



Phing User Guide

Michiel Rook <mrook@php.net>

Ken Guest <ken@linux.ie>

Siad Ardroumli <siad.ardroumli@gmail.com>

Phing User Guide

by Michiel Rook, Ken Guest, and Siad Ardroumli

Phing 3.x

Publication date 2025-04-12 18:17:30

Copyright © 2002-2022 The Phing Project

Preface	xv
1. About this book	1
1.1. Contributors (present and past)	1
1.2. Copyright	1
1.3. License	1
1.4. DocBook	1
1.4.1. Building the documentation	2
1.4.2. Template for new tasks	4
1.4.3. Customization of the look & feel of the rendered outputs	4
1.4.4. DocBook v5 elements used in the manual and their meaning	5
2. Introduction	11
2.1. What Phing Is	11
2.2. Phing & Binarycloud: History	11
2.3. How Phing Works	12
2.4. Cool, so how can I help?	12
2.4.1. Participating in the development	12
3. Setting-up Phing	15
3.1. System Requirements	15
3.1.1. Operating Systems	15
3.1.2. Software Dependencies	15
3.2. Obtaining Phing	15
3.2.1. Distribution Files	15
3.2.2. Composer Install	15
3.2.3. Phar package	16
3.2.4. Getting the latest source from Phing's Github repository	16
3.3. Running Phing	16
3.3.1. Command Line	16
3.3.2. Supported command line arguments	16
4. Getting started	19
4.1. XML And Phing	19
4.2. Writing A Simple Buildfile	19
4.2.1. Project Element	20
4.2.2. Target Element	21
4.2.3. Task Elements	21
4.2.4. Property Element	22
4.3. More Complex Buildfile	22
4.3.1. Handling source dependencies	23
4.4. Relax NG Grammar	24
5. Project components	25
5.1. Projects	25
5.2. Version	25
5.3. Project Components in General	25
5.4. Targets	26
5.5. Tasks	26
5.6. Types	27
5.6.1. Basics	27
5.6.2. Referencing Types	27
5.7. Basic Types	28
5.7.1. FileSet	28
5.7.2. FileList	28
5.7.3. FilterChains and Filters	28
5.7.4. File Mappers	29
5.8. Conditions	30
5.8.1. not	30
5.8.2. and	30
5.8.3. or	30
5.8.4. xor	30
5.8.5. os	31

5.8.6. equals	31
5.8.7. versioncompare	31
5.8.8. http	32
5.8.9. PDOSQLException	32
5.8.10. socket	33
5.8.11. hasfreespace	33
5.8.12. isset	34
5.8.13. contains	34
5.8.14. istrue	34
5.8.15. isfalse	35
5.8.16. ispropertytrue	35
5.8.17. ispropertyfalse	35
5.8.18. referenceexists	35
5.8.19. available	36
5.8.20. filesmatch	36
5.8.21. isfileselected	36
5.8.22. isfailure	36
5.8.23. matches	37
6. Extending Phing	39
6.1. Extension Possibilities	39
6.1.1. Tasks	39
6.1.2. Types	39
6.1.3. Mappers	39
6.2. Source Layout	39
6.2.1. Files And Directories	39
6.2.2. File Naming Conventions	40
6.2.3. Coding Standards	41
6.3. System Initialization	41
6.3.1. Wrapper Scripts	41
6.3.2. The Main Application (phing.php)	41
6.3.3. The Phing Class	42
6.4. System Services	42
6.4.1. The Exception system	42
6.5. Build Lifecycle	42
6.5.1. How Phing Parses Buildfiles	42
6.6. Writing Tasks	43
6.6.1. Creating A Task	43
6.6.2. Using the Task	44
6.6.3. Source Discussion	44
6.6.4. Task Structure	44
6.6.5. Includes	45
6.6.6. Class Declaration	45
6.6.7. Class Properties	45
6.6.8. The Constructor	45
6.6.9. Setter Methods	46
6.6.10. Creator Methods	46
6.6.11. init() Method	47
6.6.12. main() Method	47
6.6.13. Arbitrary Methods	47
6.7. Writing Types	47
6.7.1. Creating a DataType	48
6.7.2. Using the DataType	49
6.7.3. Source Discussion	50
6.8. Writing Mappers	50
6.8.1. Creating a Mapper	51
6.8.2. Using the Mapper	51
6.9. Writing Selectors	52
6.10. Writing Conditions	52

A. Fact Sheet	55
A.1. Built-In Properties	55
A.2. Command Line Arguments	55
A.3. Distribution File Layout	56
A.4. Program Exit Codes	57
A.5. The LGPL License	57
A.6. The GFDL License	65
B. Core tasks	73
B.1. AdhocTaskdefTask	73
B.1.1. Examples	73
B.2. AdhocTypedefTask	73
B.2.1. Example	74
B.3. AppendTask	74
B.3.1. Examples	75
B.3.2. Supported Nested Tags	75
B.4. ApplyTask	76
B.4.1. Examples	77
B.4.2. Supported Nested Tags	78
B.5. AttribTask	78
B.5.1. Example	79
B.5.2. Supported Nested Tags	79
B.6. Augment	79
B.6.1. Examples	80
B.7. AutoloaderTask	80
B.7.1. Example	80
B.8. AvailableTask	80
B.8.1. Examples	81
B.9. Basename	81
B.9.1. Examples	81
B.10. Bindtargets	82
B.10.1. Examples	82
B.11. ChmodTask	82
B.11.1. Examples	82
B.11.2. Supported Nested Tags	83
B.12. ChownTask	83
B.12.1. Examples	83
B.12.2. Supported Nested Tags	83
B.13. ConditionTask	83
B.13.1. Examples	84
B.13.2. Supported Nested Tags	84
B.14. CopyTask	84
B.14.1. Examples	85
B.14.2. Supported Nested Tags	86
B.15. DefaultExcludes	86
B.15.1. Examples	86
B.16. DeleteTask	87
B.16.1. Examples	87
B.16.2. Supported Nested Tags	88
B.17. DependSet	88
B.17.1. Examples	88
B.17.2. Supported Nested Tags	88
B.18. Diagnostics	89
B.18.1. Example	89
B.19. Dirname	89
B.19.1. Example	89
B.20. EchoPropertiesTask	89
B.20.1. Example	90
B.21. EchoTask	90

B.21.1. Examples	91
B.21.2. Supported Nested Tags	91
B.22. EchoXML	91
B.22.1. Parameters specified as nested elements	92
B.22.2. Examples	92
B.23. ExecTask	92
B.23.1. Examples	93
B.23.2. Supported Nested Tags	94
B.24. FailTask	95
B.24.1. Examples	95
B.24.2. Parameters specified as nested elements.	95
B.25. FileHashTask	96
B.25.1. Example	96
B.26. FileSizeTask	96
B.26.1. Examples	97
B.27. ForeachTask	97
B.27.1. Examples	98
B.27.2. Supported Nested Tags	98
B.28. IfTask	98
B.28.1. Examples	99
B.29. ImportTask	99
B.29.1. Target Overriding	99
B.29.2. Special Properties	100
B.29.3. Resolving Files Against the Imported File	100
B.29.4. Examples	100
B.30. IncludePathTask	101
B.30.1. Examples	101
B.31. InputTask	101
B.31.1. Examples	102
B.32. JsonValidateTask	102
B.32.1. Example	102
B.33. LoadFileTask	102
B.33.1. Examples	103
B.33.2. Supported Nested Tags:	103
B.34. ManifestTask	103
B.34.1. Supported Nested Tags	103
B.35. MkdirTask	103
B.35.1. Examples	104
B.36. MoveTask	104
B.36.1. Examples	104
B.36.2. Attributes and Nested Elements	104
B.37. PathConvert	104
B.38. PathToFileSetTask	105
B.38.1. Examples	106
B.39. PhingCallTask	106
B.39.1. Examples	106
B.39.2. Supported Nested Tags	107
B.40. PhingTask	107
B.40.1. Examples	108
B.40.2. Supported Nested Tags	108
B.40.3. Base directory of the new project	108
B.41. Phingversion	108
B.41.1. Example	109
B.42. PhpEvalTask	109
B.42.1. Examples	109
B.42.2. Supported Nested Tags	110
B.43. PhpLintTask	110
B.43.1. Example	111

B.43.2. Supported Nested Tags	111
B.44. PropertyCopy	111
B.44.1. Example	111
B.45. PropertyRegexTask	111
B.45.1. Match expressions	112
B.45.2. Replace	112
B.45.3. Example	113
B.46. PropertySelector	113
B.46.1. Select expressions	114
B.46.2. Example	114
B.47. PropertyTask	114
B.47.1. Examples	115
B.47.2. Supported Nested Tags:	115
B.48. Record	115
B.48.1. Example	116
B.49. ReflexiveTask	116
B.49.1. Examples	117
B.49.2. Supported Nested Tags:	117
B.50. Relentless	117
B.50.1. Example	118
B.51. ReplaceRegexpTask	119
B.51.1. Supported Nested Tags	119
B.52. ResolvePathTask	119
B.52.1. Examples	120
B.53. Retry	120
B.53.1. Example	120
B.54. RunTargetTask	120
B.54.1. Example	121
B.55. SleepTask	121
B.55.1. Example	121
B.56. SortList	121
B.56.1. Example	122
B.57. Subphing Task	122
B.57.1. Supported Nested Tags	123
B.58. SwitchTask	123
B.58.1. Supported Nested Tags	123
B.58.2. Examples	123
B.59. SymlinkTask	124
B.59.1. Example	124
B.59.2. Supported Nested Tags	124
B.60. TaskdefTask	125
B.60.1. Examples	125
B.60.2. Supported Nested Tags	125
B.61. Tempfile Task	125
B.61.1. Example	126
B.62. ThrowTask	126
B.62.1. Example	126
B.63. TouchTask	127
B.63.1. Examples	127
B.63.2. Supported Nested Tags	128
B.64. TruncateTask	128
B.64.1. Examples	129
B.65. TryCatchTask	129
B.65.1. Examples	129
B.66. TstampTask	130
B.66.1. Examples	130
B.66.2. Supported Nested Tags	130
B.66.3. ICU syntax	131

B.67. TypedefTask	132
B.67.1. Examples	132
B.67.2. Supported Nested Tags	132
B.68. URLEncodeTask	132
B.68.1. Example	133
B.69. UpToDateTask	133
B.69.1. Examples	133
B.69.2. Supported Nested Tags	134
B.70. Variable	134
B.70.1. Example	134
B.71. VersionTask	135
B.71.1. Example	135
B.72. WaitForTask	136
B.72.1. Examples	136
B.72.2. Supported Nested Tags	136
B.73. XsltTask	136
B.73.1. Examples	137
B.73.2. Supported Nested Tags	137
C. Optional tasks	139
C.1. ApiGenTask	139
C.1.1. Example	140
C.2. ComposerTask	140
C.2.1. Supported Nested Tags	141
C.2.2. Example	141
C.3. CoverageMergerTask	141
C.3.1. Example	142
C.3.2. Supported Nested Tags	142
C.4. CoverageReportTask	142
C.4.1. Example	142
C.4.2. Supported Nested Tags	142
C.5. CoverageSetupTask	143
C.5.1. Example	143
C.5.2. Supported Nested Tags	143
C.6. CoverageThresholdTask	144
C.6.1. Example	144
C.6.2. Supported Nested Tags	144
C.7. DbDeployTask	145
C.7.1. Example	145
C.8. FileSyncTask	146
C.8.1. Examples	147
C.9. FtpDeployTask	147
C.9.1. Example	148
C.9.2. Supported Nested Tags	148
C.10. GitArchiveTask	148
C.10.1. Example	149
C.11. GitBranchTask	149
C.11.1. Example	150
C.12. GitCheckoutTask	151
C.12.1. Example	152
C.13. GitCloneTask	152
C.13.1. Example	153
C.14. GitCommitTask	153
C.14.1. Example	153
C.14.2. Supported Nested Tags	154
C.15. GitDescribeTask	154
C.15.1. Example	155
C.16. GitFetchTask	155
C.16.1. Example	156

C.17. GitGcTask	156
C.17.1. Example	157
C.18. GitInitTask	157
C.18.1. Example	158
C.19. GitLogTask	158
C.19.1. Example	158
C.20. GitMergeTask	159
C.20.1. Example	160
C.21. GitPullTask	160
C.21.1. Example	161
C.22. GitPushTask	162
C.22.1. Example	162
C.23. GitTagTask	163
C.23.1. Example	164
C.24. GrowlNotifyTask	164
C.24.1. Examples	166
C.25. HgAddTask	167
C.25.1. Example	167
C.25.2. Supported Nested Tags	167
C.26. HgArchiveTask	167
C.26.1. Example	168
C.27. HgCloneTask	168
C.27.1. Example	168
C.28. HgCommitTask	168
C.28.1. Example	169
C.29. HgInitTask	169
C.29.1. Example	169
C.30. HgLogTask	169
C.30.1. Example	170
C.31. HgPullTask	170
C.31.1. Example	170
C.32. HgPushTask	170
C.32.1. Example	170
C.33. HgRevertTask	171
C.33.1. Example	171
C.34. HgTagTask	171
C.34.1. Example	171
C.35. HgUpdateTask	171
C.35.1. Example	172
C.36. HttpGetTask	172
C.36.1. Example	172
C.36.2. Supported Nested Tags	172
C.36.3. Global configuration	173
C.37. HttpRequestTask	173
C.37.1. Example	174
C.37.2. Supported Nested Tags	174
C.37.3. Global configuration	175
C.38. IniFileTask	175
C.38.1. Supported Nested Tags	176
C.38.2. Example	176
C.39. IoncubeEncoderTask	177
C.39.1. Example	178
C.39.2. Supported Nested Tags	179
C.40. IoncubeLicenseTask	179
C.40.1. Example	179
C.40.2. Supported Nested Tags	180
C.41. JsHintTask	180
C.41.1. Example	180

C.41.2. Supported Nested Tags	180
C.42. JsMinTask	181
C.42.1. Example	181
C.42.2. Supported Nested Tags	181
C.43. JslLintTask	181
C.43.1. Example	182
C.43.2. Supported Nested Tags	182
C.44. LiquibaseChangeLogTask	182
C.44.1. Example	183
C.44.2. Supported Nested Tags	183
C.45. LiquibaseDbDocTask	183
C.45.1. Example	184
C.45.2. Supported Nested Tags	184
C.46. LiquibaseDiffTask	184
C.46.1. Example	185
C.46.2. Supported Nested Tags	185
C.47. LiquibaseRollbackTask	186
C.47.1. Example	186
C.47.2. Supported Nested Tags	187
C.48. LiquibaseTagTask	187
C.48.1. Example	188
C.48.2. Supported Nested Tags	188
C.49. LiquibaseTask	188
C.49.1. Example	189
C.49.2. Supported Nested Tags	189
C.50. LiquibaseUpdateTask	190
C.50.1. Example	190
C.50.2. Supported Nested Tags	191
C.51. MailTask	191
C.51.1. Example	191
C.51.2. Supported Nested Tags	191
C.52. NotifySendTask	192
C.53. OpenTask	192
C.53.1. Examples	192
C.54. PDOSQLiteExecTask	192
C.54.1. Example	193
C.54.2. Supported Nested Tags	194
C.55. PHPMDTask	195
C.55.1. Example	196
C.55.2. Supported Nested Tags	196
C.56. PHPStanTask	197
C.56.1. Supported Nested Tags	198
C.56.2. Example	198
C.57. PHPUnitReport	198
C.57.1. Example	199
C.58. PHPUnitTask	199
C.58.1. Supported Nested Tags	200
C.58.2. Example	201
C.58.3. Supported Nested Tags	202
C.59. ParallelTask	202
C.59.1. Example	203
C.60. PatchTask	203
C.60.1. Example	204
C.61. PharDataTask	204
C.61.1. Example	204
C.61.2. Supported Nested Tags	205
C.62. PharPackageTask	205
C.62.1. Example	205

C.62.2. Supported Nested Tags	206
C.63. PhkPackageTask	206
C.63.1. Example	206
C.63.2. Supported Nested Tags	207
C.64. PhpCSTask	207
C.64.1. Supported Nested Tags	207
C.64.2. Examples	208
C.65. PhpDependTask	208
C.65.1. Example	209
C.65.2. Supported Nested Tags	209
C.66. PhpDocumentor2Task	209
C.66.1. Example	210
C.66.2. Supported Nested Tags	210
C.67. rSTTask	210
C.67.1. Features	211
C.67.2. Examples	211
C.67.3. Supported Nested Tags	213
C.68. S3GetTask	213
C.68.1. Example	214
C.69. S3PutTask	214
C.69.1. Example	215
C.69.2. Supported Nested Tags	215
C.70. SassTask	216
C.70.1. Example	217
C.70.2. Supported Nested Tags	217
C.71. ScpTask	217
C.71.1. Example	218
C.71.2. Supported Nested Tags	218
C.72. SmartyTask	219
C.73. SonarTask	219
C.73.1. Examples	220
C.73.2. Supported Nested Tags	221
C.74. SshTask	221
C.74.1. Example	222
C.74.2. Supported Nested Tags	222
C.75. StopwatchTask	222
C.75.1. Example	222
C.76. SvnCheckoutTask	223
C.76.1. Example	223
C.77. SvnCommitTask	224
C.77.1. Example	224
C.78. SvnCopyTask	225
C.78.1. Example	225
C.79. SvnExportTask	225
C.79.1. Example	226
C.80. SvnInfoTask	226
C.80.1. Example	227
C.81. SvnLastRevisionTask	227
C.81.1. Example	228
C.82. SvnListTask	228
C.82.1. Example	228
C.83. SvnLogTask	229
C.83.1. Example	229
C.84. SvnPropgetTask	229
C.84.1. Example	230
C.85. SvnProplistTask	230
C.85.1. Example	231
C.86. SvnPropsetTask	231

C.86.1. Example	231
C.87. SvnRevertTask	231
C.88. SvnSwitchTask	232
C.88.1. Example	232
C.89. SvnUpdateTask	233
C.89.1. Example	233
C.90. SymfonyConsoleTask	233
C.90.1. Examples	234
C.90.2. Supported Nested Tags	234
C.91. TarTask	234
C.91.1. Example	235
C.91.2. Supported Nested Tags	235
C.92. UntarTask	235
C.92.1. Example	236
C.92.2. Supported Nested Tags	236
C.93. UnzipTask	236
C.93.1. Example	236
C.93.2. Supported Nested Tags	236
C.94. VisualizerTask	237
C.94.1. Examples	237
C.94.2. Limitations	238
C.94.3. Requirements	238
C.94.4. Advanced HTTP configuration	238
C.95. WikiPublishTask	239
C.95.1. Example	240
C.96. XmlLintTask	240
C.96.1. Examples	240
C.96.2. Supported Nested Tags	241
C.97. XmlPropertyTask	241
C.97.1. Example	241
C.98. ZSDTPackTask	242
C.98.1. Example	243
C.99. ZSDTValidateTask	243
C.99.1. Example	243
C.100. ZendCodeAnalyzerTask	243
C.100.1. Example	244
C.100.2. Supported Nested Tags	244
C.101. ZipTask	244
C.101.1. Example	245
C.101.2. Supported Nested Tags	245
D. Core Types	247
D.1. Description	247
D.1.1. Usage Examples	247
D.2. Excludes	247
D.2.1. Nested tags	247
D.2.2. Usage Examples	247
D.3. FileList	248
D.3.1. Usage Examples	248
D.4. FileSet	248
D.4.1. Using wildcards	249
D.4.2. Usage Examples	249
D.4.3. Nested tags	249
D.5. DirSet	250
D.5.1. Using wildcards	250
D.5.2. Usage Examples	251
D.5.3. Nested tags	251
D.6. PatternSet	251
D.6.1. Usage Example	252

D.6.2. Nested tags	252
D.7. Path / Classpath	252
D.7.1. Nested tags	252
D.8. Regexp	252
D.8.1. Examples	253
E. Core filters	255
E.1. PhingFilterReader	255
E.1.1. Nested tags	256
E.1.2. Advanced	256
E.2. ExpandProperties	256
E.3. ConcatFilter	256
E.4. HeadFilter	257
E.5. IconvFilter	257
E.6. Line Contains	257
E.6.1. Nested tags	258
E.7. LineContainsRegexp	258
E.7.1. Nested tags	258
E.8. PrefixLines	258
E.9. ReplaceTokens	259
E.9.1. Nested tags	259
E.10. ReplaceTokensWithFile	259
E.10.1. Nested tags	260
E.11. ReplaceRegexp	260
E.11.1. Nested tags	260
E.12. SortFilter	261
E.13. StripLineBreaks	261
E.14. StripLineComments	262
E.14.1. Nested tags	262
E.15. StripPhpComments	262
E.16. StripWhitespace	262
E.17. TabToSpaces	262
E.18. TailFilter	263
E.19. TidyFilter	263
E.19.1. Nested tags	263
E.20. XincludeFilter	263
E.21. XsltFilter	264
E.21.1. Nested tags	264
E.22. ClassConstants	265
F. Core mappers	267
F.1. Common Attributes	267
F.2. ChainedMapper	267
F.2.1. Examples	267
F.3. CompositeMapper	268
F.3.1. Examples	268
F.4. FirstMatchMapper	268
F.4.1. Examples	268
F.5. CutDirsMapper	269
F.5.1. Examples	269
F.6. FlattenMapper	269
F.6.1. Examples	269
F.7. GlobMapper	269
F.7.1. Examples	270
F.8. IdentityMapper	270
F.9. MergeMapper	270
F.9.1. Examples	270
F.10. RegexpMapper	271
F.10.1. Examples	271
G. Core selectors	273

G.1. Contains	273
G.2. Date	274
G.3. Depend	275
G.4. Depth	275
G.5. Different	275
G.6. Filename	276
G.7. Present	277
G.8. Containsregexp	277
G.9. Size	278
G.10. Type	279
G.11. And	279
G.12. Majority	279
G.13. Modified	279
G.13.1. Parameters specified as nested elements	280
G.13.2. Examples	281
G.14. None	281
G.15. Not	281
G.16. Or	282
G.17. Readable	282
G.18. Writable	282
G.19. Executable	282
G.20. Selector	283
G.21. Symlink Selector	283
G.22. PosixPermissions Selector	283
H. Project Components	285
H.1. Phing Projects	285
H.1.1. Example	285
H.1.2.	285
H.1.3. Attributes	285
H.2. Targets and Extension-Points	286
H.2.1. Example	286
H.2.2. Attributes	286
H.2.3. Extension-Points	286
I. Loggers and Listeners	289
I.1. Listeners	289
I.2. Loggers	289
I.3. DefaultLogger	289
I.4. AnsiColorLogger	289
I.5. MailLogger	290
I.6. NoBannerLogger	291
I.7. ProfileLogger	291
I.8. StatisticsListener	291
I.9. TimestampedLogger	292
I.10. SilentLogger	292
I.11. MonologListener	292
I.12. DisguiseLogger	292
J. File Formats	293
J.1. Build File Format	293
J.2. Property File Format	294
Bibliography	297

Preface

PHing Is Not GNU 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, documentation generation (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.

Chapter 1. About this book

1.1. Contributors (present and past)

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

1.2. Copyright

Copyright 2002-2022, The Phing Project.

1.3. License

This documentation is made available under the GNU Free Document License (see Section A.6, “The GFDL License”)

```
Copyright (c) 2002 - 2022, The Phing Project
```

```
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.1 or  
any later version published by the Free Software Foundation;
```

1.4. DocBook

All Phing reference documentation is written using the DocBook5 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 required to extend or edit this documentation.

However, XML tends to be quite verbose and even if a plain text editor technically is all that is needed 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 is written is a valid DocBook5 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/` 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 1.4.4, "DocBook v5 elements used in the manual and their meaning"

As of this writing the build process has been validated using version 1.78.1 of the DocBook5 stylesheets.



Important

Make sure all documentation is written using UTF-8 text encoding.

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).



Tip

The easiest way to setup a complete build environment for DocBook5 for people new to DocBook is to install a clean version of Debian 7.x and then run the "`deb-setup.sh`" shell script. This will create a fully tested and working build environment for DocBook5 as it is used with Phing. This could easily be done using a virtual setup (for example using VirtualBox).

