to connect to the SVN server	none	No
`nocache`	`Boolean`	Connection credentials will not be cached	`false`	No
`todir`	`String`	Path to export to	none	Yes
`recursive`	`Boolean`	Recursive behavior	`true`	No
`ignoreexternals`	`Boolean`	Ignore externals definitions	`false`	No
`trustServerCert`	`Boolean`	Trust self-signed certificates	`false`	No

C.63.1. Example

```
<svncheckout
```

```
   svnpath = "/usr/bin/svn"
   username = "anony"
   password = "anony"
   nocache = "true"
   repositoryurl = "svn://localhost/project/trunk/"
   todir = "/home/user/svnwc"/>
```

```
<svncheckout
   svnpath = "C:/Subversion/bin/svn.exe"
   repositoryurl = "svn://localhost/project/trunk/"
   todir = "C:/projects/svnwc"/>
```

C.64. SvnCommitTask

The `SvnCommitTask` commits a local working copy to a SVN repository and sets the specified property (default `svn.committedrevision`) to the revision number of the committed revision.

Table C.87: *Attributes*

Name	Type	Description	Default	Required
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No
`username`	`String`	A username used to connect to the SVN server	none	No
`password`	`String`	A password used to connect to the SVN server	none	No
`nocache`	`Boolean`	Connection credentials will not be cached	`false`	No
`recursive`	`Boolean`	Recurse into all subdirectories	`false`	No
`workingcopy`	`String`	Working copy	none	Yes
`message`	`String`	The commit message	none	Yes
`ignoreexternals`	`Boolean`	Ignore externals definitions	`false`	No
`trustServerCert`	`Boolean`	Trust self-signed certificates	`false`	No
`propertyname`	`String`	Name of property to set to the last committed revision number	svn.committedrevision	No

C.64.1. Example

```
<svncommit
    svnpath = "/usr/bin/svn"
    username = "anony"
    password = "anony"
    nocache = "true"
    workingcopy = "/home/joe/dev/project"
    message = "Updated documentation, fixed typos" />
```

The most basic usage only needs the working copy and the commit message as in

```
<svncommit
    workingcopy = "/home/joe/dev/project"
```

```
    message = "Updated documentation, fixed typos" />
<echo message = "Committed revision: ${svn.committedrevision}"/>
```

C.65. SvnCopyTask

The `SvnCopyTask` duplicates something in a working copy or repository, remembering history.

Table C.88: *Attributes*

Name	Type	Description	Default	Required
`message`	`String`	Log message	n/a	No
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No
`repositoryurl`	`String`	URL of SVN repository	none	Yes
`username`	`String`	A username used to connect to the SVN server	none	No
`password`	`String`	A password used to connect to the SVN server	none	No
`force`	`Boolean`	Force overwrite files if they already exist	`false`	No
`nocache`	`Boolean`	Connection credentials will not be cached	`false`	No
`todir`	`String`	Path to export to	none	Yes
`recursive`	`Boolean`	Recursive behavior	`true`	No
`trustServerCert`	`Boolean`	Trust self-signed certificates	`false`	No

C.65.1. Example

```
<svncopy
   svnpath = "/usr/bin/svn"
   username = "anony"
   password = "anony"
   nocache = "true"
   repositoryurl = "svn://localhost/project/trunk/"
   todir = "svn://localhost/project/tags/0.1"/>
```

C.66. SvnExportTask

The `SvnExportTask` exports a Subversion repository to a local directory.

Table C.89: *Attributes*

Name	Type	Description	Default	Required
`revision`	`String`	Revision to use in export	HEAD	No
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No

Name	Type	Description	Default	Required
repositoryurl	String	URL of SVN repository	none	Yes
username	String	A username used to connect to the SVN server	none	No
password	String	A password used to connect to the SVN server	none	No
nocache	Boolean	Connection credentials will not be cached	false	No
todir	String	Path to export to	none	Yes
recursive	Boolean	Recursive behavior	true	No
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCert	Boolean	Trust self-signed certificates	false	No

C.66.1. Example

```
<svnexport
   svnpath = "/usr/bin/svn"
   username = "anony"
   password = "anony"
   force = "true"
   nocache = "true"
   repositoryurl = "svn://localhost/project/trunk/"
   todir = "/home/user/svnwc"/>
```

```
<svnexport
   svnpath = "C:/Subversion/bin/svn.exe"
   repositoryurl = "svn://localhost/project/trunk/"
   todir = "C:/projects/svnwc"/>
```

C.67. SvnInfoTask

The SvnInfoTask parses the output of the 'svn info --xml' command and extracts one specified element (+ optional sub element) from that output.

Table C.90: *Attributes*

Name	Type	Description	Default	Required
svnpath	String	Path to Subversion binary	/usr/bin/svn	No
workingcopy	String	Working copy directory	none	Yes, or repositoryurl
repositoryurl	String	URL of remote repository	none	Yes, or workingcopy
username	String	A username used to connect to the SVN server	none	No
password	String	A password used to connect to the SVN server	none	No
propertyname	String	Name of property to use	svn.info	No

Name	Type	Description	Default	Required
`element`	`String`	Sets whether to store actual last changed re-vision of the directory/file mentioned	url	No
`subelement`	`String`	Sets whether to force compatibility with older SVN versions (< 1.2)	none	No

C.67.1. Example

```
<svninfo
   svnpath = "/usr/bin/svn"
   workingcopy = "/home/user/svnwc"
   element = "url"
   propertyname = "svn.url"/>
```

```
<svninfo
   repositoryurl = "http://svn.phing.info/"
   element = "commit"
   subelement = "author"
   propertyname = "svn.author"/>
```

C.68. SvnLastRevisionTask

The `SvnLastRevisionTask` stores the number of the last revision of a Subversion workingcopy in a property.

Table C.91: *Attributes*

Name	Type	Description	Default	Required
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No
`workingcopy`	`String`	Working copy directory	none	Yes, or `repositoryurl`
`repositoryurl`	`String`	URL of remote repository	none	Yes, or `workingcopy`
`username`	`String`	A username used to connect to the SVN server	none	No
`password`	`String`	A password used to connect to the SVN server	none	No
`propertyname`	`String`	Name of property to use	svn.lastrevision	No
`lastChanged`	`Boolean`	Sets whether to store actual last changed re-vision of the directory/file mentioned	`false`	No
`forceCompatible`	`Boolean`	*Deprecated:* Sets whether to force compatibility with older SVN versions (< 1.2)	`false`	No

C.68.1. Example

```
<svnlastrevision
   svnpath = "/usr/bin/svn"
   workingcopy = "/home/user/svnwc"
```

```
   propertyname = "svn.lastrevision"/>
```

```
<svnlastrevision
   svnpath = "C:/Subversion/bin/svn.exe"
   workingcopy = "C:/projects/svnwc"
   propertyname = "svn.lastrevision"/>
```

```
<svnlastrevision
   svnpath = "C:/Subversion/bin/svn.exe"
   repositoryurl = "http://svn.phing.info/"
   propertyname = "svn.lastrevision"/>
```

C.69. SvnListTask

The `SvnListTask` stores the output of a svn list command on a workingcopy or repositoryurl in a property. The result will be stored in an array, one string that is separated by ' | ' (in words: space pipe space) for easy parsing.

Table C.92: *Attributes*

Name	Type	Description	Default	Required
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No
`workingcopy`	`String`	Working copy directory	none	One of the two
`repositoryurl`	`String`	URL of remote repository	none	
`username`	`String`	A username used to connect to the SVN server	none	No
`password`	`String`	A password used to connect to the SVN server	none	No
`propertyname`	`String`	Name of property to use	svn.list	No
`forceCompatible`	`Boolean`	*Deprecated:* Sets whether to force compatibility with older SVN versions (< 1.2)	`true`	No
`limit`	`Integer`	Limits the number of items to get back from the command	n/a	No
`orderDescending`	`Boolean`	Sets whether to reverse the order of the listed items	`false`	No

C.69.1. Example

```
<svnlist svnpath = "/usr/bin/svn" workingcopy = "/home/user/svnwc" propertyname = "svn.list"/>
```

```
<svnlist svnpath = "/usr/bin/svn" repositoryurl = "http://svn.example.com/myrepo/tags" orderDesce
```

The latter example could produce a list of your tags like this:

```
revision | author | date         | item
4028     | tony   | May 19 18:31 | Release_2.9.1.7
4026     | tony   | May 18 14:33 | Release_2.9.1.6
4023     | tony   | May 16 15:53 | Release_2.9.1.5
4018     | tony   | May 13 11:55 | Release_2.9.1.4
4005     | tony   | Apr 27 12:09 | Release_2.9.1.3
```

```
...
```

C.70. SvnLogTask

The `SvnLogTask` stores the output of a svn log command on a workingcopy or repositoryurl in a property. The result will be stored in an array, one string that is separated by ' | ' (in words: space pipe space) for easy parsing.