All DocBook is structured in a tree:

```

.
### source
#   ### appendixes
#   #   ### optionaltasks
#   ### 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 must be documented in a new file inside "source/appendixes/optionaltasks/", then a reference should be added in "source/appendix/optionaltasks.xml". 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/>]. By using the automated setup for Debian 7.x all these dependencies will be taken care of!

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. This currently contains two scripts, `deb-setup.sh` and `hlsaxon`. The first script helps to create a full build environment for DocBook5 starting with a clean Debian 7 installation. This is meant to help people new to DocBook5 to get a working build environment as easy as possible. This script takes care of all detailed setup and will make a fully working DocBook5 build environment out-of-the-box.

The second script (`hlsaxon`) is wrapper file used from the buildfiles to call the Saxon translator (a Java based XSL processor) with highlighting enabled and suitable paths to supporting libraries. In this script the path to the DocBook installed stylesheets must be adjusted depending on your system (unless the automated setup have been used - with the `deb-setup.sh` file which takes care of that setup automatically). *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

<code>all*</code>	Builds all available targets (default)
<code>chunk</code>	Builds the chunked HTML
<code>clean</code>	Removes all output files
<code>epub</code>	Builds the EPUB version
<code>hlhtml</code>	Builds the HTML version with syntax highlight
<code>hlpdf</code>	Builds the PDF version with syntax highlight
<code>html</code>	Builds the HTML version
<code>htmlfancy</code>	Builds the HTML version with an alternative styling for screen output
<code>pdf</code>	Builds the PDF version
<code>webhelp</code>	Builds the webhelp version (Note: This requires Java and Ant to be installed!)
<code>validate</code>	Validates all sources against the DocBook5 grammar

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. Template for new tasks

For creating documentation for new tasks the easiest thing is to use the included template `template_for_tasks.xml` which is a skeleton tasks with all commonly used elements. This will ensure a correct setting of all attributes. The skeleton can then be added to a suitable appendix as needed.



Note

All new task description should go into one of the Appendices.

1.4.3. Customization of the look & feel of the rendered outputs



Note

The following section is only meant for the maintainers that work on the core layout of the official Phing manual and is not necessary for developers adding documentation for new tasks of improving documentation for existing tasks.

Furthermore, by necessity this assumes a rudimentary knowledge of Docbook5 build process and what XSL and CSS stylesheets are. It is not possible in this short space to give a full description of that setup.

XSL Customization layer

All DocBook5 renderings are started from one of the customized XSL stylesheet under "stylesheets/xsl". All commonly adjusted properties should go into the appropriate stylesheet for that rendering. No properties should be passed on via the command line. To keep the customization layer as future proof as possible only in very rare circumstances should any cores XSL templates be copied and modified. As usual the recommended way is to use the provided hooks.

CSS stylesheets

The CSS stylesheets are used to create the look & feel for the HTML based renderings. These are entirely standard CSS files which by design are kept very simple. It should be noted that a few styling option depends in turn of the modified XSL transformations in the XSL customization layer. This had to be done in order to gain some more detailed control not provided by DocBook5 out-of-the-box.

Webhelp

The `webhelp` output rendering is a bit of a special case. This rendering depends not only on DocBook5 but also on Java as well as Ant build processor. These dependencies are inherited from the official DocBook5 webhelp process and will remain. Unfortunately adjusting the look & feel for this rendering is not as simple as for the other outputs since a fair amount of the layout (as well as look & feel) are hard-coded in the Webhelp build system. While it is perfectly possible to adjust the hard coded values and design choices it is not future proof. Since the Webhelp rendering is the newest and fastest improving output from DocBook the intention for the Phing documentation is to track these improvements and not spend time ourself to duplicate this effort with a parallel development.

1.4.4. 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 easier.

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 since 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 manual 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 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 describes 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 < ;

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">
  <append
    destFile="{process.outputfile}">
  <filterchain>
    <xsltfilter style="{process.stylesheet}">
      <param name="mode"
        expression="{process.xslt.mode}" />
    </xsltfilter>
  </append>
</programlisting>
```

```
</filterchain>
<filelist dir="book/"
listfile="book/PhingGuide.book" />
</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 elements. 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 warrants 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: "**PHing Is Not GNU 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.

In 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 its 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 run under PHP4.

Next came Phing 2.x, which required PHP5 (at least 5.2.x) and made 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. Phing became supported as a build tool in a number of various IDEs such as phpStorm, Netbeans 8.1 and the like. From versions 2.3.3, released on 7th December 2008, through to version 2.16 Phing has been available to install via PEAR.

In 2018 active work started on producing Phing 3.0 which requires PHP7.1 at a minimum. Phing 3.0 is only available through Composer or as a .phar archive and is no longer installable via the PEAR installer.

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 participating in the documentation is a great opportunity to get experience!

To get involved start by doing the following:

- Read this manual to understand Phing ;-)
- Visit the Phing website (<https://www.phing.info/>) [<https://www.phing.info/>]
- ...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 5.6 or above compiled `--with-libxml2`, as well as `--with-xml` 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 and up-to-date list of required and/or optional software and libraries, refer to the phing/phing [https://packagist.org/packages/phing/phing] package on Packagist.

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.

3.2.2. Composer Install

The preferred method to install Phing is through Composer [https://getcomposer.org/]. Add phing/phing [https://packagist.org/packages/phing/phing] to the `require-dev` or `require` of your project's `composer.json` configuration file, and run `composer install`:

```
{
  "require-dev": {
    "phing/phing": "3.*"
  }
}
```

3.2.3. Phar package

Download the Phar archive [<https://www.phing.info/get/phing-latest.phar>]. You do not need to execute any additional commands to install Phing, downloading the archive is enough. Phing can simply be started by running:

```
$ php phing-latest.phar [parameters ...]
```

3.2.4. Getting the latest source from Phing's Github 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. Running 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.

3.3.1. 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 multiple 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#Command-LineArguments](#)].

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
```

3.3.2. Supported command line arguments

The following command line arguments are supported

-h -help	print this message
-l -list	list available targets in this project
-i -init [file]	generates an initial buildfile
-v -version	print the version information and exit
-q -quiet	be extra quiet
-S -silent	print nothing but task outputs and build failures
-verbose	be extra verbose
-debug	print debugging information
-emacs, -e	produce logging information without adornments
-diagnostics	print diagnostics information
-strict	runs build in strict mode, considering a warning as error
-no-strict	runs build normally (overrides buildfile attribute)
-longtargets	show target descriptions during build
-logfile <file>	use given file for log
-logger <classname>	the class which is to perform logging
-listener <classname>	add an instance of class as a project listener
-f -buildfile <file>	use given buildfile
-D<property>=<value>	use value for given property
-keep-going, -k	execute all targets that do not depend on failed target(s)
-propertyfile <file>	load all properties from file
-propertyfileoverride	values in property file override existing values
-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" />
    </target>
</project>
```

```

    <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. This attribute controls the value of the <code>\${project.basedir}</code> property which can be used to reference files with paths relative to the project root folder. Can be a path relative to the position of the buildfile itself. If omitted, "." will be used, which means that the build file should be located in the project's root folder.	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
strict	Enables the strict-mode for the project build process.	No

See Section H.1, "Phing Projects" for a complete reference.

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 dependent 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. Looking at its depends attribute, 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-separated list of targets this target depends on.	No
if	The name of the <code>Property</code> that has to be set in order for this target to be executed	No
unless	The name of the <code>Property</code> that must not be set in order for this target to be executed.	No

See Section H.2, "Targets and Extension-Points" for a complete reference.

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 property task, 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`.

Getting the value of a Reference with `${toString:}` Any Phing type item which has been declared with a reference can also its string value extracted by using the `${toString:}` operation, with the name of the reference listed after the `toString:` text. The `__toString()` method of the php class instance that is referenced is invoked all built in types strive to produce useful and relevant output in such an instance.

For example, here is how to get a listing of the files in a fileset:

```
<fileset id = "sourcefiles" dir = "src" includes = "**/*.php"/>
<echo> sourcefiles = ${toString:sourcefiles} </echo>
```

There is no guarantee that external types provide meaningful information in such a situation

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>
</project>
```

```

</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 four tags within the `project` tag define properties. They appear in two possible variants:

- The first `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 J, *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.69, “`UpToDateTask`” for examples on how this task is 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 **L**anguage 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 and Extension-Points” 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\Filter\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\Filter\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 = "regex" 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\Filter\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 condition, if and waitfor 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. xor

The `<xor>` element performs an exclusive or on all nested elements, similar to the `^` operator in PHP. It only evaluates to true if an odd number of nested conditions are true. There is no shortcutting of eval-

uation, unlike the `<and>` and `<or>` tests. It doesn't have any attributes and accepts all nested elements of the condition task as nested elements as well.

5.8.5. 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)

Note: machines running OSX match on the `mac` and `unix` families! To test for Macs that don't run a Unix-like OS, use the following code:

```
<condition property = "isMacOsButNotMacOsX">
  <and>
    <os family = "mac" />
    <not>
      <os family = "unix" />
    </not>
  </and>
</condition>
```

5.8.6. 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 comparison. Default isNo true.	No
trim	Trim whitespace from arguments before comparingNo them. Default is false.	No

5.8.7. versioncompare

Compares two given versions

Table 5.3: versioncompare Attributes

Attribute	Description	Required
version	The version you want to compare	Yes

Attribute	Description	Required
desiredVersion	The version you want to compare against	Yes
operator	The operator to use for version comparison. Default is >=.	No
debug	Turns on debug mode, that echoes the comparison message. Default is false.	No

```
<versioncompare version = "${aProperty}" desiredVersion = "1.3" operator = "gt" />
```

This condition internally uses PHP `version_compare()`. Operators and behavior are the same.

5.8.8. http

Condition to wait for a HTTP request to succeed.

Attributes are:

- url - the URL of the request.
- errorsBeginAt - number at which errors begin at.
- quiet - Set quiet mode, which suppresses warnings and errors.

Table 5.4: http Attributes

Attribute	Description	Required
url	The URL of the request.	Yes
errorsBeginAt	Number at which errors begin at. - Default: 400	No
requestMethod	Sets the method to be used when issuing the HTTP request. - Default: GET	No
followRedirects	Whether redirects sent by the server should be followed. - Default: true	No
quiet	Set quiet mode, which suppresses warnings and errors. Default is false	No

```
<http url = "http://url.to.test" errorsBeginAt = "404" />
```

5.8.9. PDOSQLException

`PDOSQLExceptionTask` can also be used as condition. Returns `true` when the connection to a database succeeds, and `false` otherwise. This condition requires the PDO extension [<https://www.php.net/manual/en/book.pdo.php>] to work properly.

Table 5.5: PDOSQLException condition attributes

Attribute	Description	Required
url	The PDO Data Source Name (DSN).	Yes
userid	The username for current DSN.	No
password	The password for current DSN.	No

This is a typical use case for `PDOSQLExec` condition:

```
<target name = "wait-for-mysql">
  <waitfor timeoutproperty = "mysql.timeout" maxwait = "60" maxwaitunit = "second">
    <pdosqlexec url = "mysql:host=localhost;port=3306"
      userid = "${db.username}"
      password = "${db.password}" />
  </waitfor>
  <fail if = "mysql.timeout">Cannot reach database</fail>
</target>
```

If you also want to check if a specific schema exists, you can include the schema's name in your url:

```
<pdosqlexec url = "mysql:host=127.0.0.1;port=3306;dbname=foo"
  userid = "${db.username}"
  password = "${db.password}" />
```

This condition uses PDO behind the scenes. Therefore, if you have installed the appropriate driver you should also be able to reach many other DBMS [<https://www.php.net/manual/en/pdo.drivers.php>]. For example, for a PostgreSQL database:

```
<pdosqlexec url = "pgsql:host=localhost;port=5432;dbname=bar"
  userid = "${db.username}"
  password = "${db.password}" />
```

You should never hard-code sensitive data in your buildfile, you could use an unversioned property file instead. Also, be careful when using verbose or debug mode since you can expose sensitive data.

5.8.10. socket

Condition to test for a (tcp) listener on a specified host and port.

Table 5.6: *socket Attributes*

Attribute	Description	Required
server	The hostname or ip address of the server.	Yes
port	The port number of the server.	Yes

```
<socket server = "localhost" port = "80" />
```

5.8.11. hasfreespace

Condition returns true if selected partition has the requested space, false otherwise.

Table 5.7: *hasfreespace Attributes*

Attribute	Description	Required
partition	Absolute path to the partition/device to check.	Yes
needed	The amount of free space required. Examples: Yes 250M, 10G, 1T.	



Note

File size can be written using IEC and SI suffixes, bytes are assumed when suffix is not specified. The following suffixes (case-insensitive) are supported:

Table 5.8: Supported file size suffixes

Standard	Suffixes	Equivalence
IEC	B.	1 byte
	K, Ki, KiB, kibi, kibibyte.	1024 bytes
	M, Mi, MiB, mebi, mebibyte.	1024 kibibytes
	G, Gi, GiB, gibi, gibibyte.	1024 mebibytes
	T, Ti, TiB, tebi, tebibyte.	1024 gibibytes
SI	kB, kilo, kilobyte.	1000 bytes
	MB, mega, megabyte.	1000 kilobytes
	GB, giga, gigabyte.	1000 megabytes
	TB, tera, terabyte.	1000 gigabytes

On Unix-like platforms:

```
<hasfreespace partition = "/" needed = "250M" />
```

On Windows:

```
<hasfreespace partition = "c:" needed = "10M" />
```

This condition internally uses PHP `disk_free_space()`.

5.8.12. isset

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

Table 5.9: *isset* Attributes

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

5.8.13. contains

Tests whether a string contains another one.

Table 5.10: *contains* Attributes

Attribute	Description	Required
string	The string to search in.	Yes
substring	The string to search for.	Yes
casesensitive	Perform a case sensitive comparison. Default isNo true.	

5.8.14. istrue

Tests whether a string evaluates to true.

Table 5.11: *isttrue Attributes*

Attribute	Description	Required
value	value to test	Yes

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

5.8.15. isfalse

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

Table 5.12: *isfalse Attributes*

Attribute	Description	Required
value	value to test	Yes

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

5.8.16. ispropertytrue

Tests whether a property evaluates to true.

Table 5.13: *ispropertytrue Attributes*

Attribute	Description	Required
property	property to test	Yes

```
<ispropertytrue property = "someproperty" />
```

5.8.17. ispropertyfalse

Tests whether a property evaluates to not true, the negation of <ispropertytrue>

Table 5.14: *ispropertyfalse Attributes*

Attribute	Description	Required
property	property name to test	Yes

```
<ispropertyfalse property = "someproperty" />
```

5.8.18. referenceexists

Tests whether a specified reference exists.

Table 5.15: *referenceexists Attributes*

Attribute	Description	Required
ref	reference to test for	Yes

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

5.8.19. available

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

```
<if>
  <available file = "README.md" />
  <then>
    <echo message = "Please read README.md" />
  </then>
</if>
```

5.8.20. filematch

Test two files for matching. Nonexistence of one file results in "false", although if neither exists they are considered equal in terms of content. This test does a byte for byte comparison, so test time scales with byte size. NB: if the files are different sizes, one of them is missing or the filenames match the answer is so obvious the detailed test is omitted.

Table 5.16: filematch Attributes

Attribute	Description	Required
file1	First file to test.	Yes
file2	Second file to test.	Yes

```
<filematch file1 = "${file1}" file2 = "${file2}" />
```

5.8.21. isfileselected

Test whether a file passes an embedded selector.

Table 5.17: isfileselected Attributes

Attribute	Description	Required
file	The file to check if it passes the embedded selector.	Yes
basedir	The base directory to use for name based selectors. If this is not set, the project's basedirectory will be used.	No

```
<isfileselected file = "a.xml">
  <date datetime = "06/28/2000 2:02 pm" when = "equal" />
</isfileselected>
```

5.8.22. isfailure

Test the return code of an executable for failure.

Table 5.18: isfailure Attributes

Attribute	Description	Required
code	The return code to test.	Yes

```
<exec command = "test" returnProperty = "return.code" />
```

```
<if>
  <isfailure code = "${return.code}"/>
  <then><echo msg = "${return.code}"/></then>
</if>
```

5.8.23. matches

Test if the specified string matches the specified regular expression pattern.

Table 5.19: *matches Attributes*

Attribute	Description	Required
string	The string to test.	Yes
pattern	The regular expression pattern used to test.	Yes
casesensitive	Perform a case sensitive match. Default is true.	No
multiline	Perform a multi line match. Default is false.	No
modifiers	The regular expression modifiers used to test.	No

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 has 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 F, *Core mappers*. 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 in the source tree of Phing before starting to code. 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, which 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 explicit 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

These scripts are technically 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 scripts 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. `RootHandler` 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

use Phing\Task;

class MyEchoTask extends Task {

    /**
     * The message passed in the buildfile.
     */
    private $message = null;
    /**
     * Whether to reverse the message, for fun?
     */
    private $reverse = false;

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

    public function setReverse($str) {
        $this->reverse = StringHelper::booleanValue($str);
    }

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

    /**
     * The main entry point method.
     */
}
```

```

public function main() {
    if ($this->reverse) {
        print(strrev($this->message));
    } else {
        print($this->message);
    }
}
}
?>

```

This code contains a rather simple, but complete Phing task. It is assumed that the file is named `MyEchoTask.php`. For this example, we're assuming that the file is placed in `/home/example/classes`. 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 loaded and 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">
    <includepath classpath = "/home/example/classes" />
    <taskdef name = "myecho" classname = "MyEchoTask" />

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

```

To register the custom task with Phing, the `taskdef` element (line 5) is used. See Section B.60, "TaskdefTask" for a more detailed explanation. Optionally, before the `taskdef` element, the `includepath` element adds a path to PHP's include path. This is of course only required if the mentioned path isn't already on the include path. See Section B.30, "IncludePathTask" for a more detailed explanation.

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".

And for fun, if the "reverse" attribute is set to a "truth-like" value, the message will be reversed when displayed. So we get "dlroW olleH" displayed instead!

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 yourself:

- 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 our 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.

There is also another setter named `setReverse`. This uses the `StringHelper::toBoolean` static function to convert truthy values to a true/false value. This helps keep our own code nice and simple.

6.6.10. Creator Methods

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

For example, you might be developing a task that would contain a nested "color" XML tag. In this instance a creator method named `createcolor` would be required.

```
<tag>
  <color red = "..." green = "..." blue = "...">
</tag>
```

If the XML for the task and the subtag look like the above, the PHP code for it could look something like the following:

```
class TagTask extends Task
{
    protected $colors = array();

    public function createColor()
    {
        $colorObj = new TagColor();
        $this->colors[] = $colorObj;
        return $colorObj;
    }
}

class TagColor
{
    public function setRed($value)
    {
    }

    public function setGreen($value)
```

```
{  
  
    public function setBlue($value)  
    {  
    }  
}
```

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 `RegularExpression` 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\Mapper\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\Mapper\GlobMapper` or `Phing\Mapper\RegexpMapper`.

```
use Phing\Mapper\FileNameMapper;

/**
 * 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 $sourceFilename The name to be converted.
     * @return array The matched filenames.
     */
    public function main($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`, then you would refer to it like this in your build file:

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

6.9. Writing Selectors

Custom selectors are datatypes that implement `Phing\Type\Selector\FileSelector`.

There is only one method required, public function `isSelected(PhingFile $basedir, string $filename, PhingFile $file): bool`. It returns true or false depending on whether the given file should be selected or not.

An example of a custom selection that selects filenames ending in `.php` would be:

```
class PhpSelector implements FileSelector
{
    public function isSelected(PhingFile $b, string $filename, PhingFile $f)
    {
        return StringHelper::endsWith('.php', strtolower($filename));
    }
}
```

Adding the selector to the system is achieved as follows:

```
<typedef
    name = "phpselector"
    classname = "PhpSelector" />
```

This selector can now be used wherever a Core Phing selector is used, for example:

```
<copy todir = "to">
    <fileset dir = "src">
        <phpselector/>
    </fileset>
</copy>
```

6.10. Writing Conditions

Custom conditions are datatypes that implement `Phing\Task\System\Condition\Condition`. For example a custom condition that returns true if a string is all upper case could be written as:

```
class AllUpperCaseCondition implements Condition
{
    private $value;

    // The setter for the "value" attribute
    public function setValue(string $value)
    {
        $this->value = $value;
    }

    // This method evaluates the condition
    public function evaluate()
    {
        if ($this->value === null) {
            throw new BuildException("value attribute is not set");
        }
    }
}
```

```
        return strtoupper($this->value) === $this->value;
    }
}
```

Adding the condition to the system is achieved as follows:

```
<typedef
    name = "alluppercase"
    classname = "AllUpperCaseCondition"/>
```

This condition can now be used wherever a Core Phing condition is used.

```
<condition property = "allupper">
    <alluppercase value = "THIS IS ALL UPPER CASE"/>
</condition>
```

Appendix A. Fact Sheet

A.1. Built-In Properties

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

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

A.2. Command Line Arguments

The following table lists the command line arguments currently available.

Table A.2: Phing Command Line Arguments

Parameter	Meaning
<code>-h -help</code>	Display the help screen
<code>-l -list</code>	List all available targets in buildfile (excluding targets that have their <code>hidden</code> attribute set to <code>true</code>)
<code>-i -init [file]</code>	Generates an initial buildfile at Phing's start directory. Optionally you can specify buildfile's location and name.
<code>-v -version</code>	Print version information and exit
<code>-q -quiet</code>	Quiet operation, no output at all
<code>-S -silent</code>	Print nothing but task outputs and build failures
<code>-verbose</code>	Verbose, give some more output
<code>-debug</code>	Output debug information
<code>-emacs -e</code>	Produce logging information without adornments
<code>-diagnostics</code>	Print diagnostics information
<code>-longtargets</code>	Show target descriptions during build
<code>-logfile <file></code>	Use given file for log
<code>-logger path.to.Logger</code>	Specify an alternate logger. Default is <code>Phing\Listener\AnsiColorLogger</code> . Other options include <code>Phing\Listener\NoBannerLogger</code> , <code>Phing\Listener\DefaultLogger</code> , <code>Phing\Listener\XmlLogger</code> , <code>Phing\Listener\TargetLogger</code> and <code>Phing\Listener\HtmlColorLogger</code> .
<code>-f -buildfile <file></code>	Specify an alternate buildfile name. Default is <code>build.xml</code>
<code>-D<property>=<value></code>	Set the property to the specified value to be used in the buildfile
<code>-keep-going -k</code>	Execute all targets that to not depend on failed target(s)
<code>-propertyfile <file></code>	Load properties from the specified file
<code>-find <file></code>	Search for a buildfile towards the root of the filesystem and use that
<code>-inputhandler <file></code>	The class to use to handle user input. Default is <code>\Phing\Input\ConsoleInputHandler</code> . Other options are <code>\Phing\Input\NoInteractionInputHandler</code> or an own implementation of <code>InputHandler</code> .

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.

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

```
<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. Secondly, 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.output-file}`. This is a real example from the build file used to generate this book!



Note

By default, whitespace is stripped from text that is appended to a file (matching the way the Concat task works in Ant). This is because adding a nested text element may introduce additional (ignorable) whitespace. If you want to override this behavior, set the `skipsanitize` attribute to `true`.

Table B.3: Attributes

Name	Type	Description	Default	Required
<code>destFile</code>	File	Path of file to which text should be appended. If not specified the console will be used instead.	n/a	No
<code>append</code>	String	Specifies whether or not the file specified by 'destfile' should be appended. Defaults to "yes".	yes	No
<code>overwrite</code>	Boolean	Specifies whether or not the file specified by 'destfile' should be written to even if it is newer than all source files.	yes	No

Name	Type	Description	Default	Required
fixlastline	Boolean	Specifies whether or not to check if each file concatenated is terminated by a new line. If this attribute is "yes" a new line will be appended to the stream if the file did not end in a new line. This attribute does not apply to embedded text.		No
eol	String	<p>Specifies what the end of line character are for/n/a use by the fixlastline attribute. Valid values for this property are:</p> <ul style="list-style-type: none"> • cr: a single CR • lf: a single LF • crlf: the pair CRLF • mac: a single CR • unix: a single LF • dos: the pair CRLF <p>The default is platform dependent. For Unix platforms, the default is "lf". For DOS based systems (including Windows), the default is "crlf". For Mac OS, the default is "cr".</p>		No
skipsanitize	Boolean	Specifies whether to skip sanitizing text (i.e., n/a stripping spaces and newlines).		No
file	File	Path to file that should be appended to dest-n/a File.		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
- path

- `header`, `footer` Used to prepend or postpend text into the concatenated stream. The text may be in-line or be in a file.

Table B.4: Attributes

Name	Type	Description	Default	Required
<code>filtering</code>	Boolean	Whether to filter the text provided by this suby element.		No
<code>file</code>	String	A file to place at the head or tail of the concatenated text.	n/a	No
<code>trim</code>	Boolean	Whether to trim the value.	no	No
<code>trimleading</code>	Boolean	Whether to trim leading white space on each line.	no	No

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, FileLists or DirSets – are passed as arguments to the system command.

Table B.5: Attributes

Name	Type	Description	Default	Required	Alias
<code>executable</code>	String	The command to execute without any command line arguments.	n/a	Yes	
<code>dir</code>	String	The directory the command is to be executed in.	n/a	No	
<code>output</code>	String	Where to direct stdout.	n/a	No	
<code>error</code>	String	Where to direct stderr.	n/a	No	
<code>os</code>	String	Only execute if the Appendix A, <i>Fact Sheet</i> property contains specified text.	n/a	No	
<code>escape</code>	Boolean	Escape shell metacharacters before execution. Setting this to true will enable the escape precaution.	false	No	
<code>passthru</code>	Boolean	Whether to use PHP's <code>passthru()</code> function instead of <code>exec()</code> .	false	No	
<code>spawn</code>	Boolean	Whether to spawn unix programs to the background, redirecting stdout (output will not be logged by Phing).	false	No	
<code>returnProperty</code>	String	Property name to set return value from the execution.	n/a	No	
<code>outputProperty</code>	String	Property name to set output value from the execution.	n/a	No	

Name	Type	Description	Default	Required	Alias
checkreturn	Boolean	Whether to check the return code off 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, append-ing all files as arguments. If false, command will be executed once for every file.	false	No	
addsource-file	Boolean	Whether source file name(s) should be added to the end of command-line automatically. If you need to place it somewhere different, use a nested <srcfile> element between your <arg> elements to mark the insertion point.	true	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	Integer	Limit the amount of parallelism by passing at most this many sourcefiles at once. Set it to <= 0 for unlimited.		No	
skipempty-filesets	Boolean	Don't run the command, if no sourcefiles have been found or are newer than their corresponding target files. Despite its name, this attribute applies to filelists as well.	false	No	
type	String	One of file, dir or both. If set to file, only the names of plain files will be sent to the command. If set to dir, only the names of directories are considered. Note: The type attribute does not apply to nested dirsets - dirsets always implicitly assume type to be dir.	file	No	
force	Boolean	Whether to bypass timestamp comparisons for target files.	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.6: Attributes

Name	Type	Description	Default	Required
value	String	A single command-line argument; can contain space characters.		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.		
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 Phing will convert it to the platform's local conventions.		
line	String	A space-delimited list of command-line arguments.		

- fileset
- filelist
- dirset
- mapper
- srcfile
- targetfile

B.5. AttribTask

Changes the attributes of a file or all files inside specified directories. Right now it has effect only under Windows. Each of the 4 possible permissions has its own attribute, matching the arguments for the attrib command.

FileSets or FileLists can be specified using nested `fileset` and `filelist` elements.

By default this task won't do anything unless it detects it is running on a Windows system. If you know for sure that you have a "attrib" executable on your PATH that is command line compatible with the Windows command, you can use the task's `os` attribute and set its value to your current os.

Table B.7: Attributes

Name	Type	Description	Default	Required
file	String	The file or directory of which the permissions must be changed.	n/a	Yes
readonly	Boolean	The readonly permission.	n/a	at least one of the four.
archive	Boolean	The archive permission.	n/a	
system	Boolean	The system permission.	n/a	
hidden	Boolean	The hidden permission.	n/a	
verbose	Boolean	Whether to print a summary after execution or false not. Defaults to false.		No
os	String	List of Operating Systems on which the command may be executed.	n/a	No

B.5.1. Example

```
<attrib file = "${dist}/run.bat" readonly = "true" hidden = "true"/>
```

makes the "run.bat" file read-only and hidden.

```
<attrib readonly = "false">
  <fileset dir = "${meta.inf}" includes = "**/*.xml"/>
</attrib>
```

makes all ".xml" files below \${meta.inf} readable.

B.5.2. Supported Nested Tags

- filelist
- fileset

B.6. Augment

Modify an existing reference by adding nested elements or (re-)assigning properties mapped as XML attributes. This is an unusual task that makes use of Phing's internal processing mechanisms to reload a previously declared reference by means of the id attribute, then treats the declared augment element as though it were the original element.

Table B.8: Attributes

Name	Type	Description	Default	Required
id	String	The id of the reference to augment. If no such reference has been declared a BuildException is thrown.	n/a	Yes

B.6.1. Examples

Given

```
<fileset id = "input-fs" dir = "${project.basedir}"/>
```

invocation

```
<augment id = "input-fs" excludes = "foo"/>
```

modifies the excludes attribute of input-fs, whereas

```
<augment id = "input-fs">
  <filename name = "bar"/>
</augment>
```

adds a filename selector to input-fs.

B.7. AutoloaderTask

The AutoloaderTask includes autoload file to bootstrap all necessary components in Phing execution context. It could be useful if build tools (e.g. phpunit, phloc etc.) are installed as Composer dependencies.

Table B.9: Attributes

Name	Type	Description	Default	Required
autoloaderpath	String	Path to autoloader file	vendor/autoload.php	Yes

B.7.1. Example

```
<autoloader autoloaderpath = "foo/autoload.php"/>
```

B.8. 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 existence 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).

NB: the Available task can also be used as a condition, see conditions.

Table B.10: Attributes

Name	Type	Description	Default	Required
property	String	Name of the property that is to be set.	n/a	Yes

Name	Type	Description	Default	Required
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 lookn/a for a file or a directory at the position set by file. If empty, it checks for either file or directory.		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.8.1. 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.9. Basename

Task to determine the basename of a specified file, optionally minus a specified suffix.

When this task executes, it will set the specified property to the value of the last path element of the specified file. If file is a directory, the basename will be the last directory element. If file is a full-path, relative-path, or simple filename, the basename will be the simple file name, without any directory elements.

Table B.11: Attributes

Name	Type	Description	Default	Required
property	String	Name of the property that is to be set.	n/a	Yes
file	String	The path to take the basename of.	n/a	Yes
suffix	String	The suffix to remove from the resulting base-n/a name (specified either with or without the ".").		No

B.9.1. Examples

```
<basename property = "cmdname" file = "./foo.exe" suffix = ".exe"/>
```

B.10. Bindtargets

Make some target the extension of some defined extension point. It will make the list of targets dependencies of the extension point.

This target is useful when you want to have a target to participate in another build workflow which explicitly exposes an extension point for that kind of insertion. Thus the target to bind and the extension point to bind to are both declared in some imported build files. But directly modifying the target dependency graph of these external build files may have a side effect on some other project which imports them. This task helps to modify the target dependencies but only in your context.

Table B.12: Attributes

Name	Type	Description	Default	Required
targets	String	a comma separated list of target names to/a bind.	ton/a	Yes
extensionPoint	String	the name of the extension point to bind the target to.	tar-n/a	Yes
onMissingExtensionPoint	String	What to do if this target tries to extend a missing extension-point: "fail", "warn", "ignore".	fail	No

B.10.1. Examples

```
<bindtargets targets = "build-phar,build-src-phar" extensionPoint = "dist" />
```

B.11. ChmodTask

Sets the mode of a file or directory.

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

Table B.13: 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'.	inn/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.11.1. Examples

```
<chmod file = "test.txt" mode = "0755" />
```

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

B.11.2. Supported Nested Tags

- fileset
- dirset

B.12. ChownTask

Changes the owner of a file or directory.

Table B.14: Attributes

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

B.12.1. 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.12.2. Supported Nested Tags

- fileset
- dirset

B.13. ConditionTask

Sets a property if a certain condition holds true - this is a generalization of Section B.8, “AvailableTask” and Section B.69, “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 conditions for a complete list of nested elements.

Table B.15: Attributes

Name	Type	Description	Default	Required
<code>property</code>	String	The name of the property to set.	n/a	Yes
<code>value</code>	String	The value to set the property to. Defaults to <code>true</code> "true".		No
<code>else</code>	String	The value to set the property to if the condition evaluates to false. By default the property will remain unset.	n/a	No

B.13.1. Examples

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

B.13.2. Supported Nested Tags

- `or`
- `and`

B.14. 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 explicitly overwrite existing files.

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

Table B.16: Attributes

Name	Type	Description	Default	Required
<code>file</code>	String	The source file.	Yes	
<code>tofile</code>	String	The destination the file is to be written to. <code>tofile</code> specifies a full filename. If you only want to specify a directory to copy to, use <code>todir</code> .	n/a	Yes (or <code>todir</code>)
		Either this or the <code>todir</code> attribute is required.		
<code>todir</code>	String	The directory the file is to be copied to. Then file will have the same name of the source file. If you want to specify a different name, use <code>tofile</code> . The directory must exist.		Yes (or <code>tofile</code>)

Name	Type	Description	Default	Required
overwrite	Boolean	Overwrite existing files even if the destination files are newer.	false	No
tstamp or preserve-lastmodified	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
flatten	Boolean	Ignore the directory structure of the source files, and copy all files into the directory specified by the todir attribute. Note that you can achieve the same effect by using a flatten mapper.	false	No
verbose	Boolean	Whether to print the list of the copied file. Defaults to false.	false	No
granularity	Integer	The number of seconds leeway to give before deciding a file is out of date. This can also be useful if source and target files live on separate machines with clocks being out of sync.	0	No



Note

No automatic expansion 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.14.1. 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.14.2. Supported Nested Tags

- fileset
- filelist
- dirset
- filterchain
- mapper

B.15. DefaultExcludes

Alters the default excludes for all subsequent processing in the build, and prints out the current default excludes if desired.

Table B.17: Attributes

Name	Type	Description	Default	Required
echo	Boolean	whether or not to print out the default excludes	false	attribute "true" required if no other attribute specified
default	Boolean	go back to hard wired default excludes	n/a	attribute "true" required if no other attribute specified
add	String	the pattern to add to the default excludes	n/a	if no other attribute is specified
remove	String	remove the specified pattern from the default excludes	n/a	if no other attribute is specified

B.15.1. Examples

Print out the default excludes

```
<defaultexcludes echo = "true"/>
```

Print out the default excludes and exclude all *.bak files in all further processing

```
<defaultexcludes echo = "true" add = "**/*.bak" />
```

Silently allow several fileset based tasks to operate on emacs backup files and then restore normal behavior

```
<defaultexcludes remove = "**/*~" />

(do several fileset based tasks here)

<defaultexcludes default = "true" />
```

B.16. 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.18: Attributes

Name	Type	Description	Default	Required
file	String	The file that is to be deleted. You either haven/a to specify this attribute, dir, or use a fileset.		Yes (or dir)
dir	String	The directory that is to be deleted. You eithern/a have to specify this attribute, file, or use a fileset.		Yes (or file)
verbose	Boolean	Used to force listing of all names of deletedn/a files.		No
quiet	Boolean	If the file does not exist, do not display a di-n/a agnostic 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.		No
		This setting emulates the -f option to the Unixrmcommand. Default is false meaning things are verbose		
failonerror	Boolean	If this attribute is set to true, DeleteTask willfalse verbose on errors but the build process will not be stopped.	false	No
includeempty-dirs	Boolean	Determines if empty directories are also to befalse deleted.	false	No

B.16.1. Examples

```
<!-- Delete a specific file -->
<delete file = "/tmp/foo.bar" />

<!-- Delete a directory -->
<delete dir = "/tmp/dar1" includeemptydirs = "true" verbose = "true" failonerror = "true" />

<!-- Delete using a fileset -->
```

```
<delete>
  <fileset dir = "/tmp">
    <include name = "*.bar" />
  </fileset>
</delete>
```

B.16.2. Supported Nested Tags

- fileset
- filelist
- dirset

B.17. DependSet

The dependset task compares a set of sources with a set of target files. If any of the sources has been modified more recently than any of the target files, all of the target files are removed.

B.17.1. Examples

```
<dependset>
  <srcfilelist
    dir    = "${dtd.dir}"
    files  = "paper.dtd,common.dtd" />
  <srcfilelist
    dir    = "${xsl.dir}"
    files  = "common.xsl" />
  <srcfilelist
    dir    = "${basedir}"
    files  = "build.xml" />
  <targetfileset
    dir    = "${output.dir}"
    includes = "**/*.html" />
</dependset>
```

In this example derived HTML files in the `${output.dir}` directory will be removed if any are out-of-date with respect to:

- the DTD of their source XML files
- a common DTD (imported by the main DTD)
- a subordinate XSLT stylesheet (imported by the main stylesheet), or
- the buildfile

If any of the sources in the above example does not exist, all target files will also be removed. To ignore missing sources instead, use filesets instead of filelists for the sources.

B.17.2. Supported Nested Tags

- srcfileset
- srcfilelist
- targetfileset

- `targetfilelist`

B.18. Diagnostics

Runs phing's `-diagnostics` code inside phing itself. This is good for debugging phing's configuration under an IDE.

B.18.1. Example

```
<target name = "diagnostics" description = "diagnostics">
  <diagnostics/>
</target>
```

B.19. Dirname

Task to determine the directory path of a specified file.

When this task executes, it will set the specified property to the value of the specified file (or directory) up to, but not including, the last path element. If the specified file is a path that ends in a filename, the filename will be dropped. If the specified file is just a filename, the directory will be the current directory.

Note: This is not the same as the UNIX `dirname` command, which is defined as "strip non-directory suffix from filename". `<dirname>` determines the full directory path of the specified file.

Table B.19: Attributes

Name	Type	Description	Default	Required
<code>file</code>	String	The path to take the <code>dirname</code> of.	n/a	yes
<code>property</code>	String	The name of the property to set.	n/a	yes

B.19.1. Example

```
<dirname property = "foo.dirname" file = "foo.txt"/>
```

will set `foo.dirname` to the project's `basedir`.

B.20. EchoPropertiesTask

Displays all the current properties in the project. The output can be sent to a file if desired. This task can be used as a somewhat contrived means of returning data from an `<phing>` invocation, but is really for debugging build files.

Table B.20: Attributes

Name	Type	Description	Default	Required
<code>destfile</code>	String	If specified, the value indicates the name of the file to send the output of the statement to.	n/a	no

Name	Type	Description	Default	Required
		If not specified, then the output will go to the Phing log.		
srcfile	String	If specified, the value indicates the name of the property file to read from. If not specified, then the system properties will be taken.		no
prefix	String	a prefix which is used to filter the properties; only those properties starting with this prefix will be echoed.		no
regex	String	a regular expression which is used to filter the properties; only those properties whose names match it will be echoed.		no
failonerror	Boolean	By default, the "failonerror" attribute is enabled. If an error occurs while writing the properties to a file, and this attribute is enabled, then a BuildException will be thrown, causing the build to fail. If disabled, then IO errors will be reported as a log statement, and the build will continue without failure from this task.		no
format	String	One of text or xml. Determines the output format. Defaults to text.		no

B.20.1. Example

```
<echoproperties />
```

Report the current properties to the log.

```
<echoproperties destfile = "my.properties"/>
```

Report the current properties to the file "my.properties", and will fail the build if the file could not be created or written to.

```
<echoproperties destfile = "my.properties" failonerror = "false"/>
```

Report the current properties to the file "my.properties", and will log a message if the file could not be created or written to, but will still allow the build to continue.

```
<echoproperties prefix = "phing."/>
```

List all properties beginning with "phing."

```
<echoproperties regex = ".*phing.*"/>
```

Lists all properties that contain "phing" in their names.

B.21. 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.21: 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 re-info ported. One of error, warning, info, ver- bose, debug.		No

B.21.1. 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>
```

Echo a previously defined fileset element.

```
<fileset dir = "./tests" id = "test.files">
  <include name = "**/*Test.php" />
</fileset>

<echo>
  <fileset refid = "test.files"/>
</echo>
```

B.21.2. Supported Nested Tags

- fileset

B.22. EchoXML

Echo nested XML to the console or a file.

Table B.22: Attributes

Name	Type	Description	Default	Required
file	String	The file to receive the XML.	by de-No fault nest- ed XML is	

Parameters specified as nested elements

Name	Type	Description	Default	Required
append	Boolean	Whether to append file, if specified.	false	No

B.22.1. Parameters specified as nested elements

Nested XML content is required.

B.22.2. Examples

```
<echoxml file = "subbuild.xml">
  <project default = "foo">
    <target name = "foo">
      <echo>foo</echo>
    </target>
  </project>
</echoxml>
```

Create a Phing buildfile, subbuild.xml.

B.23. 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.

The `command` attribute is no longer supported. You should now use a combination of the `executable` attribute and `arg` nested elements:

```
<exec command = "echo foo"/>
<!-- should become -->
<exec executable = "/bin/echo">
  <arg value = "foo"/>
</exec>
```

Where it was once possible to pipe the output of one program to be the input of another using the `command` attribute: `<exec command="echo FLUSHALL | redis-cli">` This must now be done using a combination of the **executable and line attributes**, thus: `<exec executable="bash" line="echo FLUSHALL | redis-cli">`

Table B.23: Attributes

Name	Type	Description	Default	Required
executable	String	The command to execute without any command line arguments.		
dir	String	The directory the command is to be executed in.		No
output	String	Where to direct stdout.	n/a	No
error	String	Where to direct stderr.	Redirected to stdout, un-	No

Name	Type	Description	Default	Required
			less passthru is set to true.	
os	String	Only execute if the Appendix A, <i>Fact Sheet</i> n/a property contains specified text.	n/a	No
osfamily	String	OS family as used in the <os> condition.	n/a	No
escape	Boolean	By default, we escape shell metacharacters before executing. Setting this to false will disable this precaution.	false	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
resolveexecutable	Boolean	When this attribute is true, the name of the executable is resolved firstly against the project basedir and if that does not exist, against the execution directory if specified. On Unix systems, if you only want to allow execution of commands in the user's path, set this to false.	false	No
searchpath	Boolean	When this attribute is true, then system path environment variables will be searched when resolving the location of the executable.	false	No

B.23.1. Examples

```

<!-- List the contents of "/home" using the executable attribute -->
<exec executable = "ls" passthru = "yes">
  <arg value = "-l"/>
  <arg path = "/home"/>
</>

<!-- List the contents of "/home", but only if on Linux -->
<exec executable = "ls" passthru = "yes" os = "Linux">
  <arg value = "-l"/>
  <arg path = "/home"/>
</>

```

```

<!-- Demonstrate executable attribute and environment variables. -->
<exec executable = "php" outputProperty = "outputProperty">
  <env key = "HELLO" value = "hello"/>
  <env key = "WORLD" value = "world"/>
  <arg value = "-r"/>
  <arg value = "print getEnv('HELLO') . ' ' . getEnv('WORLD');"/>
</exec>

<!-- Demonstrate piping outputs from one command to another using the executable attribute. -->
<exec executable = "bash">
  <arg value = "-c"/>
  <arg line = '"java -jar test.jar page.xml | mysql -u user -p base'"/>
</exec>

<!-- Restart some docker service -->
<exec executable = "docker">
  <arg line = "--debug restart ${service.name}"/>
</exec>

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

```

B.23.2. Supported Nested Tags

- arg

Table B.24: 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 Phing will convert it to the platform's local conventions.	n/a	
line	String	A space-delimited list of command-line arguments.	n/a	
escape	Boolean	Force escape for this attribute.	false	

env

It is possible to specify environment variables to pass to the system command via nested `<env>` elements.

Table B.25: Attributes

Name	Type	Description	Default	Required
key	String	The name of the environment variable.	n/a	Yes
value	String	The literal value for the environment variable.	n/a	One of these

Name	Type	Description	Default	Required
file	String	The value for the environment variable. Will be replaced by the absolute filename of the file by Phing.		
path	String	The value for a PATH like environment variable. You can use ; or : as path separators and Phing will convert it to the platform's local conventions.		

B.24. FailTask

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

Table B.26: 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.		No
unless	String	Name of property that must not be set in order for script to exit.		No
status	Integer	Exit using the specified status code; assuming the generated Exception is not caught, PHP will exit with this status.		No

B.24.1. 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!" />

<!-- Using a condition to achieve the same effect:
<fail message="Detected error!">
  <condition>
    <not>
      <isset property="dontfail"/>
    </not>
  </condition>
</fail>

```

B.24.2. Parameters specified as nested elements.

As an alternative to the if/unless attributes, conditional failure can be achieved using a single nested <condition> element, which should contain exactly one core or custom condition.

B.25. FileHashTask

Calculates either MD5 or SHA1 hash value of a file and stores the value as a hex string in a property and generates a checksum file.

Other popular algorithms [<http://php.net/manual/en/function.hash-algos.php>] like "crc32" or "sha512" may be used with help of the `algorithm` attribute.

Table B.27: Attributes

Name	Type	Description	Default	Required
file	String	Filename	n/a	Yes
hashtype	Integer	Specifies what hash algorithm to use. 0=MD5, 1=SHA1	0	No
algorithm	String	Specifies what hash algorithm to use. Supported algorithms [http://php.net/manual/en/function.hash-algos.php].	n/a	No
propertyname	String	Name of property where the hash value is stored.	filehash-value	No

B.25.1. Example

```
<filehash file = "${builddir}/${tarball}.tar.${compression}" />
<echo msg = "Hashvalue is: ${filehashvalue}" />
```

B.26. FileSizeTask

Stores the size of a specified file in a property. The file size can be returned in different units.

Table B.28: 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
unit	String	File size unit. Examples: M, G, T.	B	No



Note

File size can be written using IEC and SI suffixes, bytes are assumed when suffix is not specified. The following suffixes (case-insensitive) are supported:

Table B.29: Supported file size suffixes

Standard	Suffixes	Equivalence
	B.	1 byte
IEC	K, Ki, KiB, kibi, kibibyte.	1024 bytes
	M, Mi, MiB, mebi, mebibyte.	1024 kibibytes

Standard	Suffixes	Equivalence
SI	G, Gi, GiB, gibi, gibibyte.	1024 mebibytes
	T, Ti, TiB, tebi, tebibyte.	1024 gibibytes
	kB, kilo, kilobyte.	1000 bytes
	MB, mega, megabyte.	1000 kilobytes
	GB, giga, gigabyte.	1000 megabytes
	TB, tera, terabyte.	1000 gigabytes

B.26.1. Examples

```
<filesize file = "./backup.zip"/>
<echo>Your backup size is ${filesize} Bytes</echo>
```

```
<filesize file = "./backup.zip" propertyname = "backup.size"/>
<echo>Your backup size is ${backup.size} Bytes</echo>
```

```
<filesize file = "./backup.zip" unit = "M"/>
<echo>Your backup size is ${filesize} Megabytes</echo>
```

B.27. 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.30: Attributes

Name	Type	Description	Default	Required
<code>inheritall</code>	Boolean	If <code>true</code> , pass all properties to the called target.	<code>false</code>	No
<code>inheritrefs</code>	Boolean	If <code>true</code> , pass all references to the the called target.	<code>false</code>	No
<code>trim</code>	Boolean	If <code>true</code> , any leading or trailing whitespace will be removed from the list item before it is passed to the requested target.	<code>false</code>	No
<code>list</code>	String	The list of values to process, with the <code>delimiter</code> attribute, indicated by the "delimiter" attribute, separating each value.	<code>n/a</code>	No
<code>target</code>	String	The target to call for each token, passing then/a token as the parameter with the name indicated by the "param" attribute.	<code>n/a</code>	Yes
<code>param</code>	String	The name of the parameter to pass the tokens in as to the target.	<code>n/a</code>	Yes

Name	Type	Description	Default	Required
absparam	String	The name of the absolute pathparameter ton/a pass the tokens in as to the target (used while processing nested filesets).		No
delimiter	String	The delimiter string that separates the values, in the "list" parameter. The default is ",".		No
index	String	The name of the property containing the iter-index ation count.		No

B.27.1. 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.27.2. Supported Nested Tags

- path
- dirset
- fileset
- filelist
- mapper

B.28. 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 conditions 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 many 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>` whose conditional evaluates to true will be executed. The `<else>` will be executed only if the `<if>` and all `<elseif>` conditions are false.