Table C.93: *Attributes*

Name	Type	Description	Default	Required
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No
`workingcopy`	`String`	Working copy directory	none	One of the two
`repositoryurl`	`String`	URL of remote repository	none	
`username`	`String`	A username used to connect to the SVN server	none	No
`password`	`String`	A password used to connect to the SVN server	none	No
`propertyname`	`String`	Name of property to use	svn.list	No
`forceCompatible`	`Boolean`	*Deprecated:* Sets whether to force compatibility with older SVN versions (< 1.2)	`true`	No
`limit`	`Integer`	Limits the number of items to get back from the command	n/a	No

C.70.1. Example

```
<svnlog svnpath = "/usr/bin/svn" workingcopy = "/home/user/svnwc" propertyname = "svn.log"/>
```

```
<svnlog svnpath = "/usr/bin/svn" repositoryurl = "http://svn.example.com/myrepo/trunk" limit =
```

The latter example could produce a history of the latest revisions in the trunk:

```
4033 | tony  | 2011-05-23T14:21:12.496274Z  | some svn commit comment
4032 | tony  | 2011-05-23T13:24:46.496265Z  | some svn commit comment
4031 | tony  | 2011-05-23T09:23:28.093167Z  | some svn commit comment
...
```

C.71. SvnUpdateTask

The `SvnUpdateTask` updates a local directory.

Table C.94: *Attributes*

Name	Type	Description	Default	Required
`svnpath`	`String`	Path to Subversion binary	/usr/bin/svn	No

Name	Type	Description	Default	Required
username	String	A username used to connect to the SVN server	none	No
password	String	A password used to connect to the SVN server	none	No
nocache	Boolean	Connection credentials will not be cached	false	No
todir	String	Path to the working copy	none	Yes
revision	Integer	Specific revision to update the working copy to	none	No
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCert	Boolean	Trust self-signed certificates	false	No

C.71.1. Example

```
<svnupdate
   svnpath = "/usr/bin/svn"
   username = "anony"
   password = "anony"
   nocache = "true"
   todir = "/home/user/svnwc"/>
```

```
<svnupdate
   svnpath = "C:/Subversion/bin/svn.exe"
   todir = "C:/projects/svnwc"/>
```

C.72. SvnSwitchTask

The SvnSwitchTask changes a local directory from one repository to another.

Table C.95: *Attributes*

Name	Type	Description	Default	Required
svnpath	String	Path to Subversion binary	/usr/bin/svn	No
repositoryurl	String	URL of remote repository	none	Yes
todir	String	Path to the checked out project	none	Yes
username	String	A username used to connect to the SVN server	none	No
password	String	A password used to connect to the SVN server	none	No
nocache	Boolean	Connection credentials will not be cached	false	No
recursive	Boolean	Recursive behavior	true	No
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCert	Boolean	Trust self-signed certificates	false	No

C.72.1. Example

```
<svnswitch
   svnpath = "/usr/bin/svn"
   username = "anony"
   password = "anony"
   nocache = "true"
   repositoryurl = "http://svn.phing.info/tags/2.4.2"
   todir = "/home/user/svnwc"/>
```

```
<svnswitch
   svnpath = "C:/Subversion/bin/svn.exe"
   repositoryurl = "http://svn.phing.info/tags/2.4.2"
   todir = "C:/projects/svnwc"/>
```

C.73. SymfonyConsoleTask

Executes Symfony2 console commands

Table C.96: *Attributes*

Name	Type	Description	Default	Required
command	String	The Symfony Console command to execute	n/a	Yes
console	String	The path to symfony console application	app/console	No

C.73.1. Examples

Simple example

```
<SymfonyConsole command = "cache:clear"/>
```

Complex example

```
<SymfonyConsole command = "cache:warmup">
  <arg name = "env" value = "prod" />
  <arg value = "some/path/or/single/value" quotes = "true">
</SymfonyConsole>
```

C.73.2. Supported Nested Tags

- `arg`

Table C.97: *Attributes*

Name	Type	Description	Default	Required
name	String	the name for this argument, -- will be appended	n/a	No
value	String	the value for the argument	n/a	No

Name	Type	Description	Default	Required
quotes	String	set to true if the value should be enclosed in double quotes	false	No

C.74. SymlinkTask

Creates symlink(s) to a specified file / directory or a collection of files / directories.

Table C.98: *Attributes*

Name	Type	Description	Default	Required
`target`	`String`	What you're trying to symlink from	n/a	Yes (or nested FileSet)
`link`	`String`	Where you'd like the symlink(s)	n/a	Yes
`overwrite`	`Boolean`	Whether to override the symlink if it exists but points to a different location	`false`	No

C.74.1. Example

Single symlink

```
<symlink target = "/path/to/original/file" link = "/where/to/symlink" />
```

Using filesets

```
<symlink link = "/where/to/symlink">
 <fileset dir = "/some/directory">
  <include name = "*" />
 </fileset>
</symlink>
```

In the fileset example, assuming the contents of "/some/directory" were:

- Somedir
- somefile

Then the contents of "/where/to/symlink" would be:

- Somedir -> /some/directory/Somedir
- somefile -> /some/directory/somefile

C.74.2. Supported Nested Tags

- `fileset`

C.75. TarTask

The `TarTask` creates a tarball from a fileset or directory.

Table C.99: *Attributes*

Name	Type	Description	Default	Required
`destfile`	`String`	Tarball filename	none	Yes
`basedir`	`String`	Base directory to tar (if no fileset specified, entire directory contents will be included in tar)	none	No
`compression`	`String`	Type of compression to use (gzip, bzip2, none)	none	No
`includeemptydirs`	`Boolean`	If set to `true`, also empty directories are copied.	`true`	No
`longfile`	`String`	How to handle long files, those with a path >100 chars. Allowable values are: `truncate` - paths are truncated to the maximum length, `fail` - paths greater than the maximim 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	warn	No
`prefix`	`String`	File path prefix to use when adding files to archive	none	No

Note

files are not replaced if they are already present in the archive.

Note

using basedir and fileset simultaneously can result in strange contents in the archive.

C.75.1. Example

```
<tar destfile = "phing.tar">
 <fileset dir = ".">
  <include name = "**/**" />
 </fileset>
</tar>
```

The above example uses a fileset to determine which files to include in the archive.

```
<tar destfile = "phing.tar.gz" basedir = "." compression = "gzip"/>
```

The second example uses the basedir attribute to include the contents of that directory (including sub-directories) in the archive, compressing the archive using gzip.

C.75.2. Supported Nested Tags

- `fileset`

C.76. UntarTask

The `UntarTask` unpacks one or more tar archives.

Table C.100: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Archive filename	n/a	No
`todir`	`String`	Directory to unpack the archive(s) to	none	Yes
`removepath`	`String`	Path to remove from files in the archive(s)	none	No
`forceExtract`	`Boolean`	When set to false, only extract files if the destination does not exist yet or is older than the archive. When set to true, always extract files.	`false`	No

C.76.1. Example

```
<untar file = "testtar.tar.gz" todir = "dest">
  <fileset dir = ".">
    <include name = "*.tar.gz"/>
    <include name = "*.tar"/>
  </fileset>
</untar>
```

C.76.2. Supported Nested Tags

- `fileset`

C.77. UnzipTask

The `UnzipTask` unpacks one or more ZIP archives.

Table C.101: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	Archive filename	n/a	No
`todir`	`String`	Directory to unpack the archive(s) to	none	Yes
`forceExtract`	`Boolean`	When set to false, only extract files if the destination does not exist yet or is older than the archive. When set to true, always extract files.	`false`	No

C.77.1. Example

```
<unzip file = "testzip.zip" todir = "dest">
  <fileset dir = ".">
    <include name = "*.zip"/>
  </fileset>
```

```
</unzip>
```

C.77.2. Supported Nested Tags

- `fileset`

C.78. VersionTask

The `VersionTask` increments a three-part version number from a given file and writes it back to the file. The resulting version number is also published under supplied property.

The version number in the text file is expected in the format of Major.Minor.Bugfix (e.g. 1.3.2).

Table C.102: *Attributes*

Name	Type	Description	Default	Required
`releasetype`	`String`	Specifies desired version release (Major, Minor or Bugfix)	n/a	Yes
`file`	`String`	File containing three-part version number to increment	n/a	Yes
`property`	`String`	Property which contains the resulting version number	n/a	Yes

C.78.1. Example

```
<version releasetype = "Major" file = "version.txt" property = "version.number"/>
```

C.79. WikiPublishTask

This task can publish Wiki document via Wiki WebAPI. It supports only MediaWiki [http://www.mediawiki.org/] engine for now.

cURL [http://www.php.net/manual/en/book.curl.php] extension is required.

Table C.103: *Attributes*

Name	Type	Description	Default	Required
`apiUrl`	`String`	Wiki API URL (eg. http://localhost/wiki/api.php)	n/a	Yes
`apiUser`	`String`	Wiki API user name	n/a	No
`apiPassword`	`String`	Wiki API user password	n/a	No
`id`	`Integer`	ID of page that will be changed	n/a	One of these attributes is required.

Name	Type	Description	Default	Required
`title`	`String`	Title of page that will be changes. Can also be used as page identifier	n/a	
`content`	`String`	Content of published page	n/a	No
`mode`	`String`	Edit mode (overwrite, prepend, append)	append	No

C.79.1. Example

```
<wikipublish
                apiUrl = "http://localhost/wiki/api.php"
                apiUser = "testUser"
                apiPassword = "testPassword"
                title = "Some Page"
                content = "Some content"
                mode = "prepend"/>
```

C.80. XmlLintTask

The `XmlLintTask` checks syntax (lint) one or more XML files against an XML Schema Definition.