B.28.1. 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.29. 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.

The import task may only be used as a top-level task. This means that it may not be used in a target.

Table B.31: Attributes

Name	Type	Description	Default	Required
file	String	The file to import.	n/a	Yes
optional	Boolean	If true, do not stop the build if the file does not exist.	false	No

B.29.1. 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 `bulddocs`, 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 "bulddocs.docs". This enables the new implementation to call the old target, thus enhancing it with tasks called before or after it.

B.29.2. 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.29.3. 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 = "...">
<dirname property = "imported.basedir" file = "${phing.file.imported}"/>
<property file = "${imported.basedir}/imported.properties"/>
</project>
```

or even shorter:

```
<!-- imported.xml -->
<project name = "imported" basedir = "." default = "...">
<property file = "${phing.dir.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.29.4. Examples

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

Additionally, `ImportTask` supports `Filesets`, i.e. you can easily include/exclude one or more files. For more information, see Appendix D, *Core Types*.

```
<import>
  <fileset dir = ".">
    <include name = "path/to/build.xml" />
  </fileset>
  <filelist dir = "." files = "path/to/build.xml" />
</import>
```

B.30. IncludePathTask

Modifies the PHP `include_path` [http://php.net/include_path] configuration option for the duration of this phing run.

The given path can be prepended (default) or appended to the current include path, or it can replace the include path.

Table B.32: Attributes

Name	Type	Description	Default	Required
<code>classpath</code>	String	the new include path[s]	n/a	Yes
<code>classpathRef</code>	String	Reference to a previously defined Path type	n/a	No
<code>mode</code>	String	Whether to prepend, append or replace the include path with the given path.	prepend	No

B.30.1. Examples

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

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

B.31. InputTask

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

Table B.33: Attributes

Name	Type	Description	Default	Required
<code>propertyName</code>	String	The name of the property to set.	n/a	No

Name	Type	Description	Default	Required
defaultValue	String	The default value to be set if no new value isn/a provided.		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 then/a user must supply. If used, one of these options must be chosen.		No
hidden	Boolean	Whether to hide user input.	n/a	No

B.31.1. 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.32. JsonValidateTask

The `JsonValidateTask` checks if a given file contains valid JSON data and fails if not.

Table B.34: Attributes

Name	Type	Description	Default	Required
file	String	Location of the file to be checked.	none	Yes

B.32.1. Example

```
<jsonvalidate file = "config/default.json"/>
```

B.33. LoadFileTask

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

Table B.35: 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
failonerror	Boolean	Whether to halt the build on failure.	true	No
quiet	Boolean	Do not display a diagnostic message (unless Phing has been invoked with the <code>-verbose</code> or <code>-debug</code> switches) or modify the exit status to reflect an error. Setting this to <code>true</code> implies setting <code>failonerror</code> to <code>false</code> .	false	No

B.33.1. Examples

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

B.33.2. Supported Nested Tags:

- `filterchain`

B.34. ManifestTask

This task generates a simple manifest file with optional checksums.

Table B.36: Attributes

Name	Type	Description	Default	Required
salt	String	Salt to use when generating checksums.	n/a	No
checksum	String	Comma separated list of checksums (hashing algorithms) to run, or <code>false</code> to disable checksum generation. Possible values are <code>md5</code> , <code>crc32</code> or any of the algorithms returned by <code>hash_algos()</code> [http://www.php.net/manual/en/function.hash-algos.php].	false	No
file	String	The path to the manifest file.	n/a	Yes.

B.34.1. Supported Nested Tags

- `fileset`

B.35. MkdirTask

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

Table B.37: 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.35.1. Examples

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

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

B.36. 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.36.1. 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.36.2. Attributes and Nested Elements

For further documentation, see Section B.14, “CopyTask”, since MoveTask only is a child of CopyTask and inherits all attributes.

B.37. PathConvert

Converts a path form for a particular platform, optionally storing the result into a given property. It can also be used when you need to convert FileList, FileSet, DirSet into a list, separated by a given character, such as a comma or space, or, conversely, e.g. to convert a list of files in a FileList into a path.

Nested `map` elements can be specified to map Windows drive letters to Unix paths, and vice-versa.

A single nested `mapper` element can be specified to perform any of various filename transformations.

Table B.38: Attributes

Name	Type	Description	Default	Required
targetos	String	The target architecture. This is a shorthand mechanism for specifying both <code>pathsep</code> and <code>dirsep</code> according to the specified target architecture.	N/A	No
dirsep	String	The character(s) to use as the directory separator in the generated paths.	Phing-File::\$separator	No
pathsep	String	The character(s) to use as the path-element separator in the generated paths.	Phing-File::\$pathSeparator	No
property	String	The name of the property in which to place the result converted path.	will be logged if unset	No
refid	String	What to convert, given as a reference to a path, fileset or dirset defined elsewhere	if omitted, a nested path element must be supplied.	No
setonempty	Boolean	Should the property be set, even if the result is the empty string?	true	No
preserveduplicates	Boolean	Whether to preserve duplicate resources.	false	No

B.38. PathToFileSetTask

Converts a path to a fileset. This is useful if you have a path but need to use a fileset as input in a phing task.

Table B.39: Attributes

Name	Type	Description	Default	Required
dir	String	The root of the directory tree of this FileSet.	n/a	Yes
pathrefid	String	The reference to the path to convert from.	n/a	Yes
ignorenonrelative	Boolean	This boolean controls what will happen if any of the files in the path are not in the directory for the fileset. If this is "true" the files are ignored, if this is "false" a build exception is thrown. (Note: if files are not present no check is made).	false	No
name	String	This is the identifier of the fileset to create. This fileset will contain the files that are relative to	n/a	Yes

Name	Type	Description	Default	Required
		the directory root. Any files that are not present will not be placed in the set.		

B.38.1. Examples

```
<path id = "modified.sources.path" dir = "C:\Path\to\phing\classes\phing\" />
<pathtofileset name = "modified.sources.fileset"
pathrefid = "modified.sources.path"
dir = "." />

<copy todir = "C:\Path\to\phing\docs\api">
  <mapper type = "glob" from = "*.php" to = "*.php.bak" />
  <fileset refid = "modified.sources.fileset" />
</copy>
```

B.39. 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.

When a target is invoked by `phingcall`, all of its dependent targets will also be called within the context of any new parameters. For example, if the target "doSomethingElse" depended on the target "init", then using `phingcall` to execute "doSomethingElse" will also execute "init". Note: the top level tasks of a project will always be executed!

Table B.40: 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
inheritAll	Boolean	If true, all	true	No
inheritRefs	Boolean		false	No



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.39.1. Examples

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

```

    </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.39.2. Supported Nested Tags

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

B.40. 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.47, “PropertyTask”), `<fileset>` Tags (see Section D.4, “FileSet”) or `<reference>` Tags.

Table B.41: Attributes

Name	Type	Description	Default	Required
<code>inheritAll</code>	Boolean	If true, pass all properties to the new phing project.	<code>true</code>	No
<code>inheritRefs</code>	Boolean	If true, pass all references to the new phing project.	<code>false</code>	No
<code>dir</code>	String	The directory to use as a base directory for the new phing project. Default is the current project's <code>basedir</code> , unless <code>inheritall</code> has been set to <code>false</code> , in which case it doesn't have a default value. This will override the <code>basedir</code> setting of the called project.	n/a	No
<code>phingFile</code>	String	The build file to use. Defaults to <code>"build.xml"</code> . This file is expected to be a filename relative to the <code>dir</code> attribute given.	n/a	Yes
<code>target</code>	String	The target of the new Phing project to execute. Default is the new project's default target.	n/a	No
<code>haltOnFailure</code>	Boolean	If true, fail the build process when the called build fails	<code>false</code>	No
<code>output</code>	String	Filename to write the Phing output to. This is relative to the value of the <code>dir</code> attribute if it has been set or to the <code>basedir</code> of the current project otherwise.	n/a	No
<code>useNative-basedir</code>	Boolean	If set to <code>"true"</code> , the child build will use the same <code>basedir</code> as it would have used when run from the command line (i.e. the <code>basedir</code> one would	<code>false</code>	No

Name	Type	Description	Default	Required
		expect when looking at the child build's build-file).		

B.40.1. 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.40.2. Supported Nested Tags

- fileset
- property
- reference

B.40.3. Base directory of the new project

The base directory of the new project is set dependent 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.42: How attributes are used

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

B.41. Phingversion

Stores the Phing version (when used as task) or checks for a specific Phing version (when used as condition).

Table B.43: Attributes

Name	Type	Description	Required (Task)	Required (Condition)
<code>atleast</code>	String	The version that this at least. The format is <code>major.minor.point</code> .	No	One of these.

Name	Type	Description	Required (Task)	Required (Condition)
exactly	String	The version that this phing is exactly. The for-No mat is major.minor.point.		
property	String	The name of the property to set.	Yes	No (ignored)

B.41.1. Example

```
<phingversion property = "phingversion"/>
```

Stores the current Phing version in the property `phingversion`.

```
<phingversion property = "phingversion" atleast = "2.9"/>
```

Stores the Phing version in the property `phingversion` if the current Phing version is 2.9.0 or higher. Otherwise the property remains unset.

```
<phingversion property = "phing-is-exact-292" exactly = "2.9.2"/>
```

Sets the property `phing-is-exact-292` if Phing 2.9.2 is running. Neither 2.8.2 nor 2.9.1 would match.

B.42. 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.44: 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. <i>Note:</i> if this attribute is set, the expression must return a value.		No
level	String	Control the level at which php reports statusinfo messages. One of error, warning, info, verbose, debug.		No

B.42.1. Examples

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

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

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

```
<php class = "phing.Phing" function = "start">
  <param value = "-projecthelp"/>
  <param value = "-buildfile"/>
  <param value = "${phing.file}"/>
</php>
```

B.42.2. Supported Nested Tags

- param

B.43. PhpLintTask

The `PhpLintTask` checks syntax (lint) on one or more PHP source code files.

Table B.45: 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 <code>\${php.interpreter}</code> property which is the interpreter used to execute phing itself.	No
cachefile	String	If set, enables writing of last-modified times to one 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

B.43.1. 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.

B.43.2. Supported Nested Tags

- fileset

B.44. PropertyCopy

Copies the value of a named property to another property. This is useful when you need to plug in the value of another property in order to get a property name and then want to get the value of that property name.

Table B.46: Attributes

Name	Type	Description	Default	Required
property	String	The name of the property to set.	n/a	Yes
override	Boolean	If the property is already set, should we change it's value.	false	No
from	String	The name of the property you wish to copy then/a value from.	n/a	Yes
silent	Boolean	Do you want to suppress the error if the "from" property does not exist, and just not set the property "name".	false	No

B.44.1. Example

```
<property name = "org" value = "MyOrg" />
<property name = "org.MyOrg.DisplayName" value = "My Organization" />
<propertycopy property = "displayName" from = "org.${org}.DisplayName" />
```

Sets displayName to "My Organization".

B.45. PropertyRegexTask

Performs regular expression operations on an subject string, and sets the results to a property. There are two different operations that can be performed:

- Replace - The matched regular expression is replaced with a substitution pattern
- Match - Groupings within the regular expression are matched via a selection expression.

Table B.47: Attributes

Name	Type	Description	Default	Required
property	String	The name of the property to set.	n/a	Yes
override	Boolean	If the property is already set, should we change it's value. Can be <code>true</code> or <code>false</code>	<code>false</code>	No
subject	String	The subject to be processed	n/a	Yes
pattern	String	The regular expression pattern which is matched in the subject.	n/a	Yes
match	String	A pattern which indicates what match you want in the returned value. This uses the substitution pattern syntax to indicate where to insert groupings created as a result of the regular expression match.	n/a	Yes (unless a replace is specified)
replace	String	A regular expression substitution pattern, which will be used to replace the given regular expression in the subject.	n/a	Yes (unless a match is specified)
caseSensitive	Boolean	Should the match be case sensitive	<code>true</code>	No
limit	Integer	The maximum possible replacements for each pattern in each subject string. Defaults to -1 (no limit).	-1	No
defaultValue	Integer	The value to set the output property to, if then a subject string does not match the specific regular expression.	n/a	No

B.45.1. Match expressions

Expressions are matched in a the same syntax as a regular expression substitution pattern.

- `$0` indicates the entire property name (default).
- `$1` indicates the first grouping
- `$2` indicates the second grouping
- etc...

B.45.2. Replace

It is important to note that when doing a "replace" operation, if the subject string does not match the regular expression, then the property is not set. You can change this behavior by supplying the "defaultValue" attribute. This attribute should contain the value to set the property to in this case.

- `$0` indicates the entire property name (default).
- `$1` indicates the first grouping
- `$2` indicates the second grouping

- etc...

B.45.3. Example

```
<propertyregex property = "pack.name"
  subject = "package.ABC.name"
  pattern = "package\.([^\.]*)\.name"
  match = "$1"
  casesensitive = "false"
  defaultvalue = "test1"/>

<echo message = "${pack.name}"/>

<propertyregex property = "pack.name"
  override = "true"
  subject = "package.ABC.name"
  pattern = "(package)\.([^\.]*)\. (name) "
  replace = "$1.DEF.$2"
  casesensitive = "false"
  defaultvalue = "test2"/>

<echo message = "${pack.name}"/>
```

B.46. PropertySelector

Selects property names that match a given regular expression and returns them in a delimited list.

Table B.48: Attributes

Name	Type	Description	Default	Required
property	String	The name of the property to set.	n/a	Yes
override	Boolean	If the property is already set, should we false change it's value. Can be true or false	false	No
match	String	The regular expression which is used to select n/a property names for inclusion in the list. This follows the standard regular expression syntax accepted by phing's regular expression tasks.		Yes
select	String	A pattern which indicates what selection pat-tern you want in the returned list. This used the substitution pattern syntax to indicate where to insert groupings created as a result of the regular expression match.		No
casesensitive	String	Should the match be case sensitive.	true	No
replace	String	A regular expression substitution pattern, n/a which will be used to replace the given regular expression in the subject.		Yes (unless a match is specified)
casesensitive	Boolean	Should the match be case sensitive	true	No
delimiter	String	The delimiter used to separate entries in the, resulting property		No
distinct	Boolean	Should the returned entries be a distinct set false (no duplicate entries).		No

B.46.1. Select expressions

Expressions are matched in a the same syntax as a regular expression substitution pattern.

- \$0 indicates the entire property name (default).
- \$1 indicates the first grouping
- \$2 indicates the second grouping
- etc...

B.46.2. Example

```
<property name = "package.ABC.name" value = "abc pack name" />
<property name = "package.DEF.name" value = "def pack name" />
<property name = "package.GHI.name" value = "ghi pack name" />
<property name = "package.JKL.name" value = "jkl pack name" />

<propertyselector property = "pack.list"
  delimiter = ","
  match = "package\.([^\.]*)\.name"
  select = "$1"
  casesensitive = "false" />
```

B.47. PropertyTask

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

Table B.49: 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 withn/a 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".		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.n/a Prefix is applied to properties loaded from specified file. A "." is appended to the prefix if not specified.		No

Name	Type	Description	Default	Required
refid	String	A reference to a previously defined property	n/a	No
logoutput	Boolean	Whether to log returned output as MSG_INFO instead of MSG_VERBOSE.	true	No
quiet	Boolean	Whether to display a warning if the property file does not exist.	true	No
required	Boolean	Whether to halt with an error if the property file does not exist.	false	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.47.1. 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>
      <regex pattern = "Hello" replace = "World" ignoreCase = "true"/>
    </replaceregexp>
  </filterchain>
</property>
```

B.47.2. Supported Nested Tags:

- `filterchain`

B.48. Record

A recorder is a listener to the current build process that records the output to a file.

Several recorders can exist at the same time. Each recorder is associated with a file. The filename is used as a unique identifier for the recorders. The first call to the recorder task with an unused filename will create a recorder (using the parameters provided) and add it to the listeners of the build. All subsequent calls to the recorder task using this filename will modify that recorders state (recording or not) or other properties (like logging level).

Some technical issues: the file's output stream is flushed for "finished" events (buildFinished, targetFinished and taskFinished), and is closed on a buildFinished event.

Table B.50: Attributes

Name	Type	Description	Default	Required
name	String	The name of the file this logger is associatedn/a with.		yes
action	String	This tells the logger what to do: should itn/a start recording or stop? The first time that the recorder task is called for this logfile, and if this attribute is not provided, then the default for this attribute is "start". If this attribute is not provided on subsequent calls, then the state remains as previous. [Values = {start stop}, Default = no state change]		no
append	Boolean	Should the recorder append to a file, or cre-no ate a new one? This is only applicable the first time this task is called for this file. [Values = {yes no}, Default=no]		no
emacsmode	Boolean	Removes [task] banners like Phings's -emacsfalse command line switch if set to true.	false	no
loglevel	String	At what logging level should this recorderfalse instance record to? This is not a once on-ly parameter (like append is) -- you can in-crease or decrease the logging level as the build process continues. [Values= {error warn info verbose debug}, Default = no change]	false	no

B.48.1. Example

The following build.xml snippet is an example of how to use the recorder to record just the `<echo>` task:

```
...
<record name = "log.txt" action = "start"/>
<echo ...
<record name = "log.txt" action = "stop"/>
...
```

The following two calls to `<record>` set up two recorders: one to file "records-simple.log" at logging level info (the default) and one to file "ISO.log" using logging level of verbose.

```
...
<record name = "records-simple.log"/>
<record name = "ISO.log" loglevel = "verbose"/>
...
```

B.49. ReflexiveTask

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

Table B.51: Attributes

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

B.49.1. Examples

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

B.49.2. Supported Nested Tags:

- fileset
- filterchain

B.50. Relentless

The <relentless> task will execute all of the nested tasks, regardless of whether one or more of the nested tasks fails.

When <relentless> has completed executing the nested tasks, it will either

- fail, if any one or more of the nested tasks failed; or
- succeed, if all of the nested tasks succeeded.

An appropriate message will be written to the log.

Tasks are executed in the order that they appear within the <relentless> task. It is up to the user to ensure that relentless execution of the nested tasks is safe.

Table B.52: Attributes

Name	Type	Description	Default	Required
description	String	A string that will be included in the log output. This can be useful for helping to identify sections of large phing builds.	N/A	No
terse	Boolean	Setting this to true will eliminate some of the progress output generated by <relentless>. This can reduce clutter in some cases.	false	No

The only nested element supported by <relentless> is a list of tasks to be executed. At least one task must be specified.

It is important to note that `<relentless>` only proceeds relentlessly from one task to the next - it does not apply recursively to any tasks that might be invoked by these nested tasks. If a nested task invokes some other list of tasks (perhaps by `<phingcall>` for example), and one of those other tasks fails, then the nested task will stop at that point.

B.50.1. Example

A relentless task to print out the first five canonical variable names:

```
<relentless description="The first five canonical variable names.">
  <echo>foo</echo>
  <echo>bar</echo>
  <echo>baz</echo>
  <echo>bat</echo>
  <echo>blah</echo>
</relentless>
```

which should produce output looking more or less like

```
[relentless] Relentlessly executing: The first five canonical variable names.
[relentless] Executing: task 1
[echo] foo
[relentless] Executing: task 2
[echo] bar
[relentless] Executing: task 3
[echo] baz
[relentless] Executing: task 4
[echo] bat
[relentless] Executing: task 5
[echo] blah
[relentless] All tasks completed successfully.
```

If you change the first line to set the `terse` parameter,

```
<relentless terse="true" description="The first five canonical variable names."/>
```

the output will look more like this:

```
[relentless] Relentlessly executing: The first five canonical variable names.
[echo] foo
[echo] bar
[echo] baz
[echo] bat
[echo] blah
[relentless] All tasks completed successfully.
```

If we change the third task to deliberately fail

```
<relentless terse = "true"
  description = "The first five canonical variable names.">
  <echo>foo</echo>
  <echo>bar</echo>
  <fail>baz</fail>
  <echo>bat</echo>
  <echo>blah</echo>
</relentless>
```

then the output should look something like this.

```
[relentless] Relentlessly executing: The first five canonical variable names.
[echo] foo
[echo] bar
[relentless] Task task 3 failed: baz
[echo] bat
[echo] blah
```

```
BUILD FAILED
/path/build.xml:1177: Relentless execution: 1 of 5 tasks failed.
```

B.51. ReplaceRegexpTask

Replaces the occurrences of a given regular expression with a substitution pattern in a selected file or set of files.

Table B.53:

Name	Type	Description	Default	Required
file	String	File to apply regular expression on	n/a	Yes (or file-set)
match	String	Regular expression match pattern	n/a	Yes (or pattern)
pattern	String	Regular expression match pattern	n/a	Yes
replace	String	The replacement string	n/a	Yes
flags	String	Regular expression flags	n/a	no
failonerror	Boolean	If set to true, the task will fail on error	false	No

B.51.1. Supported Nested Tags

- fileset

B.52. 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.54: 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	Yes
propertyName	String	The name of the property to set with resolved (absolute) path.	n/a	Yes

Name	Type	Description	Default	Required
level	String	Control the level at which status messages are reported. One of error, warning, info, verbose, debug.	verbose	No

B.52.1. Examples

```
<property name = "relative_path" value = "../dirname"/>
<resolvepath propertyName = "absolute_path" file = "${relative_path}"/>
<echo>Resolved [absolute] path: ${absolute_path}</echo>
```

B.53. Retry

Retry is a container which executes a single nested task until either: there is no failure; or: its retrycount has been exceeded. If this happens a BuildException is thrown..

Table B.55: Attributes

Name	Type	Description	Default	Required
retrycount	Integer	number of times to attempt to execute the nested task	1	Yes
retrydelay	Integer	number of seconds to wait between retry attempts task.	0	No, defaults to no delay

Any valid Phing task may be embedded within the retry task.

B.53.1. Example

```
<retry retrycount = "3">
  <httpget url = "http://www.unreliable-server.com/unreliable.tar.gz" dir = "/home/retry"/>
</retry>
```

This example shows how to use <retry> to wrap a task which must interact with an unreliable network resource.

B.54. RunTargetTask

Phing task that runs a target without creating a new project.



Difference to <phingcall>

The main difference of <runtarget> and <phingcall> is that <phingcall> will start the phing target in a new project and will not affect the main project. <runtarget> calls a target in the

same project, which could be have an effect on any existing properties. Dependency management would only be given by `<phingcall>`.

Table B.56: Attributes

Name	Type	Description	Default	Required
target	String	The name of the target to run.	n/a	Yes

B.54.1. Example

```
<runtarget target = "test" />
```

B.55. SleepTask

A task for sleeping a short period of time, useful when a build or deployment process requires an interval between tasks.

Table B.57: Attributes

Name	Type	Description	Default	Required
hours	Integer	hours to to add to the sleep time	0	no
minutes	Integer	minutes to add to the sleep time	0	no
seconds	Integer	seconds to add to the sleep time	0	no
milliseconds	Integer	milliseconds to add to the sleep time	0	no
failonerror	Boolean	flag controlling whether to break the build on true an error.	no	No

B.55.1. Example

```
<sleep seconds = "2" />
```

B.56. SortList

Sort a delimited list of items in their natural string order. Note that the `value` and `refid` attributes are mutually exclusive, and the `value` attribute takes precedence if both are specified.

Table B.58: Attributes

Name	Type	Description	Default	Required
property	String	The name of the property to set.	n/a	Yes
overwrite	Boolean	If the property is already set, should we false change it's value.	false	No
value	String	The list of values to process, with the delim-n/a iter character, indicated by the "delimiter" attribute, separating each value.		Yes, unless "refid" is specified.