`Note:` This assumes that the DOM extension is loaded in PHP5 since this is used to drive the validation process.

Table C.104: *Attributes*

Name	Type	Description	Default	Required
`schema`	`String`	Path to XSD file	n/a	Yes
`file`	`String`	Path to XML file	n/a	No
`haltonfailure`	`Boolean`	Stops the build when validation fails	`true`	No
`useRNG`	`Boolean`	Set to Yes if the Schema is in the n Relax NG format	`false`	No

C.80.1. Examples

```
<xmllint schema = "schema.xsd" file = "config.xml"/>
```

Validate one XML file against one XSD file.

```
<xmllint schema = "schema.xsd">
  <fileset dir = ".">
    <include name = "**/config.xml"/>
  </fileset>
</xmllint>
```

Validate more XML files against one XSD file.

```
<fileset dir = "./sources" id = "sources">
  <include name = "main.xml"/>
```

```
  <include name = "chapter*.xml"/>
  <include name = "appendix*.xml"/>
</fileset>
<property name = "docbook.relaxng" value = "/usr/share/xml/docbook/schema/rng/5.0/docbookxi.rng

<xmllint schema = "${docbook.relaxng}" useRNG = "yes">
  <fileset refid = "sources" />
</xmllint>
```

Validate a set of DocBook files against the DocBook RNG grammar

C.80.2. Supported Nested Tags

- `fileset`

C.81. XmlPropertyTask

Loads property values from a well-formed xml file. There are no other restrictions than "well-formed".

Table C.105: *Attributes*

Name	Type	Description	Default	Required
`file`	`String`	The XML file to parse.	n/a	Yes
`prefix`	`String`	The prefix to prepend to each property	n/a	No
`keepRoot`	`Boolean`	Keep the xml root tag as the first value in the property name.	`true`	No
`collapseAttributes`	`Boolean`	Treat attributes as nested elements.	`false`	No
`delimiter`	`String`	Delimiter for splitting multiple values.	,	No
`required`	`Boolean`	If this is set to `true` then a build exception will be raised if the file cannot be found otherwise only a warning will be logged.	`false`	No

C.81.1. Example

Consider the following XML file:

```
<root-tag myattr = "true">
    <inner-tag someattr = "val">Text</inner-tag>
    <a2><a3><a4>false</a4></a3></a2>
</root-tag>
```

Used with the following entry (**default**):

```
<xmlproperty file = "somefile.xml"/>
```

results in the following properties:

```
root-tag(myattr)=true
root-tag.inner-tag=Text
```

```
root-tag.inner-tag(someattr)=val
root-tag.a2.a3.a4=false
```

Used with the following entry (`collapseAttributes=true`):

```
<xmlproperty file = "somefile.xml" collapseAttributes = "true"/>
```

results in the following properties:

```
root-tag.myattr=true
root-tag.inner-tag=Text
root-tag.inner-tag.someatt=val
root-tag.a2.a3.a4=false
```

Used with the following entry (`keepRoot=false`):

```
<xmlproperty file = "somefile.xml" keepRoot = "false"/>
```

results in the following properties:

```
inner-tag=Text
inner-tag(someattr)=val
a2.a3.a4=false
```

C.82. ZendCodeAnalyzerTask

The `ZendCodeAnalyzerTask` analyze PHP source files using the Zend Code Analyzer tool that ships with all versions of Zend Studio.

Table C.106: *Attributes*

Name	Type	Description	Default	Required
`analyzerPath`	`String`	Path to Zend Code Analyzer binary	n/a	Yes
`file`	`String`	Path to PHP source file	n/a	No
`disable`	`String`	Disable warnings separated by comma	n/a	No
`enable`	`String`	Enable warnings separated by comma	n/a	No
`haltonwarning`	`Boolean`	Stop the build process if warnings occurred during the run.	`false`	No

C.82.1. Example

```
<zendcodeanalyzer
  analyzerPath = "/usr/local/Zend/ZendStudioClient-5.1.0/bin/ZendCodeAnalyzer"
  file = "SomeClass.php"/>
```

Analyze one PHP source file with all default warnings enabled.

```
<zendcodeanalyzer
  analyzerPath = "/usr/local/Zend/ZendStudioClient-5.1.0/bin/ZendCodeAnalyzer"
  disable = "var-ref-notmodified,if-if-else">
  <fileset dir = ".">
```

```
    <include name = "**/*.php"/>
  </fileset>
</zendcodeanalyzer>
```

Analyze a set of PHP source files and disable a few warnings.

NOTE: the `analyze` tag has been deprecated as of Phing 2.4.

C.82.2. Supported Nested Tags

- `fileset`

C.83. ZendGuardEncodeTask

The `ZendGuardEncodeTask` is a wrapper for ZendGuard zendenc executable. It pre-compiles the PHP code which improves speed and can prevent unauthorized code modification. Additionally it allows signing or licensing the code so it can only be used with a valid license.

For more information about ZendGuard encode parameters see the ZendGuard documentation [http://static.zend.com/topics/Zend-Guard-User-Guidev5x.pdf].

Table C.107: *Attributes*

Name	Type	Description	Default	Required
`zendEncoderPath`	`String`	Path to zendenc or zendenc5 binary.	n/a	Yes
`deleteSource`	`Boolean`	Whether to delete the original file and replace with encoded.	`true`	No
`renameSourceExt`	`String`	If defined the original file will be copied to originalfile.renameSourceExt before encoding. This property overrides the deleteSource property.	n/a	No
`shortTags`	`Boolean`	Turns on/off support for PHP short tags (<?). True to enable support.	`true`	No
`aspTags`	`Boolean`	Turns on/off support for ASP tags (<%). True to enable support.	`false`	No
`noHeader`	`Boolean`	Disables the PHP-compatible header that is added to the top of every encoded file by default and is displayed if the Zend Optimizer is not properly installed.	`false`	No
`useCrypto`	`Boolean`	Enables cryptography support.	`false`	No
`encodedOnly`	`Boolean`	If enabled the encoded files will only work with other encoded files (I.e. encoded and not-encoded files cannot be used together).	`false`	No
`forceEncode`	`Boolean`	Allow encoding previously encoded files. Not recommended.	`false`	No
`expires`	`String`	Make an encoded file to expire on the given data. Date is in yyyy-mm-dd format.	n/a	No

Name	Type	Description	Default	Required
obfuscationLevel	Integer	Level of obfuscation. Defaults to 0 (no obfuscation).	0	No
optMask	Integer	Optimization mask. Integer representing a bitmask.	n/a	No
privateKeyPath	String	Path to the company private key. This is required when either signProduct or licenseProduct is enabled.	n/a	No
licenseProduct	Boolean	Enabled product licensing. The encoded files won't work without a valid license. If enabled privateKeyPath property also needs to be defined.	false	No
signProduct	Boolean	Enabled product signing. If signing is enabled the files will be encoded with license support. However valid license won't be required for the files to work. If enabled privatKeyPath property also needs to be defined.	false	No
productName	String	Name of the product. This must match the product name in the license and is required when either licenseProduct or signProduct is enabled.	n/a	No
prologFile	String	Path to a file containing a text that will be prepended to each encoded file and displayed in case the Zend Optimizer is not installed.	n/a	No

C.83.1. Example

```
<zendguardencode
 shortTags = "false"
 productName = "Phing"
 privateKeyPath = "/var/data/phing.key"
 licenseProduct = "true"
 zendEncoderPath = "/usr/local/Zend/ZendGuard-5_0_1/bin/zendenc5"
 >

 <fileset dir = "src">
   <include name = "**/*.php" />
   <exclude name = "cache/**" />
   <exclude name = "plugins/**" />
 </fileset>

 <fileset dir = "src">
   <include name = "plugins/cs*/**/*.php" />
   <include name = "plugins/cs*/*.php" />
 </fileset>
</zendguardencode>
```

Encode all php files in the current directory and subdirectories. Exlude everything in `cache/` and `plugins/` but include everything in plugins that starts with cs.

C.83.2. Supported Nested Tags

- `fileset`

C.84. ZendGuardLicenseTask

The `ZendGuardLicenseTask` is a wrapper for ZendGuard `zendenc_sign` executable. It generates ZendGuard license either from a license template file or from the defined properties.

For more information about ZendGuard sign parameters see the ZendGuard documentation [http://static.zend.com/topics/Zend-Guard-User-Guidev5x.pdf].

Table C.108: *Attributes*

Name	Type	Description	Default	Required
`zendsignPath`	`String`	Path to zendenc_sign binary.	n/a	Yes
`privateKeyPath`	`String`	Path to the company private key.	n/a	Yes
`outputFile`	`String`	Path where should the license be generated.	n/a	Yes
`licenseTemplate`	`String`	Path to a license template file. If defined all other licensing properties will be ignored (even if they are otherwise required).	n/a	No
`productName`	`String`	Name of the product. This has to match the product name that was used to encode the files (see ZendGuardEncodeTask).	n/a	Yes
`registeredTo`	`String`	Name to which the product will be registered to.	n/a	Yes
`expires`	Mixed	This allows to define when the license will expire. The license can be issued so it either never expires or expires at a specified data. Use: `'Never'`, 0 or false to set expiry data to Never. Date in yyyy-mm-dd format to set the expiry date to a specific date. Relative format supported by strtotime function (e.g. '+6 months' to generate a license that will expire in half a year).	n/a	Yes
`ipRange`	`String`	Limits the use of the license to IP addresses that fall within specification. Supports wildcards for any of the IP place holders, as well as the two types of the net masks (e.g. 10.1.0.0/16 or 10.1.0.0./255.255.0.0).	n/a	No
`hardwareLocked`	`Boolean`	Option that indicates if the license will be locked to a specific machine using the Zend Host ID code(s). If set to true the Host-ID property is required.	`false`	No
`hostID`	`String`	Coded string (Zend Host ID) used to lock the license to a specific hardware. The Zend Host Id obtained from the machine where the encoded files and license are to be installed. Can be obtained by using the zendid utility. This is REQUIRED if the Hardware-Locked property is set to true. You can define multiple Host IDs separated by semicolon.	n/a	No
`userDefinedValues`	`String`	Optional user defined values in format key=value. Multiple key-value pairs can be defined and separated by semicolon. These values then will be part of the license and can	n/a	No

Name	Type	Description	Default	Required
		be obtained using the zend guard api (provided by Zend Optimizer). These values CANNOT be modified after the license is generated. Their modification would invalidate the license. Example: Drink=Tea;Material=Wood		
`xUserDefinedValues`	`String`	Optional user defined values in format key=value. Multiple key-value pairs can be defined and separated by semicolon. These values then will be part of the license and can be obtained using the zend guard api (provided by Zend Optimizer). These values CAN be modified after the license is generated. Their modification won't invalidate the license. Example: Drink=Tea;Material=Wood	n/a	No

C.84.1. Examples

```
<zendguardlicense
            privateKeyPath = "/var/data/phing.key"
            zendsignPath = "/usr/local/Zend/ZendGuard-5_0_1/bin/zendenc_sign"
            outputFile = "./data/license/license.zl"
            productName = "Phing"
            registeredTo = "YourCustomerName"
            hardwareLocked = "true"
            expires = "+6 months"
            HostID = "H:MFM43-Q9CXC-B9EDX-GWYSU;H:MFM43-Q9CXC-B9EDX-GWYTY"
            ipRange = "10.1.*.*"
            userDefinedValues = "Drink=Tea;Material=Wood"
            xUserDefinedValues = "Drink=Tea;Material=Wood"
/>
```

Creates a license using the given properties.

```
<zendguardlicense
            privateKeyPath = "/var/data/phing.key"
            zendsignPath = "/usr/local/Zend/ZendGuard-5_0_1/bin/zendenc_sign"
            outputFile = "./data/license/license.zl"
            licenseTemplate = "./data/license/license.template.zl"
/>
```

Creates a license using a license template file.

C.85. ZipTask

The `ZipTask` creates a .zip archive from a fileset or directory.

Table C.109: *Attributes*

Name	Type	Description	Default	Required
`destfile`	`String`	.ZIP filename	n/a	Yes
`basedir`	`String`	Base directory to zip (if no fileset specified, entire directory contents will be included in the archive)	none	No

Name	Type	Description	Default	Required
`prefix`	`String`	File path prefix to use when adding files to zip	none	No
`includeempty-dirs`	`Boolean`	If set to `true`, also empty directories are copied.	`true`	No

Important note: using basedir and fileset simultaneously can result in strange contents in the archive.

C.85.1. Example

```
<zip destfile = "phing.zip">
 <fileset dir = ".">
  <include name = "**/**" />
 </fileset>
</zip>
```

The above example uses a fileset to determine which files to include in the archive.

```
<zip destfile = "phing.zip" basedir = "."/>
```

The second example uses the basedir attribute to include the contents of that directory (including sub-directories) in the archive.

C.85.2. Supported Nested Tags

- `fileset`

Appendix D. Core Types

This appendix contains a reference of the system `data types` contained in Phing.

D.1. Excludes

Specifies a set of files, classes or methods to be excluded from processing.

This element has no attributes, only nested tags

D.1.1. Nested tags

- File
- Class
- Method

Table D.1: *Common attributes for all `File, Class, Method` tags*

Name	Type	Description	Default	Required
`name`	`String`	The name of the class, method or file. This may also be specified as a pattern.	n/a	Yes

D.1.2. Usage Examples

```
<coverage-threshold
    perProject = "50"
    perClass = "60"
    perMethod = "70"/>
    <excludes>
        <file>**/*Processor.php</file>
        <class>Model_Filter_Windows</class>
        <method>Model_System::execute()</method>
    </excludes>
```

D.2. FileList

FileLists offer a way to represent a specific list of files. Unlike FileSets, FileLists may contain files that do not exist on the filesystem. Also, FileLists can represent files in a specific order -- whereas FileSets represent files in whichever order they are returned by the filesystem.

Table D.2: *Attributes for the <`fileset`> tag*

Name	Type	Description	Default	Required
`dir`	`String`	The directory, to which the paths given in `files` or `listfile` are relative.	n/a	Yes
`files`	`String`	Comma or space-separated list of files.	n/a	Yes (or `list-file`)

Name	Type	Description	Default	Required
`listfile`	`String`	A text file with one filename per line.	n/a	Yes (or `files`)

D.2.1. Usage Examples

```
<filelist dir = "/etc" files = "httpd/conf/httpd.conf,php.ini"/>
```

Or you can use a `listfile`, which is expected to contain one filename per line:

```
<filelist dir = "conf/" listfile = "ini_files.txt"/>
```

This will grab each file as listed in `ini_files.txt`. This can be useful if one task compiles a list of files to process and another task needs to read in that list and perform some action to those files.

D.3. FileSet

FileSets offer an easy and straightforward way to include files. The tag supports Selectors [#app.selectors] and PatternSets [#PatternSet]. Additionally, you can include/exclude files in/from a fileset using the `<include>`/`<exclude>` tags. In patterns, one asterisk (`*`) maps to a part of a file/directory name within a directory level. Two asterisks (`**`) may include above the "border" of the directory separator.

Table D.3: *Attributes for the `<fileset>` tag*

Name	Type	Description	Default	Required
`dir`	`String`	The directory, the paths given in `include`/`exclude` are relative to.	n/a	Yes
`defaultexcludes`	`Boolean`	Whether default exclusions should be used or not. Default excludes are: `*~`, `#*#`, `.#*`, `%*%`, CVS, `CVS/**`, `.cvsignore`, SCCS, `SCCS/**`, `vssver.scc`, `.svn`, `.svn/**`, `._*`, `.DS_Store`, `.darcs`, `.darcs/**`	`true`	No
`casesensitive`	`Boolean`	The case sensitivity of the file system.	`true`	No
`expandsymboliclinks`	`Boolean`	Whether to expand/dereference (follow) symbolic links - set to 'true' to emulate old Phing behavior.	`false`	No
`includes`	`String`	Comma- or space-separated list of patterns of files that must be included; all files are included when omitted.	n/a	No
`includesfile`	`String`	The name of a file; each line of this file is taken to be an include pattern.	n/a	No
`excludes`	`String`	comma- or space-separated list of patterns of files that must be excluded; no files (except default excludes) are excluded when omitted.	n/a	No
`excludesfile`	`String`	The name of a file; each line of this file is taken to be an exclude pattern.	n/a	No

D.3.1. Using wildcards

- `test*.xml` will include `test_42.xml`, but it will not include `test/some.xml`.
- `test**.xml` fits to `test_42.xml` as well as to `test/bla.xml`, for example.
- `**/*.ent.xml` fits to all files that end with `ent.xml` in all subdirectories of the directory specified with the `dir` attribute of the `<fileset>` tag. However, it will not include any files that are directly in the base directory of the file set.

D.3.2. Usage Examples

```
<fileset dir = "/etc" >
  <include name = "httpd/**" />
  <include name = "php.ini" />
</fileset>

<fileset dir = "/etc" >
  <patternset>
    <include name = "**/*.php"/>
    <exclude name = "**/*Test*"/>
  </patternset>
</fileset>
```

This will include the apache configuration and PHP configuration file from `/etc`.

D.3.3. Nested tags

The tags that are supported by `Fileset` are:

- include
- exclude
- patternset

The `<include>` and the `<exclude>` tags must have a `name` attribute that contains the pattern to include/exclude.

D.4. PatternSet

The PatternSet data type defines patterns that can be grouped into sets and nested into FileSets [#FileSet]. Patterns can be specified by nested <include> or <exclude> elements.