Name	Type	Description	Default	Required
refid	String	The id of where the list of values to sort isn/a stored.		Yes, unless "value" is specified.
delimiter	String	The delimiter string that separates the values, in the "list" attribute.		No
flags	String	Sort flags depending on the php version andn/a one of: SORT_REGULAR, SORT_NUMERIC, SORT_STRING, SORT_LOCALE_STRING, SORT_NATURAL, SORT_FLAG_CASE		No

B.56.1. Example

```
<property id = "test" name = "my.list" value = "z;y;X;w;v;U;t" />
<sortlist property = "my.sorted.list" refid = "test"
  delimiter = ";"
  flags = "SORT_NATURAL|SORT_FLAG_CASE" />
```

B.57. Subphing Task

Calls a given target for all defined sub-builds. This is an extension of Phing for bulk project execution. This task must not be used outside of a target if it invokes the same build file it is part of.

subphing uses phing task internally so many things said in phing's manual page apply here as well.

Table B.59: Attributes

Name	Type	Description	Default	Required
genericphing-file	Phing-File	Build file path, to use in conjunction with directories. Use genericphingfile, in order to run the same build file with different basedirs. If this attribute is set, phingfile is ignored.	n/a	No
inheritAll	Boolean	Corresponds to <phing>'s inheritall attribute but defaults to "false" in this task.	false	No
inheritRefs	Boolean	Corresponds to <phing>'s inheritrefs attribute.	false	No
buildpath	Path	Set the buildpath to be used to find sub-projects.	n/a	No
phingFile	String	Build file name, to use in conjunction with directories. No ignored if genericphingfile is set.		No
target	String	The target to execute. Default is the new sub-project's default target.	n/a	No
failonerror	Boolean	Sets whether to fail with a build exception on error, or go on.	true	No

Name	Type	Description	Default	Required
verbose	Boolean	Enable/disable log messages showing when each sub-build path is entered/exited.	false	No

B.57.1. Supported Nested Tags

- buildpath
- buildpathelement
- fileset
- property
- reference

B.58. SwitchTask

Task definition for the phing task to switch on a particular value.

Table B.60: Attributes

Name	Type	Description	Default	Required
value	String	The value to switch on.	n/a	Yes
caseinsensitive	Boolean	Should we do case insensitive comparisons?	false	No

B.58.1. Supported Nested Tags

At least one <case> or <default> is required.

case

An individual case to consider, if the value that is being switched on matches to value attribute of the case, then the nested tasks will be executed.

Table B.61: Attributes

Name	Type	Description	Default	Required
value	String	The value to match against the tasks value attribute.	n/a	Yes

default

The default case for when no match is found. Must not appear more than once per task.

B.58.2. Examples

```
<switch value = "${foo}">
  <case value = "bar">
    <echo message = "The value of property foo is bar" />
  </case>
```

```

<case value = "baz">
  <echo message = "The value of property foo is baz" />
</case>
<default>
  <echo message = "The value of property foo is not sensible" />
</default>
</switch>

```

B.59. SymlinkTask

Creates symlink(s) to a specified file / directory or a collection of files / directories.

Table B.62: 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
relative	Boolean	Whether to create relative symlinks	false	No

B.59.1. 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

B.59.2. Supported Nested Tags

- fileset

B.60. TaskdefTask

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

Table B.63: Attributes

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

B.60.1. 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.60.2. Supported Nested Tags

- classpath

B.61. Tempfile Task

This task sets a property to the name of a temporary file. Unlike PhingFile::createTempFile(), this task does not actually create the temporary file, but it does guarantee that the file did not exist when the task was executed.

Table B.64: Attributes

Name	Type	Description	Default	Required
property	String	Sets the property you wish to assign the temporary file to.	n/a	yes
destDir	String	Sets the destination directory. If not set, the basedir directory is used instead.	basedir	no
prefix	String	Sets the optional prefix string for the temp file.	n/a	no
suffix	String	Sets the optional suffix string for the temp file.	n/a	no
deleteOnExit	Boolean	Whether the temp file will be marked for deletion on normal exit (even though the file may never be created).	false	no
createFile	Boolean	Whether the temp file should be created by this task.	false	no

B.61.1. Example

```
<tempfile property = "temp.file"/>
```

create a temporary file

```
<tempfile property = "temp.file" suffix = ".xml"/>
```

create a temporary file with the .xml suffix

```
<tempfile property = "temp.file" destDir = "build"/>
```

create a temporary file in the build subdirectory

B.62. ThrowTask

Extension of build in `FailTask` that can throw an exception that is given by a reference. This may be useful if you want to rethrow the exception that has been caught by a `TryCatchTask` in the `<catch>` block.

Table B.65: Attributes

Name	Type	Description	Default	Required
refId	String	Id of the referenced exception.	n/a	No



Note

In addition, all attributes of the `FailTask` are supported.

B.62.1. Example

```
<target name = "tryCatchThrow">
```

```

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

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

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

<echo>As property: ${foo}</echo>
<property name = "baz" refid = "bar" />
<echo>From reference: ${baz}</echo>

<echo>Throw ...</echo>
<throw refid = "bar" />

</target>

```

B.63. 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.66: Attributes

Name	Type	Description	Default	Required
file	String	The file which time is to be changed.	n/a	Yes, or nested <code><fileset></code> tag
datetime	Date-Time	The date and time the mtime of the file is to be now set to. The format is "MM/DD/YYYY HH:MM AM or PM"		No
seconds	Integer	The number of seconds since Midnight Jan 1 now 1970 (Unix epoch).		No
millis	Integer	The number of milliseconds since Midnight now Jan 1 1970 (Unix epoch). Note: milliseconds are converted to seconds internally. When using this option the value must be greater than 1000.		No
seconds	Integer	The number of seconds since Midnight Jan 1 now 1970 (Unix epoch).		No
mkdirs	Boolean	Whether to create nonexistent parent directories when touching new files.	false	No
verbose	Boolean	Whether to log the creation of new files.	true	No

B.63.1. Examples

```

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

```

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

```
<target name = "map">
  <touch file = "${tmp.dir}/touchtest"/>
  <touch>
    <fileset file = "${tmp.dir}/touchtest" />
    <mapper type = "composite">
      <mapper type = "glob" from = "*" to = "*foo" />
      <mapper type = "glob" from = "*" to = "*bar" />
    </mapper>
  </touch>
</target>
```

B.63.2. Supported Nested Tags

- filelist
- fileset
- mapper

B.64. TruncateTask

Modify the length of a file, as the intermittently available truncate Unix utility/function.

When length and adjust are not set an empty file is created.

Table B.67: Attributes

Name	Type	Description	Default	Required
file	String	The name of the file.	n/a	Yes
length	String	Specifies the file size. Examples: 5000B, n/a, 250K, 1M.		No
adjust	String	The value to increase or decrease (if you specify a negative value) the size of a file. Examples: -100, 250B, 4K.	n/a	No
create	Boolean	Whether to create nonexistent files.	true	No
makedirs	Boolean	Whether to create nonexistent parent directories when creating new files.	false	No



Note

File size can be written using IEC and SI suffixes, bytes are assumed when suffix is not specified. The following suffixes (case-insensitive) are supported:

Table B.68: Supported file size suffixes

Standard	Suffixes	Equivalence
IEC	B.	1 byte

Standard	Suffixes	Equivalence
	K, Ki, KiB, kibi, kibibyte.	1024 bytes
	M, Mi, MiB, mebi, mebibyte.	1024 kibibytes
	G, Gi, GiB, gibi, gibibyte.	1024 mebibytes
	T, Ti, TiB, tebi, tebibyte.	1024 gibibytes
	kB, kilo, kilobyte.	1000 bytes
	MB, mega, megabyte.	1000 kilobytes
SI	GB, giga, gigabyte.	1000 megabytes
	TB, tera, terabyte.	1000 gigabytes

B.64.1. Examples

```
<truncate file = "foo" />
```

B.65. TryCatchTask

This task is a wrapper task that lets you run task(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> [<http://ant-contrib.sourceforge.net/tasks/tasks/trycatch.html>].

Table B.69: Attributes

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

B.65.1. 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.66. TstampTask

Sets the `DSTAMP`, `TSTAMP`, and `TODAY` properties in the current project. By default, the `DSTAMP` property is in the format `"yyyyMMdd"`, `TSTAMP` is in the format `"HHmm"`, finally `TODAY` contains locale-sensitive date, for example `"March 13, 2023"`. 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.

The magic property `phing.tstamp.now` can be used to specify a fixed date value in order to create reproducible builds. Its value must be a number and is interpreted as seconds since the epoch (midnight 1970-01-01). With `phing.tstamp.now.iso` you could also specify that value in `DateTime` compatible format. If you specify a value in an invalid format an `INFO` message will be logged and the value will be ignored.

Table B.70: Attributes

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

B.66.1. Examples

```
<tstamp/>
```

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

```

<tstamp>
  <format property = "DAY" pattern = "EEEE" locale = "nl_NL"/>
</tstamp>

```

sets the standard properties as well as the property `DAY` with the pattern `"EEEE"` (day of week) 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.66.2. 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.

Table B.71: Attributes

Name	Type	Description	Default	Required
property	String	The property to receive the date/time string inn/a the given pattern.		Yes
pattern	String	The date/time pattern to be used, pattern mustn/a follow ICU syntax.		Yes
locale	String	The locale used to create date/time string.	n/a	No
timezone	String	The timezone to use for displaying time.	n/a	No

B.66.3. ICU syntax

ICU syntax replaces string of characters with proper date and time data. Please refer to official ICU documentation to see the complete Date/Time Format Syntax [https://unicode-org.github.io/icu/user-guide/format_parse/datetime/#date-field-symbol-table] reference.

Table B.72: ICU syntax

Symbol	Meaning	Pattern	Example Output
y	year	yy y or yyyy	96 1996
M	month in year	M MM MMMM	9 09 September
d	day in month	d dd	2 02
E	day of week	E, EE, or EEE EEEE	Tue Tuesday
a	AM or PM	a, aa, or aaa aaaa	PM [abbrev] PM [wide]
h	hour in am/pm (1~12)	h hh	7 07
H	hour in day (0~23)	H HH	0 00
m	minute in hour	m mm	4 04
s	seconds in minute	s ss	5 05
Z	Time Zone: ISO8601 ba-Z, ZZ, or ZZZ sic hms? / RFC 822 ZZZZ Time Zone: long localized ZZZZZ GMT (=OOOO) Time Zone: ISO8601 ex- tended hms? (=XXXXX)		-0800 GMT-08:00 -08:00, -07:52:58, Z
'	escape for text	'	(nothing)

Symbol	Meaning	Pattern	Example Output
' '	two single quotes duce one	pro-' '	'



Note

ICU syntax was introduced in **Phing 3**, if you are using an older version then you must use strftime [https://www.php.net/manual/en/function.strftime.php] format syntax.

B.67. TypedefTask

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

Table B.73: Attributes

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

B.67.1. Examples

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

B.67.2. Supported Nested Tags

- classpath

B.68. URLEncodeTask

The URLEncode task will encode a given property for use within a a URL string. This value which is actually set will be encoded via the urlencode() function. Typically, you must do this for all parameter values within a URL.

Table B.74: Attributes

Name	Type	Description	Default	Required
property	String	The name of the property to set.	n/a	Yes
override	Boolean	If the property is already set, should we false change it's value. Can be true or false	we false	No
value	String	The value of the property.	n/a	No, if refid is specified
refid	String	The id of a saved reference whose value will n/a be the value of the property.		No, if value is specified

B.68.1. Example

```
<urlencode name = "file.location" value = "C:\\wwwhome\\my reports\\report.xml" />
```

B.69. 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.75: 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 then/a status		Yes (or nested mapper)

B.69.1. 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.69.2. Supported Nested Tags

- filelist
- fileset
- mapper

B.70. Variable

The Variable task provides a mutable property to Phing and works much like variable assignment in PHP. This task is similar to the standard Phing Property task, except that THESE PROPERTIES ARE MUTABLE. While this goes against the standard Phing use of properties, occasionally it is useful to be able to change a property value within the build. In general, use of this task is DISCOURAGED, and the standard Phing Property should be used if possible. Having said that, in real life I use this a lot.

Variables can be set individually or loaded from a standard properties file. A 'feature' of variables is that they can override properties, but properties cannot override variables. So if an already established property exists, its value can be reassigned by use of this task.

Table B.76: Attributes

Name	Type	Description	Default	Required
name	String	The name of the property to set.	None	Yes, unless 'file' is used.
value	String	The value of the property.	""	No
unset	Boolean	Removes the property from the project as if it had never been set.	false	No
file	String	The name of a standard properties file to load variables from.	None	No

B.70.1. Example

```

<var name = "x" value = "6"/>
<echo>x = ${x}</echo> <!-- print: 6 -->

<var name = "x" value = "12"/>
<echo>x = ${x}</echo> <!-- print: 12 -->

<var name = "x" value = "6 + ${x}"/>

```

```

<echo>x = ${x}</echo> <!-- print: 6 + 12 -->

<var name = "str" value = "I "/>
<var name = "str" value = "${str} am "/>
<var name = "str" value = "${str} a "/>
<var name = "str" value = "${str} string."/>
<echo>${str}</echo> <!-- print: I am a string. -->

<var name = "x" value = "6"/>
<echo>x = ${x}</echo> <!-- print: 6 -->

<property name = "x" value = "12"/>
<echo>x = ${x}</echo> <!-- print: 6 (property can't override) -->

<var name = "x" value = "blue"/>
<tstamp>
  <format property = "x" pattern = "%A"/>
</tstamp>
<echo>Today is ${x}.</echo> <!-- print: Today is blue. -->

<var name = "x" value = "" unset = "true"/>
<tstamp>
  <format property = "x" pattern = "%A"/>
</tstamp>
<echo>Today is ${x}.</echo> <!-- print: Today is Friday. -->

```

B.71. 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). Alternatively you can use 'v' as prefix (e.g. v1.3.2).

Table B.77: Attributes

Name	Type	Description	Default	Required
releasetype	String	Specifies desired version release (Major, Minor or Bugfix)		Yes
file	String	File containing three-part version number to build version increment	build.version	No
property	String	Property which contains the resulting version number	build.version	No
propFile	Boolean	If true, version will be saved using <i>property</i> file format (i.e. key=value).	false	No
startingVersion	String	Starting version string, if version file does not exist.	0.0.0	No

B.71.1. Example

```
<version releasetype = "Major" file = "version.txt" property = "version.number"/>
```

```
<version releasetype = "Minor" startingVersion = "v5.7" propFile = "true"/>
```

B.72. WaitForTask

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

Table B.78: Attributes

Name	Type	Description	Default	Required
MaxWait	Integer	Set the maximum length of time to wait in units	3min	Yes
MaxWaitUnit	String	Set the max wait time unit. Must be one of "week", "day", "hour", "minute", "second", "millisecond"		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"		No
TimeoutProperty	String	Name of the property to set after a timeout.	null	No

B.72.1. Examples

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

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

B.72.2. Supported Nested Tags

All conditionals including and, or, not etc.

B.73. 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 XsltFilter 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.79: 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

Name	Type	Description	Default	Required
re-solvestylesheetexternals	Boolean	Whether to resolve entities in the stylesheet.	false	No
html	Boolean	Whether to work on HTML or XML.	false	No

Note: You can also use all the attributes available for Section B.14, “CopyTask”.

B.73.1. Examples

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

B.73.2. Supported Nested Tags

- mapper
- filterchain
- param

Note: You can use all the elements also available for Section B.14, “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
<code>executable</code>	String	ApiGen executable name.	<code>apigen</code>	No
<code>action</code>	String	ApiGen action to be executed.	<code>generate</code>	No
<code>config</code>	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.
<code>source</code>	String	List of source files or directories.	n/a	
<code>destination</code>	String	Destination directory.	n/a	
<code>exclude</code>	String	List of masks (case sensitive) to exclude files or directories from processing.	n/a	
<code>skipdocpath</code>	String	List of masks (case sensitive) to exclude elements from documentation generating.	n/a	No
<code>charset</code>	String	Character set of source files.	UTF-8	No
<code>main</code>	String	Main project name prefix.	n/a	No
<code>title</code>	String	Title of generated documentation.	n/a	No
<code>baseurl</code>	String	Documentation base URL.	n/a	No
<code>googlecseid</code>	String	Google Custom Search ID.	n/a	No
<code>googlecselabel</code>	String	Google Custom Search label.	n/a	No
<code>googleanalytics</code>	String	Google Analytics tracking code.	n/a	No
<code>templateconfig</code>	String	Template config file name.	n/a	If not set the default template is used.

Name	Type	Description	Default	Required
templatetheme	String	Template theme file name.	n/a	If not set the default template is used.
accesslevels	String	Element access levels. Documentation onlypublic, for methods and properties with the given access level will be generated.		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
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. ComposerTask

The `ComposerTask` runs the Composer tool (<http://getcomposer.org>) directly from Phing.

Table C.2: Attributes

Name	Type	Description	Default	Required
command	String	The Composer command to execute.	n/a	Yes
composer	String	Path to Composer.	composer.phar, if not found it tries to use com-	No

Name	Type	Description	Default	Required
			poser executable from your system.	
php	String	Path to the PHP interpreter	Defaults to the <code>\${php-interpreter}</code> property which is the interpreter used to execute phing itself.	No

C.2.1. Supported Nested Tags

- `arg`

Table C.3: Attributes

Name	Type	Description	Default	Required
value	String	A single command-line argument; can contain space characters.	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 <code>;</code> or <code>:</code> as path separators and Phing will convert it to the platform's local conventions.	n/a	
line	String	A space-delimited list of command-line arguments.	n/a	

C.2.2. Example

```
<composer command = "install">
  <arg value = "--no-dev"/>
  <arg value = "--no-interaction"/>
</composer>
```

C.3. 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.3.1. Example

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

C.3.2. Supported Nested Tags

- fileset

C.4. CoverageReportTask

The CoverageReportTask formats a coverage database into a framed HTML report using XSLT. The report can optionally make use of the **Generic Syntax Highlighting** library, GeSHi (See GeSHi Home-page [<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.4: Attributes

Name	Type	Description	Default	Required
outfile	String	The location for the intermediate XML file.	coverage.db	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 is to be enabled
geshilanguagepath	String	Language to use with GeSHi.	n/a	No

C.4.1. Example

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

C.4.2. Supported Nested Tags

- report

Table C.5: Attributes

Name	Type	Description	Default	Required
styledir	String	The directory where the stylesheets are located.	The directory in the Phing installation.	No
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 sortable JavaScript library (see http://www.kryogenix.org/code/browser/sortable/).	false	No

C.5. CoverageSetupTask

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

Table C.6: Attributes

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

C.5.1. 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.5.2. Supported Nested Tags

- classpath
- fileset

- filelist

C.6. 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.7: Attributes

Name	Type	Description	Default	Required
database	String	The location of the coverage database. (This is optional if CoverageSetupTask has run before.)		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.6.1. Example

```
<coverage-threshold database = "../reports/coverage.db"/>
```

C.6.2. 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.7. 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.8: 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 dbdeploy will apply all changes that aren't applied already (in ascending order).	false	No

C.7.1. 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.8. 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.9: Attributes

Name	Type	Description	Default	Required
rsyncPath	String	Path to rsync command.	/usr/bin/rsync	Yes
sourceDir	String	Source directory (use [user@]host:path for remote sources).	n/a	Yes
destinationDir	String	Destination directory (use [user@]host:path/n/a for remote destinations). Note: sub directories are created by default if they do not exist in the destination directory.	n/a	Yes
exclude	String	Excluded file matching pattern. Use comma-separated values to exclude multiple files/directories, e.g.: a,b	n/a	No
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
options	String	Any options that rsync supports, removes the default options. Should you wish to change the port ssh uses for remote transfers, set this attribute to -e 'ssh -p XXXXX' -rpKz1	n/a	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

Name	Type	Description	Default	Required
port	Integer	Port for ssh authentication used by identityFile.	22	No

C.8.1. Examples

```
<filesync sourcedir = "/var/www/development/project1"
  destinationdir = "/var/www/project1" />

<filesync sourcedir = "host::module" destinationdir = "/var/www/project1/" />

<filesync
  sourcedir = "/var/www/development/project1"
  destinationdir = "user@server:/var/www/project1"
  dryrun = "true"
  itemizechanges = "true"
  verbose = "true"
  checksum = "true" />
```

In the `sourcedir` and `destinationdir` properties user name for remote connections is optional.

C.9. FtpDeployTask

Deploys a set of files to a remote FTP server.

Table C.10: 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
ssl	boolean	Whether to connect via SSL. This requires Net/FTP to be installed.	false	No
dir	String	Directory on the remote server.	none	No
mode	String	The transfer mode to use, either <code>ascii</code> or <code>binary</code> .		No
clearfirst	Boolean	Delete all files in the remote directory before uploading.	false	No
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

Name	Type	Description	Default	Required
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
rawdatafallback	boolean	If Net_FTP is not able to parse the raw ftp data, the depends option does not work at all. Setting rawdatafallback will cause phing trying to parse the ftp data on its own, so the depends option might work again. If depends is set to false, rawdatafallback is ignored.	No	No
skiponsamesize	Boolean	Skip upload, if file of same size exists.	false	No

C.9.1. Example

```
<ftpdeploy
  host = "${ftp.host}"
  port = "${ftp.port}"
  username = "${ftp.username}"
  password = "${ftp.password}"
  dir = "${ftp.dir}"
  ssl = "true"
  passive = "false"
  mode = "${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.9.2. Supported Nested Tags

- fileset

The files to deploy

C.10. GitArchiveTask

Create an archive of files from a named tree.

Table C.11: Attributes

Name	Type	Description	Default	Required
gitPath	String	Path to Git binary	/usr/bin/ git	No

Name	Type	Description	Default	Required
repository	String	The repository.	n/a	One of these attributes is required.
remoterepo	String	The remote repository.	n/a	
treeish	String	The tree or commit to produce an archive for.	n/a	Yes
output	String	Write the archive to file.	n/a	No
prefix	String	Prepend prefix to each filename in the archive.	n/a	No
format	String	Format of the resulting archive: tar or zip. If this option is not given, and the output file is specified, the format is inferred from the filename if possible (e.g. writing to "foo.zip" makes the output to be in the zip format). Otherwise the output format is tar	n/a	No

C.10.1. Example

```

<gitclone gitPath = "${git-path}"
    singleBranch = "true"
    repository = "${repo.dir.resolved}"
    targetPath = "${tmp.dir.resolved}/test" />
<gitarchive
    gitPath = "${git-path}"
    repository = "${tmp.dir.resolved}/test"
    treeish = "HEAD"
    format = "zip"
    output = "${tmp.dir.resolved}/output.zip"
/>

```

C.11. 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.12: 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	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

Name	Type	Description	Default	Required
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.11.1. 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.12. 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.13: 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> inorigin git-checkout [http://www.kernel.org/pub/software/scm/git/docs/git-checkout.html].		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-check-	false	No

Name	Type	Description	Default	Required
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.12.1. 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.13. 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/a 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

Name	Type	Description	Default	Required
depth	Integer	Create a shallow clone with a history truncated to the specified number of revisions. See --depth option of git-clone [http://www.kernel.org/pub/software/scm/git/docs/git-clone.html].	0	No
singleBranch	Boolean	Clone only one branch. See --single-branch option of git-clone [http://www.kernel.org/pub/software/scm/git/docs/git-clone.html].	false	No
branch	String	Checkout branch instead of the remote's HEAD.	sn/a	Yes

C.13.1. 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.14. GitCommitTask

Record changes to the repository. See official documentation [http://www.kernel.org/pub/software/scm/git/docs/git-commit.html].

Table C.15: 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.14.1. Example