Table D.4: *Attributes for `<patternset>` tag*

Name	Type	Description	Default	Required
`includes`	`String`	Comma- or space-separated list of patterns of files that must be included; all files are included when omitted.	n/a	No
`includesfile`	`String`	The name of a file; each line of this file is taken to be an include pattern.	n/a	No
`excludes`	`String`	comma- or space-separated list of patterns of files that must be excluded; no files (except default excludes) are excluded when omitted.	n/a	No

Name	Type	Description	Default	Required
excludesfile	String	The name of a file; each line of this file is taken to be an exclude pattern.	n/a	No

D.4.1. Usage Example

```
<patternset id = "no.tests">
  <include name = "**/*.php"/>
  <exclude name = "**/*Test*"/>
</patternset>
```

D.4.2. Nested tags

The <patternset> tag only supports <include> and <exclude>. The <include> and the <exclude> tags must have a name attribute that contains the pattern to include/exclude.

D.5. Path / Classpath

The Path data type can be used to respresent path structures. In many cases the path type will be used for nested <classpaentry> tags. E.g.

```
<path id = "project.class.path">
  <pathelement dir = "lib/"/>
  <pathelement dir = "ext/"/>
</paentry>

<target name = "blah">
  <taskdef name = "mytask" path = "myapp.phing.tasks.MyTask">
    <classpath refid = "project.class.path"/>
  </taskdef>
</target>
```

Table D.5: *Attributes for <paentry> tag*

Name	Type	Description	Default	Required
dir	String	Specific path to directory	n/a	No
path	String	A path (which contains multiple locations separated by path.separator) to add.	n/a	No

D.5.1. Nested tags

The <paentry> tag supports nested <fileset> and <dirset> tags.

Appendix E. Core filters

Filters are a subset of Phing `data types` which provide for the transformation of file contents during the operation of another task. For example, a filter might replace tokens in a file as part of a copy task.

Filters have to be defined within a `<filterchain>` context to work. Example:

```
<filterchain>
  <expandproperties />
</filterchain>
```

There are two ways to use a filter: System filters (the ones shipped with Phing) can be used with their own tag name, such as `<xsltfilter>`, `<expandpropertyfilter>` or `<tabtospaces>`. User-defined filters can use the way is to use the `<filterreader>` tag.

E.1. PhingFilterReader

The PhingFilterReader is used when you want to use filters that are not directly available through their own tag. Example:

```
<filterchain>
  <filterreader classname = "phing.filter.ReplaceTokens">
    <!-- other way to set attributes -->
    <param name = "begintoken" value = "@@" />
    <param name = "endtoken" value = "@@" />

    <!-- other way to set nested tags -->
    <param type = "token" key = "bar" value = "foo" />
  </filterreader>
</filterchain>
```

In the `filterreader` tag you have to specify the path the class is in. The `FilterReader` will then load this class and pass the parameters to the loaded filter. There are two types of parameters: First, you can pass "normal" parameters to the loaded filter. That means, you can pass parameters as if they were attributes. If you want to do this, you only specify the `name` and `value` attributes in the `param` tag. You can also pass nested elements to the filter. Then, you have to specify the `type` attribute. This attribute specifies the name of the nested tag.

The result of the example above is identical with the following code:

```
<filterchain>
  <replacetokens begintoken = "@@" endtoken = "@@">
    <token key = "bar" value = "foo" />
  </replacetokens>
</filterchain>
```

Table E.1: *Attributes for `<filterreader>`*

Name	Type	Description	Default	Required
`classname`	`String`	Name of class to use (in dot-path notation).	n/a	Yes
`classpath`	`String`	The classpath to use when including classes. This is added to PHP's include_path.	n/a	No
classpatxlink:href	`String`	Reference to classpath to use when including classes. This is added to PHP's include_path.	n/a	No

E.1.1. Nested tags

The `PhingFilterReader` supports nested <classpaentry>.

E.1.2. Advanced

In order to support the `<filterreader ... />` sytax, your class must extend the BaseParamFilterReader class. Most of the filters that are bundled with Phing can be invoked using this syntax. The noteable exception (at time of writing) is the ReplaceRegexp filter, which expects find/replace parameters that do not fit the name/value mold. For this reason, you must always use the shorthand `<replaceregexp .../>` to invoke this filter.

E.2. ExpandProperties

The ExpandProperties simply replaces property names with their property values. For example, if you have the following in your build file:

```
<property name = "description.txt" value = "This is a text file" />

<copy todir = "/tmp">
  <filterchain>
    <expandproperties />
  </filterchain>

  <fileset dir = ".">
    <include name = "**" />
  </fileset>
</copy>
```

And the string `${description.txt}` it will be replaced by `This is a text file`.

Table E.2: *Attributes for <expandproperties>*

Name	Type	Description	Default	Required
`level`	`String`	Control the level at which this message is reported. One of `error`, `warning`, `info`, `verbose`, `debug`.	`info`	No

E.3. HeadFilter

This filter reads the first n lines of a file; the others are not further passed through the filter chain. Usage example:

```
<filterchain>
  <headfilter lines = "20" />
</filterchain>
```

Table E.3: *Attributes for the <headfilter> tag*

Name	Type	Description	Default	Required
`lines`	`Integer`	Number of lines to read.	10	No

E.4. IconvFilter

The IconvFilter encodes file from `in` encoding to `out` encoding. Usage example:

```
<filterchain>
    <iconvfilter inputencoding = "UTF-8" outputencoding = "CP1251" />
</filterchain>
```

Table E.4: *Attributes for the `<iconvfilter>` tag*

Name	Type	Description	Default	Required
`inputencoding`	`String`	Input encoding.	n/a	Yes
`outputencoding`	`String`	Output encoding.	n/a	Yes

E.5. Line Contains

This filter is only "permeable" for lines that contain the expression given as parameter. For example, the following filterchain would only let all the lines pass that contain `class`:

```
<filterchain>
  <linecontains>
    <contains value = "class" />
  </linecontains>
</filterchain>
```

E.5.1. Nested tags

The `linecontains` tag must contain one or more `contains` tags. Latter must have a `value` attribute that has to be set to the string the line has to contain to be let through.

E.6. LineContainsRegexp

This filter is similar to Section E.5, “Line Contains ” but you can specify regular expressions instead of simple strings.

```
<filterchain>
  <linecontainsregexp>
    <regexp pattern = "foo(.*)bar" />
  </linecontainsregexp>
</filterchain>
```

E.6.1. Nested tags

The `LineContains` filter has to contain at least one `regexp` tag. This must have a `pattern` attribute that is set to a regular expression.

E.7. PrefixLines

This filter adds a prefix to every line. The following example will add the string `foo:` in front of every line.

```
<filterchain>
  <prefixlines prefix = "foo: " />
</filterchain>
```

Table E.5: *Attributes for the <prefixlines> tag*

Name	Type	Description	Default	Required
`prefix`	`String`	String to prepend to every line.	n/a	Yes

E.8. ReplaceTokens

The `ReplaceTokens` filter will replace certain tokens. Tokens are strings enclosed in special characters. If you want to replace `##BCHOME##` by the path to the directory set in the environment variable `BCHOME`, you could do the following:

```
<property environment = "env" />

<filterchain>
  <replacetokens begintoken = "##" endtoken = "##">
    <token key = "BCHOME" value = "${env.BCHOME}" />
  </replacetokens>
</filterchain>
```

Table E.6: *Attributes for the <replacetokens> tag*

Name	Type	Description	Default	Required
`begintoken`	`String`	The string that marks the beginning of a token.	@	No
`endtoken`	`String`	The string that marks the end of a token.	@	No

E.8.1. Nested tags

The `ReplaceTokens` filter must contain one or more `token` tags. These must have a `key` and a `value` attribute.

E.9. ReplaceTokensWithFile

The `ReplaceTokensWithFile` filter will replace certain tokens with the contents of a file. The name of the file to use as replacement is derived from the token name itself. Tokens are strings enclosed in special characters which are user selectable.

This filter could for example be used to insert code examples in documentation where the example code are real executable files kept outside the documentation.

If you for example want to replace `#!example1##` with the content of the file " `example1.php` " you could do the following

```
<filterchain>
      <replacetokenswithfile begintoken = "#!" endtoken = "##"
          dir = "exampledir/" postfix = ".php" />
      </filterchain>
```

The filer above will replace all tokens within the begin and end token specified with the contents of the file whose base name is that of the token with the added postfix ".php". Only the directory specified in the `dir` attribute is searched. If the file is not found the token is left untouched and an error message is given. It is important to note that *all* found tokens will be replaced with the corresponding file. So in the example below even `#!example2##` will be replaced with the content of the file " `example2.php` "

Table E.7: *Attributes for the <`replacetokenswithfile`> tag*

Name	Type	Description	Default	Required
`begintoken`	`String`	The string that marks the beginning of a token.	#@#	No
`endtoken`	`String`	The string that marks the end of a token.	#@#	No
`prefix`	`String`	A string that will be added in front of the token to construct the filename that will be used as source when replacing the token.	"	No
`postfix`	`String`	A string that will be added to the end of the token to construct the filename that will be used as source when replacing the token.	"	No
`dir`	`String`	The directory where to look for the files to use as replacements for the tokens	'./'	No
`translatehtml`	`Boolean`	If true all html special characters (e.g. ">") in the file to there corresponding html entities (e.g. ">") before the file is inserted.	`true`	No

E.9.1. Nested tags

None.

E.10. ReplaceRegexp

The `ReplaceRegexp` filter will perform a regexp find/replace on the input stream. For example, if you want to replace ANT with Phing (ignoring case) and you want to replace references to *.java with *.php:

```
<filterchain>
  <replaceregexp>
 <regexp pattern = "ANT" replace = "Phing" ignoreCase = "true"/>
 <regexp pattern = "(\w+)\.java" replace = "\1.php"/>
  </replaceregexp>
</filterchain>
```

Or, replace all Windows line-endings with Unix line-endings:

```
<filterchain>
  <replaceregexp>
 <regexp pattern = "\r(\n)" replace = "\1"/>
  </replaceregexp>
</filterchain>
```

E.10.1. Nested tags

The `ReplaceRegExp` filter must contain one or more `regexp` tags. These must have `pattern` and `replace` attributes. The full list of supported attributes is as following:

Table E.8: *Attributes for the <`regexp`> tag*

Name	Type	Description	Default	Required
`pattern`	`String`	Regular expression used as needle. Phing relies on Perl-compatible [http://php.net/pcre] regular expressions.	n/a	Yes
`replace`	`String`	Replacement string.	n/a	Yes
`ignoreCase`	`Boolean`	Whether search is case-insensitive.	`false`	No
`multiline`	`Boolean`	Whether regular expression is applied in multi-line mode.	`false`	No
`modifiers`	`String`	Raw regular expression modifiers [http://php.net/manual/en/reference.pcre.pattern.modifiers.php]. You can pass several modifiers as single string, and use raw modifiers with `ignoreCase` and `multiline` attributes. In case of conflict, value specified by dedicated attribute takes precedence.		No

The previous example (using `modifiers` attribute this time):

```
<filterchain>
  <replaceregexp>
 <regexp pattern = "ANT" replace = "Phing" modifiers = "i"/>
 <regexp pattern = "(\w+)\.java" replace = "\1.php"/>
  </replaceregexp>
</filterchain>
```

E.11. StripLineBreaks

The `StripLineBreaks` filter removes all linebreaks from the stream passed through the filter chain.

```
<filterchain>
  <striplinebreaks />
</filterchain>
```

E.12. StripLineComments

The `StripLineComments` filter removes all line comments from the stream passed through the filter chain:

```
<filterchain>
  <striplinecomments>
    <comment value = "#" />
    <comment value = "--" />
    <comment value = "//" />
```

```
  </striplinecomments>
</filterchain>
```

E.12.1. Nested tags

The `striplinecomments` tag must contain one or more `comment` tags. These must have a `value` attribute that specifies the character(s) that start a line comment.

E.13. StripPhpComments

The `StripPhpComments` filter removes all PHP comments from the stream passed through the filter.

```
<filterchain>
  <stripphpcomments />
</filterchain>
```

E.14. StripWhitespace

The `StripWhitespace` filter removes all PHP comments and whitespace from the stream passed through the filter. Internally, this filter uses the php_strip_whitespace() function.

```
<filterchain>
  <stripwhitespace />
</filterchain>
```

E.15. TabToSpaces

The TabToSpaces filter replaces all tab characters with a given count of space characters.

```
<filterchain>
  <tabtospaces tablength = "8" />
</filterchain>
```

Table E.9: *Attributes for the <`tabtospaces`> filter*

Name	Type	Description	Default	Required
`tablength`	`Integer`	The number of space characters that a tab is to represent.	8	No

E.16. TailFilter

Similar to Section E.3, “HeadFilter”, this filter reads the last n lines of a file; the others are not further passed through the filter chain. Usage example:

```
<filterchain>
```

```
  <tailfilter lines = "20" />
</filterchain>
```

Table E.10: *Attributes for the <tailfilter> tag*

Name	Type	Description	Default	Required
lines	Integer	Number of lines from the back to read.	10	No

E.17. TidyFilter

The TidyFilter allows you to use the PHP tidy extension [http://php.net/tidy] to clean up and repair HTML documents. Usage example:

```
<filterchain>
  <tidyfilter encoding = "utf8">
    <config name = "indent" value = "true" />
    <config name = "output-xhtml" value = "true" />
  </tidyfilter>
</filterchain>
```

Table E.11: *Attributes for the <tidyfilter> tag*

Name	Type	Description	Default	Required
encoding	String	The expected input encoding of the file.	utf8	No

E.17.1. Nested tags

The TidyFilter supports nested <config> tags to configure how Tidy should manipulate the documents. For a complete list of configuration options see the offical Quick Reference [http://tidy.sourceforge.net/docs/quickref.html].

E.18. XincludeFilter

The XincludeFilter processes a stream for Xinclude tags, and processes the inclusions. This is useful for processing modular XML files. DocBook book files are one example of modular XML files. Usage example:

```
<!--
  Render a DocBook book file called manual.xml, which
  contains Xinclude tags to include individual book sections.
  -->
<copy todir = "${manual.dest.dir}">
  <filterchain>
    <xincludefilter basedir = "${manual.src.dir}" />
    <xsltfilter style = "${manual.src.dir}/html.xsl">
      <param name = "base.dir" expression = "${manual.dest.dir}/" />
    </xsltfilter>
  </filterchain>
  <fileset dir = "${manual.src.dir}">
    <include name = "manual.xml" />
  </fileset>
</copy>
```

Table E.12: *Attributes for the <xincludefilter> tag*

Name	Type	Description	Default	Required
basedir	String	The working directory from which to process the Xincludes. Relative pathnames in the include tags are based on this location.	Project basedir	No
resolveexternals	Boolean	Whether to resolve entities. (see this link [http://www.php.net/manual/en/class.domdocument.php#domdocument.props.resolveexternals] for details)	false	No

E.19. XsltFilter

The XsltFilter applies a XSL template to the stream. Though you can use this filter directly, you should use XslTask Appendix B, *Core tasks* which is shortcut to the following lines:

```
<filterchain>
  <xsltfilter style = "somexslt.xsl" />
</filterchain>
```

This filter relies on PHP5 XSL support via libxslt which must be available for php5. Usually this means including the php5_xsl module when configuring PHP5. In essence this uses the same core libraries as "xsltproc" processor.

Table E.13: *Attributes for the <xsltfilter> tag*

Name	Type	Description	Default	Required
style	String	The XSLT stylesheet to use for transformation.	n/a	Yes
html	Boolean	Whether to parse the input as HTML (using libxml2 DOMDocument::loadHTML()).	false	No
resolvedocumentexternals	Boolean	Whether to resolve entities in the XML document. (see this link [http://www.php.net/manual/en/class.domdocument.php#domdocument.props.resolveexternals] for details)	false	No
resolvestyleshee-texternals	Boolean	Whether to resolve entities in the stylesheet.	false	No

E.19.1. Nested tags

The XsltFilter filter may contain one or more param tags to pass any XSLT parameters to the stylesheet. These param tags must have name and expression attributes.

Appendix F. Core mappers

While filters are applied to the `content` of files, Mappers are applied to the `filenames`. All mappers have the same API, i.e. the way you use them is the same:

```
<mapper type = "mappername" from="frompattern" to="topattern" />
```

F.1. Common Attributes

Table F.1: *Attributes for the <mapper> tag*

Name	Type	Description	Default	Required
`type`	`String`	Type of the mapper.	n/a	One of these is required.
`classname`	`String`	Dot-path to a custom mapper class to use.	n/a	
`from`	`String`	The pattern the filename is to be matched to. The exact meaning is dependent on the implementation of the mapper.	n/a	depends on the implementation of the mapper
`to`	`String`	The pattern according to which the filename is to be changed to. Here, the usage is dependent on the implementation of the mapper, too.	n/a	depends on the implementation of the mapper

F.2. FlattenMapper

The `FlattenMapper` removes the directories from a filename and solely returns the filename.

```
<copy todir = "/tmp">
  <mapper type = "flatten" />

  <fileset refid = "someid" />
</copy>
```

This code will copy all files in the fileset to /tmp. All files will be in the target directory.

F.2.1. Examples

```
<mapper type = "flatten" />
```

Applying the mapper, you will get the following results from the following filenames:

Table F.2: *Result of mapping*

From	To
`test.txt`	`test.txt`
`./foo/bar/test.bak`	`test.bak`

F.3. GlobMapper

The `GlobMapper` works like the `copy` command in DOS:

```
<copy todir = "/tmp">
  <mapper type = "glob" from = "*.php" to = "*.php.bak"/>

  <fileset refid = "someid" />
</copy>
```

This will change the extension of all files matching the pattern `*.php` to `.php.bak`.

F.3.1. Examples

```
<mapper type = "glob" from = "*txt" to = "*txt.bak"/>
```

Applying the mapper, you will get the following results from the following filenames:

Table F.3: *Result of mapping*

From	To
`test.txt`	`test.txt.bak`
`./foo/bar/test.txt`	`./foo/bar/test.txt.bak`
`mytxt`	`mytxt.bak`
`SomeClass.php`	`ignored`, SomeClass.php

F.4. IdentityMapper

The `IdentityMapper` will not change anything on the source filenames.

F.5. MergeMapper

The `MergeMapper` changes all source filenames to the same filename.