```
<!-- commit all modified / deleted files -->
```

```
<gitcommit
repository = "/path/to/repo"
message = "Commit message" allFiles = "true" />;
```

C.14.2. Supported Nested Tags

- fileset

C.15. GitDescribeTask

This task finds the most recent tag that is reachable from a commit. If the tag points to the commit, then only the tag is shown. Otherwise, it suffixes the tag name with the number of additional commits on top of the tagged object and the abbreviated object name of the most recent commit.

Table C.16: 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
outputProperty	String	Property name to set with output value from git-describe.	n/a	No
all	Boolean	Instead of using only the annotated tags, use any ref found in refs/ namespace. This op- tion enables matching any known branch, re- mote-tracking branch, or lightweight tag.	false	No
tags	String	Instead of using only the annotated tags, use any tag found in refs/tags namespace. This option enables matching a lightweight (non- annotated) tag.	false	No
contains	Boolean	Instead of finding the tag that predates the commit, find the tag that comes after the com- mit, and thus contains it. Automatically implies --tags.	false	No
long	Boolean	Always output the long format (the tag, the number of commits and the abbreviated com- mit name) even when it matches a tag.	false	No
always	Boolean	Show uniquely abbreviated commit object as fallback.	false	No
abbrev	Integer	Instead of using the default 7 hexadecimal dig- its as the abbreviated object name, use n dig- its, or as many digits as needed to form a unique object name. An n of 0 will suppress long format, only showing the closest tag.	n/a	No
match	String	Only consider tags matching the given glob pattern, excluding the "refs/tags/" prefix. This can be used to avoid leaking private tags from the repository.	n/a	No

Name	Type	Description	Default	Required
committish	String	Commit-ish object names to describe. De-HEAD faults to HEAD if omitted.		No
candidates	Integer	Instead of considering only the 10 most recent tags as candidates to describe the input commit-ish consider up to n candidates. Increasing n above 10 will take slightly longer but may produce a more accurate result. An n of 0 will cause only exact matches to be output.		No

C.15.1. Example

```
<gitdescribe repository = "${repo.dir}"
  tags = "true"
  abbrev = "0"
  match = "**-*.*"
  outputProperty = "mostRecentTag" />
```

C.16. 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 origin of a fetch or pull operation. See <repository> in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].		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 the standard error stream. See --quiet in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	false	No
all	Boolean	Fetch all remotes. See --all in git-fetch [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	false	No

Name	Type	Description	Default	Required
keep	Boolean	Keep downloaded pack. See <code>--keep</code> in <code>git-fetch</code> [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 <code>--prune</code> in <code>git-fetch</code> [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	false	No
tags	Boolean	See <code>--tags</code> in <code>git-fetch</code> [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	false	No
notags	Boolean	See <code>--no-tags</code> in <code>git-fetch</code> [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	false	No
force	Boolean	When <code>git fetch</code> is used with <code><rbranch></code> refspect, it refuses to update the local branch <code><lbranch></code> unless the remote branch <code><rbranch></code> it fetches is a descendant of <code><lbranch></code> . This option overrides that check. See <code>--force</code> in <code>git-fetch</code> [http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html].	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}" />

<!-- 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.17. GitGcTask

Cleanup unnecessary files and optimize the local repository.

Table C.18: Attributes

Name	Type	Description	Default	Required
gitPath	String	Path to Git binary.	/usr/bin/ git	No

Name	Type	Description	Default	Required
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 2.weeks ago. See --prune option of git-gc [http://www.kernel.org/pub/software/scm/git/docs/git-gc.html].	2.weeks ago	No

C.17.1. 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.18. GitInitTask

Create an empty git repository or reinitialize an existing one.

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
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.18.1. 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.19. GitLogTask

Show commit logs. See official documentation [<http://www.kernel.org/pub/software/scm/git/docs/git-log.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
paths	String	<paentry> arguments to git-log. Accepts one/a or more paths delimited by PATH_SEPARA- TOR		No
outputProperty	String	Property name to set with output value from n/a git-log		No
format	String	Commit format. See --format of git-log. medium Can be one of oneline, short, medi- um, full, fuller, email, raw and for- mat:<string>		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.	n/a	No
stat	String	Generate a diffstat. See --stat of git-log	n/a	No
nameStatus	Boolean	Names + status of changed files. See --name-false status of git-log.		No
maxCount	Integer	Number of commits to show. See -<n> -n --n/a max-count of git-log.		No
noMerges	Boolean	Don't show commits with more than one par-false ent. See --no-merges of git-log.		No

C.19.1. 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.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.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
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
fastForward-Commit	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

Name	Type	Description	Default	Required
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.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	Fetch all remotes	false	No
source	String	The "remote" repository that is the source of aorigin fetch or pull operation. See <repository> in git-pull [http://www.kernel.org/pub/software/scm/git/docs/git-pull.html].		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

Name	Type	Description	Default	Required
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
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/foobranh and rebase when merging -->
<gitpull
  repository = "${repo.dir.resolved}"
  source = "origin" refspec = "foobranh"
  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.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
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].		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].		No
mirror	Boolean	See --mirror in git-push [http://www.kernel.org/pub/software/scm/git/docs/git-push.html].	false	No
delete	Boolean	Delete "remote" reference. Same as --delete 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:foobranh" 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.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
message	String	Use given tag message. See -m of git-n/a 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 fromn/a git-tag	n/a	No
file	String	Take tag message from given file. See -n/a 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. Seefalse -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. Seefalse -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. Seefalse -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.false See -l of git-tag [http://www.kernel.org/pub/software/scm/git/docs/git-tag.html]	false	No

Name	Type	Description	Default	Required
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. 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 line break		Yes
title	String	Title of notification	GrowlNotify	No
notification	String	The notification name/type	General Notification	No
appicon	String	<ul style="list-style-type: none"> • 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 : <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • absolute url (http://domain/image.png) 	embedded growl icon v2	No

Name	Type	Description	Default	Required
		<ul style="list-style-type: none"> absolute file path (c:\temp\image.png) relative file path (.\\folder\image.png) 		

C.24.1. 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">

  <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 GNTN, you should use the basic UDP protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "phing-GrowlNotifyTask" basedir = "." default = "notification">

  <target name = "notification"
    description = "display a single message with growl udp over network"
  >
    <growlnotify message = "Notify my MAC that does not accept GNTN."
      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 miss 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">

  <target name = "notification"
    description = "display a sticky message on desktop"
  >
    <growlnotify message = "Project LAMBDA, unit tests FAILED."
      priority = "high"
      sticky = "true"
    />
  </target>

</project>
```

```
</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">

  <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"
    </growlnotify>
  </target>
</project>
```

C.25. HgAddTask

Add files to Mercurial repository on the next commit. This is available for PHP 5.4 and higher.

Table C.26: Attributes

Name	Type	Description	Default	Required
repository	String	Path to Mercurial repository.	n/a	Yes

C.25.1. Example

```
<property name = "repo.dir" value = "../repo.directory" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />
<hgadd repository = "${repo.dir.resolved}">
  <fileset dir = "." />
</hgadd>
```

C.25.2. Supported Nested Tags

- fileset

C.26. HgArchiveTask

Create an unversioned archive of a Mercurial repository revision. This is available for PHP 5.4 and higher.

Table C.27: Attributes

Name	Type	Description	Default	Required
destination	String	Name of archive to create.	n/a	Yes
revision	String	Revision to distribute in the archive.	n/a	No

C.26.1. Example

```
<property name = "version" value = "v0_1_2" />
<hgarchive destination = "${version}.zip" />
<hgarchive destination = "${version}.tgz" />
```

C.27. HgCloneTask

Make a copy of an existing Mercurial repository. This is available for PHP 5.4 and higher.

Table C.28: Attributes

Name	Type	Description	Default	Required
insecure	Boolean	Do not verify server certificate.	false	No
repository	String	Path to Mercurial repository.	n/a	Yes
targetPath	String	Directory to clone into.	n/a	Yes
quiet	Boolean	Work silently unless an error occurs.	false	No

C.27.1. Example

```
<property name = "repo.dir" value = "./repo.directory" />
<property name = "repo.url" value = "https://bitbucket.org/spaetz/ceyx-mapcss" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />
<hgclone repository = "${repo.url}" quiet = "false" insecure = "true" targetPath = "${repo.dir.resolved}" />
```

C.28. HgCommitTask

Commit changes to a Mercurial repository. This is available for PHP 5.4 and higher.

Table C.29: Attributes

Name	Type	Description	Default	Required
message	String	Commit message.	n/a	Yes
quiet	Boolean	Work silently unless an error occurs.	false	No
repository	String	Path to Mercurial repository.	n/a	No

Name	Type	Description	Default	Required
user	String	User to record as the committer.	n/a	No

C.28.1. Example

```
<property name = "repo.dir" value = "../repo.directory" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />
<hgcommit message = "[ci skip] Compress .js files." user = "phingbot" repository = "${repo.dir}
```

C.29. HgInitTask

Create a new Mercurial repository. This is available for PHP 5.4 and higher.

Table C.30: Attributes

Name	Type	Description	Default	Required
insecure	Boolean	Do not verify server certificate.	false	No
quiet	Boolean	Work silently unless an error occurs.	false	No
repository	String	Path to Mercurial repository.	n/a	No

C.29.1. Example

```
<property name = "repo.dir" value = "../repo.directory" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />
<hginit repository = "${repo.dir.resolved}" />
```

C.30. HgLogTask

Show revision history of entire Mercurial repository or files, or limit to a number of revisions. Optionally store the history to a phing property. This is available for PHP 5.4 and higher.

Table C.31: Attributes

Name	Type	Description	Default	Required
format	String	Display with template, e.g. "{rev}\n", "{branch}" etc.	n/a	No
maxCount	Integer	Number of commits to show/limit.	n/a	No
outputProperty	String	Property name to set output value to from then/a execution.	n/a	No
repository	String	Path to Mercurial repository.	n/a	Yes

Name	Type	Description	Default	Required
revision	String	Show the specified revision or range.	n/a	Yes

C.30.1. Example

```
<property name = "repo.dir" value = "./repo.directory" />
<resolvepath propertyName = "repo.dir.resolved" file = "${repo.dir}" />
<hglog maxCount = "1" format = "{files}\n" outputproperty = "hgfiles" repository = "${repo.dir.re
```

C.31. HgPullTask

Pull changes from a specified Mercurial repository to a local one. This is available for PHP 5.4 and higher.

Table C.32: Attributes

Name	Type	Description	Default	Required
insecure	Boolean	Do not verify server certificate.	false	No
quiet	Boolean	Work silently unless an error occurs.	false	No
repository	String	Path to Mercurial repository.	n/a	No

C.31.1. Example

```
<hgpull quiet = "false" insecure = "true" repository = "${repo.dir}"/>
```

C.32. HgPushTask

Push changes from the local Mercurial repository to the specified destination. This is available for PHP 5.4 and higher.

Table C.33: Attributes

Name	Type	Description	Default	Required
insecure	Boolean	Do not verify server certificate.	false	No
quiet	Boolean	Work silently unless an error occurs.	false	No
repository	String	Path to Mercurial repository.	n/a	No

C.32.1. Example

```
<property name = "repo.dir" value = "./repo.directory" />
```

```
<hgpush haltonerror = "true" repository = "{repo.dir.resolved}"/>
```

C.33. HgRevertTask

Revert files to their checkout state from the Mercurial repository. This is available for PHP 5.4 and higher.

Table C.34: Attributes

Name	Type	Description	Default	Required
all	Boolean	Revert all Changes when no other details are given.	false	No
name	String	Name of file to revert.	n/a	No
quiet	Boolean	Work silently unless an error occurs.	false	No
revision	String	Revision to revert to.	n/a	No

C.33.1. Example

```
<hgrevert all = "true"/>
```

C.34. HgTagTask

Add a tag for the current or specified revision of the local Mercurial repository. This is available for PHP 5.4 and higher.

Table C.35: Attributes

Name	Type	Description	Default	Required
message	String	Message to add/edit tag with.	n/a	No
name	String	Name of tag.	n/a	Yes
repository	String	Path to Mercurial repository.	n/a	No
revision	String	Revision to tag.	n/a	No
user	String	User to record as the committer.	n/a	No

C.34.1. Example

```
<hgtag user = "phingbot" message = "tagging new release" name = "v0.1.2"/>
```

C.35. HgUpdateTask

Update the Mercurial repository's working directory or switch revisions. This is available for PHP 5.4 and higher.

Table C.36: Attributes

Name	Type	Description	Default	Required
branch	String	A specific branch to pull.	n/a	No
clean	Boolean	Discard uncommitted changes.	false	No
quiet	Boolean	Work silently unless an error occurs.	false	No
repository	String	Path to Mercurial repository.	n/a	Yes

C.35.1. Example

```
<property name = "repo.dir" value = "./repo.directory" />
<hgupdate repository = "${repo.dir.resolved}" branch = "dev" />
```

C.36. HttpGetTask

This task will download a file through HTTP GET and save it to a specified directory. You need an installed version of Guzzle [<http://docs.guzzlephp.org/en/stable/>] to use this task.

Table C.37: 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-No name part of the URL	No
followRedi- rects	Boolean	Whether to follow HTTP redirects	false	No
sslVerifyPeer	Boolean	Whether to verify SSL certificates	true	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
quiet	Boolean	If true, set default log level to false Project.MSG_ERR	false	No

C.36.1. Example

```
<httpget url = "http://buildserver.com/builds/latest.stable.tar.bz2" dir = "/usr/local/lib"/>
```

C.36.2. Supported Nested Tags

- config

Holds additional config data. See Guzzle documentation [<http://docs.guzzlephp.org/en/stable/request-options.html>] for supported values.

Table C.38: 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.39: Attributes

Name	Type	Description	Default	Required
name	String	Header name	n/a	Yes
value	String	Header value	n/a	Yes

C.36.3. Global configuration

In addition to configuring a particular instance of Guzzle via nested `<config>` tags it is also possible to set default configuration values for `HttpGetTask` / `HttpRequestTask` / `VisualizerTask` by setting `phing.http.*` properties.

```
<property name="phing.http.proxy" value="socks5://localhost:1080/" />
<!-- This request will go through the default proxy -->
<httpget url="http://example.com/file.zip" dir="." />
<httpget url="http://example.org/file.exe" dir=".">
  <!-- This proxy will be used instead of the default one -->
  <config name="proxy" value="http://foo:bar@proxy.example.org:3128/" />
</httpget>
```

C.37. 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 Guzzle [<http://docs.guzzlephp.org/en/stable/>] to use this task.

Table C.40: 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
responseCodeRegex	String	The regular expression for matching the response code	n/a	No
authUser	String	The authentication user name	n/a	No

Name	Type	Description	Default	Required
authPassword	String	The authentication password	n/a	No
authScheme	String	The authentication scheme	basic	No
verbose	Boolean	Whether to enable detailed logging	false	No
method	String	The HTTP method of the request, currently on-GET ly GET or POST supported		No

C.37.1. 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">
  <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.37.2. Supported Nested Tags

- config

Holds additional config data. See Guzzle documentation [<http://docs.guzzlephp.org/en/stable/request-options.html>] for supported values.

Table C.41: 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.42: Attributes

Name	Type	Description	Default	Required
<code>name</code>	String	Header name	n/a	Yes
<code>value</code>	String	Header value	n/a	Yes

- `postparameter`

Used when performing a POST request. Contains `name` and `value` of a form field.

Table C.43: Attributes

Name	Type	Description	Default	Required
<code>name</code>	String	Field name	n/a	Yes
<code>value</code>	String	Field value	n/a	Yes

C.37.3. Global configuration

In addition to configuring a particular instance of `Guzzle` via nested `<config>` tags it is also possible to set default configuration values for `HttpGetTask` / `HttpRequestTask` / `VisualizerTask` by setting `phing.http.*` properties.

```
<property name="phing.http.proxy" value="socks5://localhost:1080/" />
<!-- This request will go through the default proxy -->
<http-request url="http://example.com/foo" />
<http-request url="http://example.org/restricted" dir=".">
  <!-- This proxy will be used instead of the default one -->
  <config name="proxy" value="http://foo:bar@proxy.example.org:3128/" />
</http-request>
```

C.38. IniFileTask

The `IniFileTask` is inspired by the Ant-Contrib `IniFile` [<http://ant-contrib.sourceforge.net/tasks/tasks/inifile.html>] and can be used to build and edit `.ini` files. Unlike the Ant equivalent, it can also read values from different sections of an `.ini` file and set the retrieved values to specified properties.

Table C.44: Attributes

Name	Type	Description	Default	Required
<code>dest</code>	string	The name of the <code>.ini</code> file to write to. If not specified, the source file will be modified instead.	none	No
<code>haltOnError</code>	boolean	Should the build fail when problems occur?	false	No
<code>source</code>	string	The name of the <code>.ini</code> file to read from. If not specified, the dest file will be used instead.	none	No

C.38.1. Supported Nested Tags

- `get`

Use to read a value from a specific key and section of an .ini file

Table C.45: Attributes

Name	Type	Description	Default	Required
<code>default</code>	String	Value to return if section, property or value are not set	n/a	No
<code>section</code>	String	Name of the section.	n/a	Yes
<code>property</code>	String	Name of the key, in the specified section, to read	n/a	Yes
<code>outputpropertyString</code>		Name of the property to set the value to	n/a	Yes

- `remove`

Use to remove either a specific key or section from an .ini file

Table C.46: Attributes

Name	Type	Description	Default	Required
<code>section</code>	String	Name of the section.	n/a	Yes
<code>property</code>	String	Name of the key to remove. If not specified the entire section is removed.	n/a	No

- `set`

Use to set a key in a section to a specific value

Table C.47: Attributes

Name	Type	Description	Default	Required
<code>section</code>	String	Name of the section.	n/a	Yes
<code>property</code>	String	Name of the key/property.	n/a	Yes
<code>operation</code>	String	The operation to perform on the existing value, which must be numeric. Possible values are "+" and "-", which add and subtract 1, respectively from the existing value. If the value doesn't already exist, the set is not performed, triggering an error.	n/a	No
<code>value</code>	String	The new value for the property.	n/a	No, if operation is specified.

C.38.2. Example

```
<inifile
  haltonerror = "no"
  dest = "${project.basedir}/application/configs/application.ini">
  <set section = "production" property = "buildTimestamp" value = "${DSTAMP}${TSTAMP}" />
  <set section = "production" property = "buildNumber" operation = "+" />
  <remove section = "development : staging" />
```


</inifile>

C.39. 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.48: Attributes

Name	Type	Description	Default	Required
<code>allowedserver</code>	String	Restricts the encoded files to particular servers and/or domains. Consult the IonCube documentation for more information.	none	No
<code>binary</code>	Boolean	Whether to save encoded files in binary format (default is ASCII format)	false	No
<code>copy</code>	String	Specifies files or directories to exclude from being encoded or encrypted and copy them to the target directory (separated by space).	none	No
<code>encode</code>	String	Specifies additional file patterns, files or directories to encode, or to reverse the effect of <code>copy</code>	none	No
<code>encrypt</code>	String	Specify files or directories (space separated list) that are to be encrypted.	none	No
<code>expirein</code>	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
<code>expireon</code>	String	Sets a YYYY-MM-DD date to expire the files.	none	No
<code>fromdir</code>	String	Path containing source files	none	Yes
<code>ignore</code>	String	Set files and directories to ignore entirely and exclude from the target directory (separated by space).	none	Yes
<code>ioncubepath</code>	String	Path to the ionCube binaries	/usr/local/ioncube	No
<code>keep</code>	String	Set files and directories not to be ignored (separated by space).	none	No
<code>licensepath</code>	String	Path to the license file that will be used by the encoded files	none	No
<code>nodoccomments</code>	String	Omits documents comments (/* ... */) from the encoded files.	none	No
<code>obfuscationkey</code>	String	The obfuscation key must be supplied when using the obfuscate option	none	No
<code>obfuscate</code>	String	The Encoder can obfuscate the names of global functions, the names of local variables	none	No

Name	Type	Description	Default	Required
		in global functions, and line numbers. Use either all or any of functions, locals or linenos separated by a space.		
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 anone license file		No
phpversion	String	Defines which php encoder version will be5 used (suffix of the encoder file)		No
targetoption	String	Option to use when target directory exists, ac-cepts replace, merge, update and rename	none	No
todir	String	Path to save encoded files to	none	Yes
withoutrun- timeloadersup- port	Boolean	Whether to disable support for runtime initial-ization of the ionCube Loader	false	No
noshortopen- tags	Boolean	Whether to disable support for short PHP tags	false	No
callbackfile	String	Path to callback file (.php)	n/a	No
obfuscationex- clusionsfile	String	Path to obfuscation exclusions file	n/a	No
ignoredepre- catedwarnings	Boolean	Whether to ignore deprecated warnings	false	No
ignorestrict- warnings	Boolean	Whether to ignore strict warnings	false	No
allowencoding- intosource	Boolean	Whether to allow encoding into the source tree	false	No
mes- sageifnoloader	String	A valid PHP expression to customize the "non/a loader installed" message		No
action- ifnoloader	String	A valid PHP expression to replace the "non/a loader installed" action		No
showcommand- line	Boolean	whether to show command line before it is ex-ecuted	false	No

C.39.1. 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"
obfuscationexclusionsfile = "obfex.txt">
<comment>A project encoded with the ionCube encoder.</comment>
</ioncubeencoder>

```

C.39.2. Supported Nested Tags

- comment

Custom text that is added to the start of each encoded file.

C.40. 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.49: Attributes

Name	Type	Description	Default	Required
ioncubepath	String	Path to the ionCube binaries	/usr/ local/ion- cube	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 IonCube documenta- tion 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 li- cense.	none	No

C.40.1. 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.40.2. Supported Nested Tags

- comment

Custom text that is added to the start of each encoded file.

C.41. JsHintTask

This task runs JSHint [<http://www.jshint.com/>], a tool that helps to detect errors and potential problems in JavaScript code. JSHint 2.5.6+ is supported, although latest JSHint is recommended.

Table C.50: Attributes

Name	Type	Description	Default	Required
file	String	Single file to perform check on.	n/a	No, unless no fileset elements are present
haltOnError	boolean	Should the build fail when there are errors in the JS code?	false	No
haltOnWarning	boolean	Should the build fail when there are warnings in the JS code?	false	No
reporter	String	JSHint reporter.	checkstyle	No
checkstyleReportPath	String	Path where the the report in Checkstyle format should be saved.	n/a	No
config	String	JSHint config path.	n/a	No

C.41.1. Example

```
<jshint
  haltonerror = "false"
  haltOnWarning = "false"
  reporter = "jshint"
  checkstyleReportPath = "${project.basedir}/build/checkstyle-jshint.xml">
  <fileset dir = "${project.basedir}/public_html/www/js">
    <include name = "**/*.js"/>
    <exclude name = "js-cache/**"/>
  </fileset>
</jshint>
```

C.41.2. Supported Nested Tags

- fileset

C.42. JsMinTask

The `JsMinTask` minifies JavaScript files using JShrink [<https://github.com/tedivm/JShrink>], which can be installed using composer (Phing will try to use the composer autoloader)

Table C.51: Attributes

Name	Type	Description	Default	Required
<code>targetDir</code>	String	Path where to store minified JavaScript files	none	Yes
<code>suffix</code>	String	Suffix to append to the filenames.	-min	No
<code>failOnError</code>	Boolean	Whether an error while minifying a JavaScript file should stop the build or not	false	No

C.42.1. Example

```
<jsMin targetDir = "docroot/script/minified" failOnError = "false">
  <fileset dir = "docroot/script">
    <include name = "**/*.js"/>
  </fileset>
</jsMin>
```

C.42.2. Supported Nested Tags

- `fileset`
JavaScript files to be minified.

C.43. JsLintTask

The `JsLintTask` uses the Javascript Lint [<http://www.javascriptlint.com>] program to check the syntax on one or more JavaScript source code files.

NB: the Javascript lint program must be in the system path!

Table C.52: Attributes

Name	Type	Description	Default	Required
<code>executable</code>	String	Path to JSL executable	jsl	No
<code>file</code>	String	Path to source file	n/a	No, unless no fileset el- ements are present
<code>haltonfailure</code>	Boolean	Stop the build process if the linting process encounters an error.	false	No
<code>haltonwarning</code>	Boolean	Stop the build process if the linting process encounters a warning.	false	No
<code>showwarnings</code>	Boolean	Sets the flag if warnings should be shown.	true	No
<code>cachefile</code>	String	If set, enables writing of last-modified times to cachefile, to speed up processing of files that rarely change	none	No

Name	Type	Description	Default	Required
conffile	String	Path to JSL config file	none	No
tofile	String	File to write list of 'bad files' to.	n/a	No

C.43.1. Example

```
<jslint
    file = "path/to/source.js" />
```

Checking syntax of one particular source file.