F.5.1. Examples

```
<mapper type = "merge" to = "test.tar"/>
```

Applying the mapper, you will get the following results from the following filenames:

Table F.4: *Result of mapping*

From	To
`test.txt`	`test.tar`
`./foo/bar/test.txt`	`test.tar`
`mytxt`	`test.tar`

From	To
`SomeClass.php`	`test.tar`

F.6. RegexpMapper

The `RegexpMapper` changes filenames according to a pattern defined by a regular expression. This is the most powerful mapper and you should be able to use it for every possible application.

F.6.1. Examples

```
<mapper type = "regexp" from = "^(.*)\.conf\.xml" to = "\1.php"/>
```

The mapper as above will do the following mappings:

Table F.5: *Result of mapping*

From	To
`test.txt`	ignore, `test.txt`
`./foo/bar/test.conf.xml`	`./foo/bar/test.php`
`someconf.conf.xml`	`someconf.php`

Appendix G. Core selectors

Selectors are a specific subset of Phing `data types` that allow you to fine-tune matching in a Appendix D, *Core Types*FileSet (or DirSet).

Phing supports the following core selectors, which typically match on both files and directories in a `<fileset>`:

- <Contains> - Select files that contain a specific string
- <date> - Select files/directories that have been modified either before or after a specific date/time
- <Depend> - Select files/directories that have been modified more recently than equivalent items elsewhere
- <Depth> - Select files/directories that appear at a specific depth in a directory tree
- <Filename> - Select files/directories whose name matches a particular pattern. Equivalent to the include and exclude elements of a patternset.
- <Present> - Select files/directories that either do or do not exist in some other location
- <Containsregexp><containsregexp> - Select files that contain text matching a regular expression
- <Size><size> - Select files that are larger or smaller than a particular number of bytes.
- <Type><type> - Select files/directories by type ('file' or 'dir')

Additionally, to create more complex selections, a variety of selectors that contain other selectors are available for your use. They combine the selections of their child selectors in various ways.

Phing supports the following selector containers:

- <And><and> - Select a file only if all the contained selectors select it.
- <Majority><majority> - Select a file only if all the contained selectors select it.
- <None><none> - Select a file only if none of the contained selectors select it.
- <Not><not> - Can contain only one selector, and reverses what it selects and doesn't select.
- <Or><or> - Select a file if any one of the contained selectors selects it.
- <Selector><selector> - Contains only one selector and forwards all requests to it without alteration. This is the selector to use if you want to define a reference. It is usable as an element of `<project>`.

G.1. Contains

The `<contains>` tag selects files that contain the string specified by the `text` attribute.

```
<fileset dir = "${src}" includes = "**/*.php">
  <contains text = "PHP"/>
</fileset>
```

Table G.1: *Attributes for the `<contains>` selector*

Name	Description	Default	Required
`text`	Specifies the text that every file must contain	n/a	Yes

Name	Description	Default	Required
`casesensitive`	Whether to pay attention to case when looking for the string in the text attribute.	`true`	No

G.2. Date

The <date> tag selects files whose last modified date meet the date limits specified by the selector.

```
<fileset dir = "${src}" includes = "**/*.php">
  <date datetime = "01/01/2001 12:00 AM" when = "before"/>
</fileset>
```

Table G.2: *Attributes for the <date> selector*

Name	Description	Default	Required
`datetime`	Specifies the date and time to test for. It should be in a format parsable by PHP's strtotime() [http://www.php.net/strtotime] function.	n/a	One of the two
`seconds`	The number of seconds since 1970 that should be tested for.	n/a	
`when`	Indicates how to interpret the date, whether the files to be selected are those whose last modified times should be before, after, or equal to the specified value. Accepted values are: `before` - select files whose last modified date is before the indicated date `after` - select files whose last modified date is after the indicated date `equal` - select files whose last modified date is this exact date	equal	No
`granularity`	The number of milliseconds leeway to use when comparing file modification times. This is needed because not every file system supports tracking the last modified time to the millisecond level.	0	No
`checkdirs`	Indicates whether or not to check dates on directories.	`false`	No

G.3. Depend

The <depend> tag selects files whose last modified date is later than another, equivalent file in another location.

The <depend> tag supports the use of a contained Appendix F, *Core mappers* element to define the location of the file to be compared against. If no `mapper` element is specified, the identity type mapper is used.

The <depend> tag is case-sensitive.

```
<fileset dir = "phing-2.4.5/classes" includes = "**/*.php">
  <depend targetdir = "phing-2.4.6/classes"/>
```

```
</fileset>
```

Table G.3: *Attributes for the <depend> selector*

Name	Description	Default	Required
`targetdir`	The base directory to look for the files to compare against. The precise location depends on a combination of this attribute and the `mapper` element, if any.	n/a	Yes
`granularity`	The number of milliseconds leeway to give before deciding a file is out of date. This is needed because not every file system supports tracking the last modified time to the millisecond level.	0	No

G.4. Depth

The `<dependtry>` tag selects files based on how many directory levels deep they are in relation to the base directory of the fileset.

```
<fileset dir = "phing/classes" includes = "**/*.php">
  <depth max = "1"/>
</fileset>
```

Table G.4: *Attributes for the <dependtry> selector*

Name	Description	Default	Required
`min`	The minimum number of directory levels below the base directory that a file must be in order to be selected.	0	One of the two
`max`	The maximum number of directory levels below the base directory that a file can be and still be selected.	0	

G.5. Filename

The `<filename>` tag acts like the `<include>` and `<exclude>` tags within a fileset. By using a selector instead, however, one can combine it with all the other selectors using whatever selector container is desired.

```
<fileset dir = "${src}" includes = "**/*">
  <filename name = "**/*.php">
</fileset>
```

Table G.5: *Attributes for the <filename> selector*

Name	Description	Default	Required
`name`	The name of files to select. The name parameter can contain the standard Ant wildcard characters.	n/a	Yes
`casesensitive`	Whether to pay attention to case when looking at file names.	`true`	No

Name	Description	Default	Required
negate	Whether to reverse the effects of this filename selection, therefore emulating an exclude rather than include tag.	false	No

G.6. Present

The <present> tag selects files that have an equivalent file in another directory tree.

The <present> tag supports the use of a contained Appendix F, *Core mappers*Mapper element to define the location of the file to be compared against. If no mapper element is specified, the identity type mapper is used.

The <present> tag is case-sensitive.

```
<fileset dir = "phing-2.4.6/classes" includes = "**/*.php">
  <present present = "srconly" targetdir = "phing-2.4.5/classes">
</fileset>
```

Table G.6: *Attributes for the <present> selector*

Name	Description	Default	Required
targetdir	The base directory to look for the files to compare against. The precise location depends on a combination of this attribute and the <mapper> element, if any.	n/a	Yes
present	Whether we are requiring that a file is present in the src directory tree only, or in both the src and the target directory tree. Valid values are: srconly - select files only if they are in the src directory tree but not in the target directory tree both - select files only if they are present both in the src and target directory trees	both	No

G.7. Containsregexp

The <containsregexp> tag selects the files whose contents contain a match to the regular expression specified by the expression attribute.

```
<fileset dir = "${src}" includes = "*.txt">
  <containsregexp expression = "[4-6]\.[0-9]"/>
</fileset>
```

Table G.7: *Attributes for the <containsregexp> selector*

Name	Description	Default	Required
expression	Specifies the regular expression that must match true in every file.	n/a	Yes
casesensitive	Perform a case sensitive match.	true	No

G.8. Size

The <size> tag selects files matching a specified size limit.

```
<fileset dir = "${src}">
  <size value = "4" when = "more"/>
</fileset>
```

Table G.8: *Attributes for the <size> selector*

Name	Description	Default	Required
value	The size of the file which should be tested for.	n/a	Yes
units	The units that the value attribute is expressed in. When using the standard single letter SI designations, such as "k","M", or "G", multiples of 1000 are used. If you want to use power of 2 units, use the IEC standard: "Ki" for 1024, "Mi" for 1048576, and so on. The default is no units, which means the value attribute expresses the exact number of bytes.	n/a	No
when	Indicates how to interpret the size, whether the files to be selected should be larger, smaller, or equal to that value. Accepted values are: less - select files less than the indicated size more - select files greater than the indicated size equal - select files this exact size	equal	No

G.9. Type

The <type> tag selects files of a certain type: directory or regular.

```
<fileset dir = "${src}">
  <type type = "dir"/>
</fileset>
```

Table G.9: *Attributes for the <type> selector*

Name	Description	Default	Required
type	The type of file which should be tested for. Either file or dir.	n/a	Yes

G.10. And

The <and> tag selects files that are selected by all of the elements it contains. It returns as soon as it finds a selector that does not select the file, so it is not guaranteed to check every selector.

```
<fileset dir = "${src}" includes = "**/*.php">
  <and>
    <size value = "1000" when = "more"/>
    <date datetime = "01/01/2011 12:00 AM" when = "before"/>
  </and>
```

```
</fileset>
```

G.11. Majority

The <majority> tag selects files provided that a majority of the contained elements also select it. Ties are dealt with as specified by the allowtie attribute.

```
<fileset dir = "${src}" includes = "**/*.php">
  <majority>
    <contains text = "project" casesensitive = "false"/>
    <contains text = "taskdef" casesensitive = "false"/>
    <contains text = "BaseSelector" casesensitive = "true"/>
  </majority>
</fileset>
```