```
<jslint>
  <fileset dir = "src">
    <include name = "**/*.js" />
  </fileset>
</jslint>
```

Check syntax of a fileset of source files.

C.43.2. Supported Nested Tags

- fileset

C.44. LiquibaseChangeLogTask

The `LiquibaseChangeLogTask` writes the Change Log XML to copy the current state of the database to the given `changeLogFile`.

Table C.53: 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. <code>jdbc:mysql://localhost/mydatabase</code>	n/a	Yes
display	Boolean	Whether to display the output of the command. Only used if <code>passthru</code> isn't true.	false	No
passthru	Boolean	Whether to use PHP's <code>passthru()</code> function instead of <code>exec()</code> . True by default for backwards compatibility. When true, the attributes	true	No

Name	Type	Description	Default	Required
		display, outputProperty and checkReturn are ignored.		
checkReturn	Boolean	Whether to check the return code of the execution, throws a BuildException when return-code != 0.	false	No
outputProperty	String	Property name to set output value to from then/a execution. Ignored if passthru attribute is true.		No

C.44.1. 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.44.2. Supported Nested Tags

- `parameter`
Same as for Section C.49, "LiquibaseTask".
- `property`
Same as for Section C.49, "LiquibaseTask".

C.45. LiquibaseDbDocTask

The `LiquibaseDbDocTask` generates a Javadoc-like documentation based on current database and the given changelog file.

Table C.54: 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 database data source, e.g. jdbc:mysql://localhost/mydatabase	n/a	Yes

Name	Type	Description	Default	Required
outputDir	String	Absolute path where the documentation getsn/a written to. If the given directory does not exist, it get's created automatically.		Yes
display	Boolean	Whether to display the output of the command.false Only used if passthru isn't true.		No
passthru	Boolean	Whether to use PHP's passthru() functiontrue instead of exec(). True by default for backwards compatibility. When true, the attributes display, outputProperty and checkReturnare ignored.		No
checkreturn	Boolean	Whether to check the return code of the exe-false cution, throws a BuildException when return-code != 0.		No
outputProperty	String	Property name to set output value to from then/a execution. Ignored if passthru attribute is true.		No

C.45.1. 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.45.2. Supported Nested Tags

- parameter

Same as for Section C.49, "LiquibaseTask".

- property

Same as for Section C.49, "LiquibaseTask".

C.46. 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.55: Attributes

Name	Type	Description	Default	Required
jar	String	Location of the Liquibase jar file.	n/a	Yes

Name	Type	Description	Default	Required
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 data-n/a base.		Yes
password	String	The password needed to connect to the data-n/a base.		Yes
url	String	The JDBC Url representing the databasen/a datasource, e.g jdbc:mysql://local-host/mydatabase		Yes
referenceUser- name	String	The username needed to connect to the refer-n/a ence database.		Yes
referencePass- word	String	The password needed to connect to the refer-n/a ence database.		Yes
referenceUrl	String	The JDBC Url representing the database ref-n/a erence datasource, e.g jdbc:mysql://lo-calhost/refdatabase		Yes
display	Boolean	Whether to display the output of the command.false Only used if passthru isn't true.		No
passthru	Boolean	Whether to use PHP's passthru() functiontrue instead of exec(). True by default for backwards compatibility. When true, the attributes display, outputProperty and checkRe-turnare ignored.		No
checkreturn	Boolean	Whether to check the return code of the exe-false cution, throws a BuildException when return-code != 0.		No
outputProperty	String	Property name to set output value to from then/a execution. Ignored if passthru attribute is true.		No

C.46.1. 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.46.2. Supported Nested Tags

- parameter

Same as for Section C.49, "LiquibaseTask".

- property

Same as for Section C.49, "LiquibaseTask".

C.47. LiquibaseRollbackTask

The `LiquibaseRollbackTask` rolls back the database to the state it was in when the tag was applied.

Table C.56: Attributes

Name	Type	Description	Default	Required
<code>jar</code>	String	Location of the Liquibase jar file.	n/a	Yes
<code>classpath</code>	String	Additional classpath entries.	n/a	Yes
<code>changeLogFile</code>	String	Location of the changelog file in which then/a changes get written or read from.		Yes
<code>username</code>	String	The username needed to connect to the data-n/a base.		Yes
<code>password</code>	String	The password needed to connect to the data-n/a base.		Yes
<code>url</code>	String	The JDBC Url representing the databasen/a datasource, e.g <code>jdbc:mysql://local-host/mydatabase</code>		Yes
<code>rollbackTag</code>	String	The name of the tag to roll the database backn/a to.		Yes
<code>display</code>	Boolean	Whether to display the output of the command.false Only used if <code>passthru</code> isn't true.		No
<code>passthru</code>	Boolean	Whether to use PHP's <code>passthru()</code> functiontrue instead of <code>exec()</code> . True by default for backwards compatibility. When true, the attributes <code>display</code> , <code>outputProperty</code> and <code>checkReturn</code> are ignored.		No
<code>checkreturn</code>	Boolean	Whether to check the return code of the exe-false cution, throws a <code>BuildException</code> when return-code <code>!= 0</code> .		No
<code>outputProperty</code>	String	Property name to set output value to from then/a execution. Ignored if <code>passthru</code> attribute is true.		No

C.47.1. 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.47.2. Supported Nested Tags

- `parameter`

Same as for Section C.49, “LiquibaseTask”.

- `property`

Same as for Section C.49, “LiquibaseTask”.

C.48. LiquibaseTagTask

The `LiquibaseTagTask` tags the current database state for future rollback.

Table C.57: Attributes

Name	Type	Description	Default	Required
<code>jar</code>	String	Location of the Liquibase jar file.	n/a	Yes
<code>classpath</code>	String	Additional classpath entries.	n/a	Yes
<code>changeLogFile</code>	String	Location of the changelog file in which then/a changes get written or read from.		Yes
<code>username</code>	String	The username needed to connect to the data-n/a base.		Yes
<code>password</code>	String	The password needed to connect to the data-n/a base.		Yes
<code>url</code>	String	The JDBC Url representing the databasen/a datasource, e.g <code>jdbc:mysql://local-host/mydatabase</code>		Yes
<code>tag</code>	String	The name of the tag to apply.	n/a	Yes
<code>display</code>	Boolean	Whether to display the output of the command.false Only used if <code>passthru</code> isn't true.		No
<code>passthru</code>	Boolean	Whether to use PHP's <code>passthru()</code> functiontrue instead of <code>exec()</code> . True by default for backwards compatibility. When true, the attributes <code>display</code> , <code>outputProperty</code> and <code>checkReturn</code> are ignored.		No
<code>checkreturn</code>	Boolean	Whether to check the return code of the exe-false cution, throws a <code>BuildException</code> when return-code != 0.		No
<code>outputProperty</code>	String	Property name to set output value to from then/a execution. Ignored if <code>passthru</code> attribute is true.		No

C.48.1. 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.48.2. Supported Nested Tags

- parameter

Same as for Section C.49, "LiquibaseTask".

- property

Same as for Section C.49, "LiquibaseTask".

C.49. LiquibaseTask

The `LiquibaseTask` is a generic task for liquibase commands that don't require extra command parameters. You can run commands like `updateSQL`, `validate` or `updateTestingRollback` with this task but not `rollbackToDateSQL` since it requires a date parameter after the command.

Table C.58: 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 data-n/a base.		Yes
password	String	The password needed to connect to the data-n/a base.		Yes
url	String	The JDBC Url representing the databasen/a datasource, e.g <code>jdbc:mysql://localhost/mydatabase</code>		Yes
command	String	What liquibase command to run. Currently on-n/a ly supports commands that doesn't require command parameters, such as <code>validate</code> and <code>updateSQL</code> .		Yes
display	Boolean	Whether to display the output of the command.false Only used if <code>passthru</code> isn't true.		No
passthru	Boolean	Whether to use PHP's <code>passthru()</code> functiontrue instead of <code>exec()</code> . True by default for back-		No

Name	Type	Description	Default	Required
		wards compatibility. When true, the attributes display, outputProperty and checkReturn are ignored.		
checkreturn	Boolean	Whether to check the return code of the execution, throws a BuildException when return-code != 0.	false	No
outputProperty	String	Property name to set output value to from then/a execution. Ignored if passthru attribute is true.		No

C.49.1. Example

```

<liquibase
  jar = "../vendor/alcaeus/liquibase/liquibase.jar"
  classpathref = "../libs/mysql-connector-java.jar"
  changelogFile = "../DB/master.xml"
  username = "${deploy.user}"
  password = "${deploy.password}"
  url = "jdbc:mysql://${database.host}/${database.name}"
  display = 'true'
  checkreturn = "true"
  passthru = 'false'
  outputProperty = "liquibase.updateSQL.output"
  command = "updateSQL"
>
  <parameter name = "logLevel" value = "info" />
  <property name = "tablename" value = "Person" />
</liquibase>

```

The nested parameters in the example above will result in the command:

```
--logLevel='info' updateSQL -Dtablename='Person'
```

C.49.2. Supported Nested Tags

- parameter

Use these nested parameter tags to set optional liquibase commands like --logLevel or --defaultsFile.

Table C.59: Attributes

Name	Type	Description	Default	Required
name	String	Name of the liquibase parameter. Do not include the '--'.	n/a	Yes
value	String	Value of the liquibase parameter.	n/a	Yes

- property

These tags are used to set what Liquibase calls "Change Log Properties" which are used for substitution in the change log(s). Note that they are not the same thing as regular Phing properties.

Table C.60: Attributes

Name	Type	Description	Default	Required
name	String	Name of the property. Do not include the '-D'.	n/a	Yes
value	String	Value of the property.	n/a	Yes

C.50. LiquibaseUpdateTask

The `LiquibaseUpdateTask` applies the latest changes from the changelog file to the defined database.

Table C.61: 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 data source, e.g. <code>jdbc:mysql://localhost/mydatabase</code>	n/a	Yes
display	Boolean	Whether to display the output of the command. Only used if <code>passthru</code> isn't true.	false	No
passthru	Boolean	Whether to use PHP's <code>passthru()</code> function instead of <code>exec()</code> . True by default for backwards compatibility. When true, the attributes <code>display</code> , <code>outputProperty</code> and <code>checkReturn</code> are ignored.	true	No
checkReturn	Boolean	Whether to check the return code of the execution, throws a <code>BuildException</code> when return code $\neq 0$.	false	No
outputProperty	String	Property name to set output value to from the execution. Ignored if <code>passthru</code> attribute is true.	n/a	No

C.50.1. 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.50.2. Supported Nested Tags

- `parameter`

Same as for Section C.49, “LiquibaseTask”.

- `property`

Same as for Section C.49, “LiquibaseTask”.

C.51. MailTask

A task to send email. Attachments are supported if the PEAR Mail package [<http://pear.php.net/package/Mail>] is installed.

Table C.62: *Attributes*

Name	Type	Description	Default	Required
<code>from</code>	String	Email address of sender.	none	Yes
<code>tolist</code>	String	Comma-separated list of recipients.	none	Yes
<code>message</code>	String	Message to send in the body of the email.	none	No
<code>subject</code>	String	Email subject line.	none	No
<code>backend</code>	String	PEAR Mail backend (see here [http://pear.php.net/manual/en/package.mail.mail.factory.php] for possible values).		No
<code>backendParams</code>	String	Comma-separated key-value pairs with backend specific parameters (see here [http://pear.php.net/manual/en/package.mail.mail.factory.php] for possible values).	none	No

C.51.1. Example

```
<mail tolist = "user@example.org" subject = "build complete">  
  The build process is a success...  
</mail>
```

C.51.2. Supported Nested Tags

- `fileset`

Files to be attached.

C.52. NotifySendTask

This is a wrapper for *notify-send*, a Linux program that sends desktop notifications to a notification daemon.

On Windows machines, this port [<http://vaskovsky.net/notify-send/>] may help.

Table C.63: Attributes

Name	Type	Description	Default	Required
icon	string	Specify an icon filename or stock icon to display.		No
message	String	Text to display. Use \n to specify a line break	n/a	Yes
title	String	Title, or summary, of the notification.	none	No

C.53. OpenTask

Open a file or URL in the user's preferred application.

`OpenTask` will not fail if `path` attribute is invalid. Therefore, this task must not be used to check the validity of a file or URL.

Table C.64: Attributes

Name	Type	Description	Default	Required
path	String	File path or URL to open.	n/a	Yes

C.53.1. Examples

Open `http://localhost:8080/` on your browser.

```
<open path = "http://localhost:8080/" />
```

Open `./docs/images/screenshot.png` on your image viewer.

```
<open path = "./docs/images/screenshot.png" />
```

C.54. PDOSQLException

The `PDOSQLException` executes SQL statements using PDO.



Note

The combination of large SQL files and `delimiter` type set to `normal` can trigger segmentation faults with large files.

Table C.65: 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, abort or abort)	none	No
failonconnectionerror	Boolean	If false, will not execute any statement if the task fails to connect to the database.	true	No
delimiter	String	The delimiter to separate SQL statements; (e.g. "GO" in MSSQL)	none	No
delimitertype	String	The delimiter type ("normal", "row" or "none"). Normal means that any occurrence of the delimiter terminate the SQL command whereas with row, only a line containing just the delimiter is recognized as the end of the command. None disables all delimiter detection.	none	No
autocommit	Boolean	Whether to auto (implicitly) commit every single statement, disabling transactions.	false	No
encoding	String	Encoding to use for read SQL files	none	No
keepformat	Boolean	Control whether the format of SQL will be preserved. Useful when loading packages and procedures.	false	No
expandproperties	Boolean	Set to false to turn off property expansion in nested SQL, inline in the task or nested transactions.	true	No
errorproperty	String	The name of a property to set in the event of an error.	none	No
statement-countproperty	String	The name of a property to set to the number of statements executed successfully.	none	No

You can also use `PDOSQLExecTask` as condition

C.54.1. 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>
```

```
<property name = "color" value = "orange"/>
<pdosqlexec url = "mysql:host=localhost;dbname=test"
  userid = "username" password = "password">
  <transaction>
    SELECT * FROM products WHERE color = '${color}';
  </transaction>
  <formatter type = "xml" outfile = "path/to/output.xml"/>
</pdosqlexec>
```



Note

Because of backwards compatibility, 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.54.2. 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.66: Attributes

Name	Type	Description	Default	Required
src	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\Task\System\Pdo\PDOResultFormatter` can be specified.

Table C.67: Attributes

Name	Type	Description	Default	Required
<code>type</code>	String	Use a predefined formatter (either <code>xml</code> or <code>plain</code>).	n/a	One of these attributes is required.
<code>classname</code>	String	Name of a custom formatter class (must extend <code>Phing\Task\System\Pdo\PDOResultFormatter</code>).	n/a	
<code>usefile</code>	Boolean	Boolean that determines whether output should be sent to a file.	<code>true</code>	No
<code>outfile</code>	File	Path to file in which to store result.	Depends on formatter	No
<code>append</code>	Boolean	Whether output should be appended to or overwrite an existing file.	<code>false</code>	No
<code>showheaders</code>	Boolean	(only applies to plain formatter) Whether to show column headers.	<code>false</code>	No
<code>showtrailers</code>	Boolean	(only applies to plain formatter) Whether to show successful executed statement counter trailers.	<code>false</code>	No
<code>coldelim</code>	String	(only applies to plain formatter) The column delimiter.		No
<code>rowdelim</code>	String	(only applies to plain formatter) The row delimiter.	<code>\n</code>	No
<code>encoding</code>	String	(only applies to XML formatter) The XML document encoding.	(PHP default)	No
<code>formatoutput</code>	Boolean	(only applies to XML formatter) Whether to format XML output.	<code>true</code>	No

C.55. 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.

NB: if you have installed the PHPMD Phar file, make sure you set the `pharLocation` attribute!

Table C.68: Attributes

Name	Type	Description	Default	Required
<code>file</code>	String	Path to source file or path	n/a	Only when there are no

Name	Type	Description	Default	Required nested file- set elements
rulesets	String	Sets the rulesets used for analyzing thecodesize, No source code	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, .bzz, .hg	No
format	String	The format for the report when no nested for-text matter is used.		No
pharlocation	String	Location of the PHPMD Phar file.	n/a	No
cacheFile	String	If set, enables writing of last-modified times to cacheFile, to speed up processing of files that rarely change	none	No

C.55.1. 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.55.2. 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.69: Attributes

Name	Type	Description	Default	Required
<code>type</code>	String	The output format. Accepts the same valuesn/a as the <code>format</code> attribute (<code>xml</code> , <code>html</code> , <code>text</code>).		Yes
<code>usefile</code>	Boolean	Boolean that determines whether outputtrue should be sent to a file.		No
<code>outfile</code>	String	Path to write output file to.	n/a	Yes

C.56. PHPStanTask

The `PHPStanTask` executes `PHPStan` - a PHP static analysis tool - with given configuration.

Table C.70: Base attributes

Name	Type	Description	Default	Required
<code>command</code>	String	PHPStan command name	<code>analyse</code>	No
<code>executable</code>	String	Path to PHPStan executable	<code>phpstan</code>	No
<code>checkReturn</code>	Boolean	Whether to check the return code.	<code>false</code>	No
<code>passthru</code>	Boolean	Whether to echo PHPStan's output on thefalse console.		No

Table C.71: Analyse command attributes

Name	Type	Description	Default	Required
<code>configuration</code>	String	Path to configuration		No
<code>level</code>	String	Analyse level		No
<code>noProgress</code>	String	NO progress flag	<code>false</code>	No
<code>debug</code>	String	Debug flag	<code>false</code>	No
<code>autoloadFile</code>	String	Path to autoload file		No
<code>errorFormat</code>	String	Error format		No
<code>memoryLimit</code>	String	Memory limit		No
<code>paths</code>	String	Paths (space separated)		No

Table C.72: List command attributes

Name	Type	Description	Default	Required
<code>format</code>	String	Help format		No
<code>raw</code>	String	Raw flag	<code>false</code>	No
<code>namespace</code>	String	Namespace		No

Table C.73: Help command attributes

Name	Type	Description	Default	Required
format	String	Help format		No
raw	String	Raw flag	false	No
commandName	String	Command name		No

Table C.74: Common attributes

Name	Type	Description	Default	Required
help	String	Help flag	false	No
quiet	String	Quiet flag	false	No
version	String	Version flag	false	No
ansi	String	ANSI flag	false	No
noAnsi	String	No ANSI flag	false	No
noInteraction	String	No interaction flag	false	No
verbose	String	Verbose flag	false	No

C.56.1. Supported Nested Tags

- fileset

C.56.2. Example

```
<phpstan
  command = "analyse"
  configuration = "anyConfiguration"
  level = "anyLevel"
  noProgress = "true"
  debug = "true"
  autoloadFile = "anyAutoloadFile"
  errorFormat = "anyErrorFormat"
  memoryLimit = "anyMemoryLimit"
  paths = "path1 path2"
/>
```

```
<phpstan command = "analyse">
  <fileset refid = "files-to-analyse" />
</phpstan>
```

C.57. PHPUnitReport

This task transforms PHPUnit xml reports to HTML using XSLT.

Table C.75: Attributes

Name	Type	Description	Default	Required
infile	String	The filename of the XML results file to use.	test-suites.xml	No
format	String	The format of the generated report. Must be <code>noframes</code> or <code>frames</code> .		No
styledir	String	The directory where the stylesheets are located. They must conform to the following conventions: <code>frames</code> format: the stylesheet must be named <code>phpunit-frames.xsl</code> . <code>noframes</code> format: the stylesheet must be named <code>phpunit-noframes.xsl</code> . 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).		No
todir	String	An existing directory where the files resulting from the transformation should be written to.		Yes
usesorttable	Boolean	Whether to use the sortable JavaScript library (see http://www.kryogenix.org/code/browser/sorttable/)	false	No

C.57.1. 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/test-suites.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.58. 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: if you want to use the PHPUnit .phar file, please make sure you download the library version (`phpunit-library.phar`) and you set the `pharlocation` attribute!

Table C.76: 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 none executing the tests.		No

Name	Type	Description	Default	Required
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
haltondefect	Boolean	Stop the build process if a test has failures, errors or warnings.	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
haltonwarning	Boolean	Stop the build process if any warnings are encountered.	false	No
haltonrisky	Boolean	Stop the build process if any risky 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
warningproperty	String	Name of property to set (to true) on warnings.	n/a	No
riskyproperty	String	Name of property to set (to true) on risky tests.	n/a	No
usecustomererrorhandler	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 <code>phpunit.xml</code>). Supported elements are: <code>bootstrap</code> , <code>processIsolation</code> , <code>stopOnFailure</code> , <code>stopOnError</code> , <code>stopOnIncomplete</code> and <code>stopOnSkipped</code> . Values provided overwrite other attributes!	n/a	No
groups	String	Only run tests from the specified group(s).	n/a	No
excludeGroups	String	Exclude tests from the specified group(s).	n/a	No
pharlocation	String	Location of the PHPUnit PHAR package.	n/a	No

C.58.1. 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 six predefined formatters. `xml`, `clover`, and `crap4j` print the test results in the JUnit, Clover, and Crap4J XML formats respectively. The `clover-html` formatter prints code coverage details to a set of HTML files. The `plain` formatter emits a short statistics line for all test cases. The `summary` formatter print the same statistics as the plain formatter but only to the log output. Custom formatters that implement `Phing\Task\Ext\Formatter\PHPUnitResultFormatter` can be specified.

Table C.77: Attributes

Name	Type	Description	Default	Required
<code>type</code>	String	Use a predefined formatter (either <code>xml</code> , <code>n/a</code> , <code>plain</code> , <code>clover</code> , <code>clover-html</code> , <code>crap4j</code> , or <code>summary</code>).		One of these is required.
<code>classname</code>	String	Name of a custom formatter class.	<code>n/a</code>	
<code>usefile</code>	Boolean	Boolean that determines whether output should be sent to a file.	<code>true</code>	No
<code>todir</code>	String	Directory to write the file to.	<code>n/a</code>	No
<code>outfile</code>	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.78: Attributes

Name	Type	Description	Default	Required
<code>exclude</code>	String	A list of classes to exclude from the pattern matching. For example, when you have two baseclasses <code>BaseWebTest</code> and <code>BaseMathTest</code> , which are included a number of testcases (and thus added to the list of test-classes), you can exclude those classes from the list by typing <code>exclude="BaseWebTest BaseMathTest"</code> .	<code>n/a</code>	No
<code>classpath</code>	String	Used to define more paths on which - besides the PHP <code>include_path</code> - to look for the test files.	<code>n/a</code>	No
<code>name</code>	String	The name that is used to create a testsuite from this batchtest.	<code>Phing Batchtest</code>	No

C.58.2. 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.58.3. Supported Nested Tags

- `fileset`

C.59. ParallelTask

Executes nested tasks in parallel.

Parallel tasks have a number of uses in a Phing build file including:

- Taking advantage of available processing resources to execute external programs simultaneously.
- Testing servers, where the server can be run in one thread and the test harness is run in another thread.

Any valid Phing task may be embedded within a parallel task, including other parallel tasks.

While the tasks within the parallel task are being run, the main thread will be blocked waiting for all the child threads to complete. If one of the tasks within the `parallel` task fails, the remaining tasks will continue to run until all tasks have completed. In this situation, the parallel task will also fail.

The `threadCount` attribute can be used to place a maximum number of available threads for the execution. When not present the value is based on the number of processors present. When present then the maximum number of concurrently executing tasks will not exceed the number of threads specified. Furthermore, each task will be started in the order they are given. But no guarantee is made as to the speed of execution or the order of completion of the tasks, only that each will be started before the next.



Warning

This task is highly experimental, and will only work on *nix machines that have the PHP `pcntl` extension installed.



Warning

In some cases, such as when running this task from the Phing's `.phar` bundle, stability issues can occur. See this issue [<https://github.com/phingofficial/guide/issues/51>] for more details.