Table G.10: *Attributes for the <majority> selector container*

Name	Description	Default	Required
allowtie	Whether files should be selected if there are an even number of selectors selecting them as are not selecting them.	true	No

G.12. None

The <none> tag selects files that are not selected by any of the elements it contains. It returns as soon as it finds a selector that selects the file, so it is not guaranteed to check every selector.

```
<fileset dir = "${src}" includes = "**/*.php">
  <none>
    <size value = "1000" when = "more"/>
    <date datetime = "01/01/2011 12:00 AM" when = "before"/>
  </none>
</fileset>
```

G.13. Not

The <not> tag reverses the meaning of the single selector it contains.

```
<fileset dir = "${src}" includes = "**/*.php">
  <not>
    <contains text = "Phing"/>
  </not>
</fileset>
```

G.14. Or

The <or> tag selects files that are selected by any one of the elements it contains. It returns as soon as it finds a selector that selects the file, so it is not guaranteed to check every selector.

```
<fileset dir = "${src}">
  <or>
    <depth max = "0"/>
    <filename name = "*.png"/>
    <filename name = "*.gif"/>
    <filename name = "*.jpg"/>
  </or>
</fileset>
```

G.15. Selector

The <selector> tag is used to create selectors that can be reused through references. It is the only selector which can be used outside of any target, as an element of the <project> tag. It can contain only one other selector, but of course that selector can be a container.

Appendix H. Project Components

This file will give you a quick introduction and a reference of the things that you may see in a build files besides Appendix B, *Core tasks*tasks and Appendix D, *Core Types*types.

H.1. Phing Projects

Projects are the outermost container for everything in build files. The `<project>` tag also is the root tag in build files. It contains the name, the directory, a short description and a default target.

Project may contain task calls and targets (see below).

H.1.1. Example

```
<?xml version="1.0" ?>

<project name = "TestProject" basedir = "." default = "main"
         description = "This is a test project to show how to use projects ;-)">

  <!-- Everything else goes here -->

</project>
```

H.1.2. Attributes

Table H.1: *Attributes*

Name	Type	Description	Default	Required
`basedir`	`String`	The base directory of the project, i.e. the directory all paths are relative to.	n/a	No
`default`	`String`	The name of the target that is executed if none is explicitely specified when calling Phing	all	Yes
`description`	`String`	A free text description of the project	n/a	No
`name`	`String`	Name of the project	n/a	No
`phingVersion`	`String`	The minimum Phing version required to execute the build file, in order to prevent compatibility issues.	n/a	No

H.2. Targets

H.2.1. Example

```
<target if = "lang" unless = "lang.en" depends = "foo1,foo2"
        name = "main" description = "This is an example target" >

  <!-- everything else goes here -->
```

```
</target>
```

The target defined in the example above is only executed, if the property `${lang}` is set and the property `${lang.en}` is not set. Additionally, it depends on the targets `foo1` and `foo2`. That means, the targets `foo1` and `foo2` are executed before the target `main` is executed. The name of the target is `main` and it also has a description.

H.2.2. Attributes

Table H.2: *Parameters*

Name	Type	Description	Default	Required
`depends`	`String`	One or more names of targets that have to be executed before this target can be executed.	n/a	No
`description`	`String`	A free text description of the target.	n/a	No
`if`	`String`	The name of the property that is to be set if the target is to be executed.	n/a	No
`name`	`String`	The name of the target	n/a	Yes
`unless`	`String`	The name of the property that is to be set if the target is not to be executed.	n/a	No
`hidden`	`Boolean`	Whether or not to include this target in the list of targets generated by `phing -l`	False	No

Caution

The if and unless attributes only enable or disable the target to which they are attached. They do not control whether or not targets that a conditional target depends upon get executed. In fact, they do not even get evaluated until the target is about to be executed, and all its predecessors have already run.

Appendix I. File Formats

I.1. Build File Format

The following XML file shows a basic Phing build file skeleton that can be used as a starting point for your own build files. See the references in Appendix A, *Fact Sheet* and Appendix B, *Core tasks* for more detailed information on properties and tasks.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  ======================================================================
  The root tag of each build file must be a "project" tag.
  ======================================================================
-->
<project name = "(projectname)" basedir = "(projectbasedir)"
         default = "(targetname)" description = "(projectdescription)">

  <!--
    ====================================================================
    Inclusion of optional overall project properties.
    ====================================================================
  -->
  <property file = "(main property file)" />

  <!--
    ====================================================================
     Build file wide properties used in the targets below
    ====================================================================
  -->

  <!-- Useful to make the current buildtime available as a property -->
  <tstamp>
      <!-- Format is, e.g. Sat, 03 Oct 2009, 16:31 -->
      <format property = "buildtime" pattern = "%a, %d %b %Y, %H:%M"/>
  </tstamp>

  <property name = "(first.property1)"  value = "(value1)" override = "true" />
  <property name = "(second.property2)"  value = "(value2)" override = "true" />

  <!--
    ====================================================================
    Type and task calls here, i.e. filesets, patternsets,
    CopyTask calls etc.
    ====================================================================
  -->
    <!-- Filesets  -->
    <fileset dir = "(fileset.directory)" id = "(fileset.reference)">
        <include name = "(include.pattern)"/>
    </fileset>

    <!-- Custom tasks  -->
    <taskdef classname = "(task.classname)" name = "task.name" />

  <!--
    ====================================================================
    All target definitions
    ("if" and "unless" attributes are optional)
    ====================================================================
  -->
```

```
  <target name = "(targetname)" [depends = "targetname1,targetname2"]
          [if = "(ifproperty)"]  [unless = "(unlessproperty)"]  >
    <!--
      ================================================================
      Type and task calls here, i.e. filesets, patternsets,
      CopyTask calls, etc.
      ================================================================
    -->
  </target>

  <!--
    ================================================================
    More targets here
    ================================================================
  -->
  <target name = "..." >
    <!--
      ================================================================
      Type and task calls here, i.e. filesets, patternsets,
      CopyTask calls, etc.
      ================================================================
    -->

  </target>
</project>
```

Note

By convention properties are named in *dot* notation in Phing build files, e.g. `ftp.upload`, `temp.builddir` and so on

I.2. Property File Format

Property Files define properties. Properties are stored in key/value pairs and may only contain plain text. The suffix of these files should be `.properties`, the default Property File for a Build File is `build.properties`

```
# Property files contain key/value pairs
key=value

# Property keys may contain alphanumeric chars and colons, but
# not special chars. This way you can create pseudo-namespaces
myapp.window.hsize=300
myapp.window.vsize=200
myapp.window.xpos=10
myapp.window.ypos=100

# You can refer to values of other properties by enclosing their
# keys in "${}".
text.width=${myapp.window.hsize}

# Everything behind the equal sign is the value, you do
# not have to enclose strings:
text=This is some text, Your OS is ${php.os}

# I guess that is all there is to property files
```

Bibliography

International Standards

[osi-model] *OSI (Open System Interconnect) Model*. http://www.iso.org . http://www.instantweb.com/foldoc/foldoc.cgi?OSI .

[xml10-spec] *W3C XML 1.0 Specifications*. http://www.w3.org/XML/ .

[unicode] *Unicode*. http://www.unicode.org .

Licenses

[gnu-lgpl] *The GPL (Gnu Lesser Public License)*. http://www.gnu.org/licenses/lgpl.html .

[gnu-fdl] *The Gnu FDL (Free Documentation License), the license used for this documentation*. http://www.gnu.org/licenses/fdl.html .

Open Source Projects

[bc] *Binarycloud*. http://www.binarycloud.com http://binarycloud.tigris.org .

[w3c-tidy] *HTMLTidy, a W3C (x)HTML and XML syntax checker and code beautifier*. http://www.w3c.org/People/Ragget/tidy/ [http://www.w3c.org/People/Raggett/tidy/] .

[phpdoc] *The PHPDoc Project*. http://www.phpdoc.de .

[phpclasses] *Manuel Lemos' PHPClasses Repository*. http://www.phpclasses.org .

[pear] *PEAR (Php Extension Archive Repository)*. http://pear.php.net .

[ant] *Ant, a Java Build Tool, the main inspiration for Phing*. http://ant.apache.org .

[gnumake] *GNU make, an inspiration for Phing*. http://www.gnu.org/software/make/make.html .

[pollo] *Pollo, a visual editor for XML files. A schema to edit phing build files is shipped with Phing.*. http://pollo.sourceforge.net .

[gingerall] *Ginger Alliance - Home Of Sablotorn*. http://www.gingerall.com .

[php] *The PHP homepage - PHP Hypertext Preprocessor*. http://www.php.net .

[gnu] *The GNU (GNU's Not Unix) Organization*. http://www.gnu.org .

[phing] *Phing (PHing Is Not Gnumake)*. http://www.phing.info .

Manuals

[svn-howto] *Version Control with Subversion (free book)*. http://svnbook.red-bean.com/ .

[git-book] *Pro-git (free book)*. http://progit.org/ .

Other Resources

[javadoc] *Sun Javadoc*. http://java.sun.com/j2se/javadoc/ .

[zend] *Zend Technologies, Ltd.*. http://www.zend.com .