Table C.79: Attributes

Name	Type	Description	Default	Required
<code>threadCount</code>	Integer	Maximum number of threads / processes use.	ton/a	No

C.59.1. Example

```
<parallel threadCount = "4">
    <echo>Job 1</echo>
    <echo>Job 2</echo>
    <echo>Job 3</echo>
    <echo>Job 4</echo>
</parallel>
```

C.60. 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.80: Attributes

Name	Type	Description	Default	Required
<code>patchfile</code>	String	File that includes the diff output	n/a	Yes
<code>originalfile</code>	String	File to patch. If not specified Task tries to guess it from the diff file	tonone	No
<code>destfile</code>	String	File to send the output to instead of patching the file in place	none	No
<code>backups</code>	Boolean	Keep backups of the unpatched files	false	No
<code>quiet</code>	Boolean	Work silently unless an error occurs	false	No

Name	Type	Description	Default	Required
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 command	none	No
haltonfailure	Boolean	Stop the build process if the patching process encounters an error.	false	No
forward	Boolean	Ignore patches that appear to be reversed or already applied.	false	No
fuzz	String	Set the fuzz factor to LINES for inexact matching.	n/a	No

C.60.1. Example

```
<patch
  patchfile = "/path/to/patches/file.ext.patch"
  dir = "/path/to/original"
/>
```

Apply "file.ext.path" to original file located in "/path/to/original" folder.

C.61. PharDataTask

PharData [<http://php.net/manual/en/class.phardata.php>] archives generating with Phing. This task requires 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.81: Attributes

Name	Type	Description	Default	Required
basedir	String	Base directory, which will be deleted from each included file (from path). Paths with deleted basedir part are local paths in archive.	n/a	Yes
destfile	String	Destination (output) file. Will be recreated, if it exists!	n/a	Yes
compression	String	Compression type (gzip, bzip2, none) to apply to the archive.	none	No

C.61.1. Example

Sample build command:

```

<phardata
  destfile = "./build/archive.tar"
  basedir = "."
  compression = "gzip">
  <fileset dir = "./classes">
    <include name = "**/*" />
  </fileset>
</phardata>

```

C.61.2. Supported Nested Tags

- fileset

C.62. 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.82: 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
compression	String	Compression type (gzip, bzip2, none) to applynone to the packed files.		No
webstub	String	Relative path within the phar package to run,n/a if accessed through a web browser.		No
clistub	String	Relative path within the phar package to run,n/a if accessed on the command line.		No
stub	String	A path to a php file that contains a custom stubn/a		No
alias	String	An alias to assign to the phar package	n/a	No
signature	String	Signature algorithm (md5, sha1, sha256,sha1 sha512), used for this package.		No
key	String	The private key to sign the phar package withn/a (PEM or PKCS#12 encoded)		No
keyPassword	String	The password to use for the private key	n/a	No

C.62.1. 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.62.2. Supported Nested Tags

- fileset
- metadata

C.63. 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.83: 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
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 <code>php_strip_whitespace()</code>) 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.63.1. 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.63.2. Supported Nested Tags

- webaccess

Collection of path tags (see example below), that will be visible outside package in web mode.

C.64. PhpCSTask

This task runs PHP_CodeSniffer Version 3+ [http://pear.php.net/package/PHP_CodeSniffer] to detect violations of a defined set of coding standards.

Table C.84: Attributes

Name	Type	Description	Default	Required
file	String	File or directory to check.	n/a	Yes
bin	String	Path to phpcs binary.	phpcs	No
standard	String	The list of coding standards to test against. Separated by space, comma or semicolon.		No
format	String	The output format. (ex checkstyle, full, summary, ...)		No
outfile	String	Filename to write output/report to. If not set n/a output will be sent to STDOUT.		No
cache	Boolean	Cache results between runs.	false	No
ignoreAnnotations	Boolean	Ignore all phpcs annotations in code comments.	false	No
checkreturn	Boolean	Whether to check the return code.	false	No
level	String	Set the log level of generated messages. info Change this to verbose, if you only want output in verbose mode for example. Valid log levels are one of debug, info, verbose, warning or error		No

C.64.1. Supported Nested Tags

- FileSet
- 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.85: Attributes

Name	Type	Description	Default	Required
type	String	The output format. Accepts the same values as the <code>format</code> attribute (default, xml, checkstyle, csv, report, summary & doc).	n/a	Yes
outfile	String	Path to write output file to.	n/a	Yes

C.64.2. Examples

```
<phpcs bin = "bin/phpcs" file = "classes" checkreturn = "true"/>
```

C.65. 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.

NB: if you have installed the PHP_Depend Phar file, make sure you set the `pharLocation` attribute!

Table C.86: 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
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
pharlocation	String	Location of the PHP_Depend Phar file.	n/a	No

C.65.1. 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.65.2. 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.87: Attributes

Name	Type	Description	Default	Required
type	String	The name of the logger. Valid log-n/a gers are: jdepend-chart, jdepend-xml, overview-pyramid, phpunit-xml and summary-xml.		Yes
outfile	String	Path to write output file to.	n/a	Yes

- analyzer

Some additional analyzers can be added to the runner.

Table C.88: Attributes

Name	Type	Description	Default	Required
type	String	The name of the analyzer. Valid analyzersn/a are: coderank-mode.		Yes
value	String	The value for the analyzer.	n/a	Yes

C.66. PhpDocumentor2Task

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.89: 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
template	String	Name of the documentation template to use.	responsive-twig	No
defaultPackageName	String	Name of the default package.	Default	No
pharlocation	String	Location of the phpDocumentor PHAR package.	n/a	No

C.66.1. Example

```
<phpdoc2 title = "API Documentation"
          destdir = "apidocs"
          template = "responsive-twig">
  <fileset dir = "./classes">
    <include name = "**/*.php" />
  </fileset>
</phpdoc2>
```

C.66.2. Supported Nested Tags

- `fileset` - Files that should be included for parsing

C.67. rSTTask

Renders rST (reStructuredText) files into different output formats.

This task requires the `python docutils` installed. They contain `rst2html`, `rst2latex`, `rst2man`, `rst2odt`, `rst2s5`, `rst2xml`.

Table C.90: Attributes

Name	Type	Description	Default	Required
file	String	rST input file to render	n/a	Yes (or fileset)
format	String	Output format: <ul style="list-style-type: none"> • html • latex • man • odt • s5 • xml 	html	No
destination	String	Path to store the rendered file to. Used as directory if it ends with a <code>/</code> .	magically deter-	No

Name	Type	Description	Default	Required
			mined from input file	
uptodate	Boolean	Only render if the input file is newer than the target file	false	No
toolpath	String	Path to the rst2* tool	deter- mined from for- mat	No
toolparam	String	Additional commandline parameters to the rst2* tool	then/a	No
mode	Integer	The mode to create directories with.	From umask	No

C.67.1. 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.67.2. 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.67.3. Supported Nested Tags

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

C.68. 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.91: 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.68.1. Example

Downloading an object

```
<s3get object = "file.txt" target = "${project.basedir}" bucket = "mybucket"
key = "AmazonKey" secret = "AmazonSecret" />
```

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.69. 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.92: 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)

Name	Type	Description	Default	Required
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	stream	No
fileNameOnly	Boolean	Whether filenames should contain paths when uploaded to a bucket	false	No

C.69.1. Example

Uploading a file

```
<s3put source = "/path/to/file.txt" object = "file.txt" bucket = "mybucket"
key = "AmazonKey" secret = "AmazonSecret" />
```

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.69.2. Supported Nested Tags

- fileset

C.70. SassTask

The `SassTask` converts SCSS or Sass files to CSS using either the 'sass' gem [http://sass-lang.com/documentation/file.SASS_REFERENCE.html#using_sass] or the `scssphp` package [<https://scssphp.github.io/scssphp/>].

Table C.93: Attributes

Name	Type	Description	Default	Required
<code>check</code>	Boolean	Whether to just check the syntax of the input files.	False	No
<code>compact</code>	Boolean	Set the style to compact.	False	No
<code>compressed</code>	Boolean	Set the style to compressed.	False	No
<code>crunched</code>	Boolean	Set the style to crunched. Supported by <code>scssphp</code> , not <code>sass</code> .	False	No
<code>expand</code>	Boolean	Set the style to expanded.	False	No
<code>encoding</code>	String	Default encoding for input files. Supported by <code>utf-8</code> <code>scssphp</code> .		No
<code>executable</code>	String	Location/name of the sass executable, if required.		No
<code>extfilter</code>	String	Extension to filter against.	n/a	No
<code>failonerror</code>	Boolean	Whether to fail/halt if an error occurs.	False	No
<code>file</code>	String	Name of single file to process.	N/A	No
<code>flags</code>	String	Additional flags to set for sass executable.	n/a	No
<code>input</code>	String	Name of single file to process. Synonym for <code>file</code> .	N/A	No
<code>keepsubdirectories</code>	Boolean	Whether to keep the directory structure when compiling.	True	No
<code>linenumbers</code>	Boolean	Whether to annotate generated CSS with source file and line numbers.	False	No
<code>nested</code>	Boolean	Set the style to expanded.	true	No
<code>newext</code>	String	Extension for newly created files.	css	No
<code>nocache</code>	Boolean	Whether to cache parsed sass files.	n/a	No
<code>output</code>	String	Corresponding output file for 'file'/'input' parameter. If not specified and <code>outputpath</code> is, then the generated file is placed there, with the filename based on the input file. If neither is specified, then the generated file is placed into the directory that the input file is in.	N/A	No
<code>outputpath</code>	String	Where to place the generated CSS files.	n/a	Yes
<code>path</code>	String	Specify sass import path. e.g. <code>--load-path ...</code>	n/a	No
<code>removeoldext</code>	Boolean	Whether to strip existing extension off the output filename.	True	No
<code>style</code>	String	Name of style to output. Must be one of 'nested', 'compact', 'compressed', 'crunched' or 'expanded'.		No

Name	Type	Description	Default	Required
		panded'. 'Helper' attributes may also be used. 'crunched' is supported by scssphp only.		
trace	Boolean	Whether to show a full stack trace on error.	False	No
unixnewlines	Boolean	Use Unix-style newlines in written files.	True	No
useSass	Boolean	Whether to use the 'sass' command line tool. Takes precedence over scssphp if both are available and enabled.	True	No
useScssphp	Boolean	Whether to use the 'scssphp' PHP package.	True	No

The useSass and useScssphp attributes can be used to indicate which compiler should be used, which would be useful if both are available. If both are available and enabled, then the 'sass' compiler is used rather than the scssphp library.

C.70.1. Example

```
<sass style = "compact" trace = "yes" unixnewlines = "yes" outputpath = "${compiled.dir.resolve}
  <fileset dir = "."/>
</sass>
```

C.70.2. Supported Nested Tags

- fileset

C.71. 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.94: 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

Name	Type	Description	Default	Required
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.71.1. 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.71.2. 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.95: Attributes

Name	Type	Description	Default	Required
kex	String	List of key exchange methods to advertise, n/a comma separated in order of preference.		No
hostkey	String	List of hostkey methods to advertise, comen/a separated in order of preference.		No

Name	Type	Description	Default	Required
client	Nested Tag	Element containing attributes crypt, comp,n/a and mac method preferences for messages sent from client to server. All attributes are optional.		No
server	Nested Tag	Element containing attributes crypt, comp,n/a and mac method preferences for messages sent from server to client. All attributes are optional.		No

C.72. SmartyTask

A task for generating output by using Smarty.

Table C.96: Attributes

Name	Type	Description	Default	Required
controlTemplate	String	The control template used to generate the output.	none	Yes
templatePath	String	The path where Smarty will look for templates.	none	Yes
outputDirectory	String	The output directory, will be created if it doesn't exist.	none	Yes
compilePath	String	The path Smarty uses as a "cache" for compiled templates.	none	No
forceCompile	Boolean	Whether Smarty should always recompile templates.	false	No
configPath	String	The path where Smarty will look for config files.	none	No
leftDelimiter	String	The template left delimiter.	none	No
rightDelimiter	String	The template right delimiter.	none	No
contextProperties	String	The path to a property file that will be fed into the initial template context.	none	No

C.73. SonarTask

This task runs SonarQube Scanner [<http://www.sonarqube.org/>], a tool for code analysis and *continuous inspection*.

Table C.97: Attributes

Name	Type	Description	Default	Required
executable	String	Fully-qualified path of SonarQube Scanner executable. If executable is in <i>PATH</i> environment		Yes

Name	Type	Description	Default	Required
		ment variable, the executable name is sufficient.		
configuration	String	Path of configuration file. The file format is that of a properties file (as used by Java), i.e. a list of key-value pairs <key>=<value>.		No
errors	String	Sets errors flag of SonarQube Scanner. Allowed values are "true", "false", "yes", "no", "1", and "0".	Al-false	No
debug	String	Sets debug flag of SonarQube Scanner. Allowed values are "true", "false", "yes", "no", "1", and "0".	Al-false	No

C.73.1. Examples

Minimal Example

This example assumes that the SonarQube Scanner is called *sonarqube-scanner* and is available on the *PATH*.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "sonar-minimal-example" default = "sonar">
  <sonar executable = "sonarqube-scanner">
    <property name = "sonar.projectKey" value = "my-unique-project-key" />
    <property name = "sonar.projectName" value = "Foo Project" />
    <property name = "sonar.projectVersion" value = "0.1.0" />
    <property name = "sonar.sources" value = "src" />
  </sonar>
</project>
```

Full Example

This example consists of two files – *build.xml* and *sonar-project.properties*.

The *build.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name = "sonar-full-example" default = "sonar">
  <sonar
    executable = "path/to/sonarqube-scanner"
    errors = "true"
    debug = "true"
    configuration = "path/to/sonar-project.properties"
  >
    <!-- Assume that mandatory SonarQube parameters are defined in configuration file! -->
    <property name = "sonar.log.level" value = "DEBUG" />
  </sonar>
</project>
```

The configuration file *path/to/sonar-project.properties*:

```
sonar.projectKey      = my-unique-project-key
sonar.projectName     = Foo Project
sonar.projectVersion  = 0.1.0
```

```
sonar.sources = src
```

C.73.2. Supported Nested Tags

- `property`

Analysis parameters of SonarQube Scanner can be defined in a configuration file or via nested `property` elements. If both a configuration file and property elements are provided, the properties are merged. Values from `property` elements overwrite values from the configuration file if their property keys are equal.

Table C.98: Attributes

Name	Type	Description	Default	Required
name	String	Name of property.	n/a	Yes
value	String	Value of property.	n/a	Yes

C.74. 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.99: 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
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.74.1. Example

```
<ssh username = "john" password = "smith"
host = "webserver" command = "ls" />
```

Executes a single command on the remote server.

C.74.2. 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.100: Attributes

Name	Type	Description	Default	Required
kex	String	List of key exchange methods to advertise,n/a comma separated in order of preference.		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,n/a and mac method preferences for messages sent from client to server. All attributes are optional.		No
server	Nested Tag	Element containing attributes crypt, comp,n/a and mac method preferences for messages sent from server to client. All attributes are optional.		No

C.75. StopwatchTask

The StopwatchTask provides an easy way to measure execution time of phing tasks.

Table C.101: Attributes

Name	Type	Description	Default	Required
name	String	Name of the timer.		Yes
category	String	Set a category for the timer.		No
action	String	Action could be one of start, stop or lap	start	No

C.75.1. Example

```
<stopwatch name = "test" />
```

```

<!-- some other task... -->
<stopwatch name = "test" action = "lap" />
<!-- some other task... -->
<stopwatch name = "test" action = "lap" />
<!-- some other task... -->
<stopwatch name = "test" action = "lap" />
<!-- some other task... -->
<stopwatch name = "test" action = "stop" />

```

C.76. SvnCheckoutTask

The SvnCheckoutTask checks out a Subversion repository to a local directory.

Table C.102: 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
depth	String	Limit operation by depth	empty, files, immediates or infinity	No
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCertificate	Boolean	Trust self-signed certificates	false	No
configOption	String	Override subversion's config option	n/a	No

C.76.1. 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.77. 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.103: 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
depth	String	Limit operation by depth	empty, files, immediates or infinity	No
workingcopy	String	Working copy	none	Yes
message	String	The commit message	none	Yes
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCertificate	Boolean	Trust self-signed certificates	false	No
propertyname	String	Name of property to set to the last committed revision number	svn.committedrevision	No
configOption	String	Override subversion's config option	n/a	No

C.77.1. 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.78. SvnCopyTask

The SvnCopyTask duplicates something in a working copy or repository, remembering history.

Table C.104: 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
depth	String	Limit operation by depth	empty, files, immediates or infinity	No
trustServerCertificate	Boolean	Trust self-signed certificates	false	No
configOption	String	Override subversion's config option	n/a	No

C.78.1. 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.79. SvnExportTask

The SvnExportTask exports a Subversion repository to a local directory.

Table C.105: 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
repositoryurl	String	URL of SVN repository	none	Yes

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 export to	none	Yes
depth	String	Limit operation by depth	empty, files, immediate or infinity	No
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCertificate	Boolean	Trust self-signed certificates	false	No
configOption	String	Override subversion's config option	n/a	No

C.79.1. Example

```
<svnexport
  svnpath = "/usr/bin/svn"
  username = "anony"
  password = "anony"
  force = "true"
  nocache = "true"
  repositoryurl = "svn://localhost/project/trunk/"
  todir = "/home/user/svnwc"
  configoption = "config:miscellany:use-commit-times=yes" />
```

```
<svnexport
  svnpath = "C:/Subversion/bin/svn.exe"
  repositoryurl = "svn://localhost/project/trunk/"
  todir = "C:/projects/svnwc"/>
```

C.80. 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.106: 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

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
propertyname	String	Name of property to use	svn.info	No
element	String	Sets whether to store actual last changed revision of the directory/file mentioned	url	No
subelement	String	Sets whether to force compatibility with older SVN versions (< 1.2)	none	No
configOption	String	Override subversion's config option	n/a	No

C.80.1. 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.81. SvnLastRevisionTask

The SvnLastRevisionTask stores the number of the last revision of a Subversion workingcopy in a property.

Table C.107: 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 revision of the directory/file mentioned	false	No
configOption	String	Override subversion's config option	n/a	No

C.81.1. 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.82. 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.108: 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
limit	Integer	Limits the number of items to get back from/a the command		No
orderDescending	Boolean	Sets whether to reverse the order of the listed items	false	No
configOption	String	Override subversion's config option	n/a	No

C.82.1. 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"
  orderDescending = "true" limit = "10" />
```

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.83. 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.109: Attributes

Name	Type	Description	Default	Required
<code>svnpath</code>	String	Path to Subversion binary	<code>/usr/bin/svn</code>	No
<code>workingcopy</code>	String	Working copy directory	none	One of the two
<code>repositoryurl</code>	String	URL of remote repository	none	
<code>username</code>	String	A username used to connect to the SVN server	none	No
<code>password</code>	String	A password used to connect to the SVN server	none	No
<code>propertyname</code>	String	Name of property to use	<code>svn.list</code>	No
<code>limit</code>	Integer	Limits the number of items to get back from the command	n/a	No
<code>configOption</code>	String	Override subversion's config option	n/a	No

C.83.1. 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 = "10" />
```

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.84. SvnPropgetTask

The `SvnPropgetTask` gets a property on files, dirs, or revisions from the working copy.

Table C.110: 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
propertyname	String	Name of property to use.	svn.propget	No
svnproperty-name	String	The svn property to get.	none	Yes
fromdir	String	Thr dir the properties are from.	none	Yes
configOption	String	Override subversion's config option	n/a	No

C.84.1. Example

```
<svnpropget
    svnpath = "/usr/bin/svn"
    username = "anony"
    password = "anony"
    repositoryurl = "http://svn.phing.info/tags/2.4.2"
    fromdir = "/home/user/svnwc"
    svnpropertyname = "propertyname"
    propertyname = "propget" />
```

C.85. SvnProplistTask

The SvnProplistTask lists all properties on files, dirs, or revisions from the working copy.

Table C.111: 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.proplist	No
recursive	Boolean	Recursive proplist usage?	false	No

Name	Type	Description	Default	Required
configOption	String	Override subversion's config option	n/a	No

C.85.1. Example

```
<svnproplist
  svnpath = "/usr/bin/svn"
  username = "anony"
  password = "anony"
  repositoryurl = "http://svn.phing.info/tags/2.4.2"
  todir = "/home/user/svnwc"
  recursive = "true"
  propertyname = "proplist"/>
```

C.86. SvnPropsetTask

The `SvnSwitchTask` sets a property on files, dirs, or revisions from the working copy.

Table C.112: 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
svnproperty-name	String	The svn property to set	none	Yes
configOption	String	Override subversion's config option	n/a	No

C.86.1. Example

```
<svnpropset
  svnpath = "/usr/bin/svn"
  username = "anony"
  password = "anony"
  repositoryurl = "http://svn.phing.info/tags/2.4.2"
  todir = "/home/user/svnwc"
  svnpropset = "propertyname"/>
```

C.87. SvnRevertTask

The `SvnRevertTask` reverts a svn repository.

Table C.113: Attributes

Name	Type	Description	Default	Required
workingcopy	String	Working copy directory	none	One of the two
repositoryurl	String	URL of remote repository	none	
recursive	Boolean	Flag for recursive revert.	none	Yes
configOption	String	Override subversion's config option	n/a	No

C.88. SvnSwitchTask

The SvnSwitchTask changes a local directory from one repository to another.

Table C.114: 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
depth	String	Limit operation by depth	empty, files, immediates or infinity	No
ignoreexternals	Boolean	Ignore externals definitions	false	No
trustServerCertificate	Boolean	Trust self-signed certificates	false	No
configOption	String	Override subversion's config option	n/a	No

C.88.1. 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.89. SvnUpdateTask

The SvnUpdateTask updates a local directory.

Table C.115: 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
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
trustServerCertificate	Boolean	Trust self-signed certificates	false	No
configOption	String	Override subversion's config option	n/a	No

C.89.1. 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.90. SymfonyConsoleTask

Executes Symfony2 console commands

Table C.116: 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	bin/console	No
debug	Boolean	The symfony cli debug mode	true	No
silent	Boolean	Disable task output except errors. Use in conjunction with additional output helper like Symfonys ProgressBar	false	No

Name	Type	Description	Default	Required
propertyName	String	The name of the property to store the application output in	n/a	No
checkReturn	Boolean	Whether to check the return code.	false	No

C.90.1. 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.90.2. Supported Nested Tags

- arg

Table C.117: 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
quotes	String	set to true if the value should be enclosed in double quotes	false	No

C.91. TarTask

The TarTask creates a tarball from a fileset or directory.

Table C.118: 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, lzma, 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	warn	No

Name	Type	Description	Default	Required
		- paths are truncated to the maximum length, <code>fail</code> - paths greater than the maximum cause a build exception <code>warn</code> - paths greater than the maximum cause a warning and GNU is used, <code>gnu</code> - GNU extensions are used for any paths greater than the maximum, <code>omit</code> - paths greater than the maximum are omitted from the archive		
<code>prefix</code>	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.91.1. 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.91.2. Supported Nested Tags

- `fileset`

C.92. UntarTask

The `UntarTask` unpacks one or more tar archives.

Table C.119: Attributes

Name	Type	Description	Default	Required
<code>file</code>	String	Archive filename	n/a	No

Name	Type	Description	Default	Required
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
preservePermissions	Boolean	When set to true, preserve permissions (mode, uid, gid) as set in the tar file..	false	No

C.92.1. Example

```
<untar file = "testtar.tar.gz" todir = "dest">
  <fileset dir = ".">
    <include name = "*.tar.gz"/>
    <include name = "*.tar"/>
  </fileset>
</untar>
```

C.92.2. Supported Nested Tags

- fileset

C.93. UnzipTask

The UnzipTask unpacks one or more ZIP archives.

Table C.120: 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.93.1. Example

```
<unzip file = "testzip.zip" todir = "dest">
  <fileset dir = ".">
    <include name = "*.zip"/>
  </fileset>
</unzip>
```

C.93.2. Supported Nested Tags

- fileset

C.94. VisualizerTask

The `VisualizerTask` generates a graphical representation of your current buildfile. This allows you to see all available targets but also the calls and dependencies among targets.

`VisualizerTask` is able to represent:

- Target's depends
- `RunTargetTask`
- `PhingCallTask`
- `ForeachTask`

Table C.121: Basic attributes

Name	Type	Description	Default	Required
<code>format</code>	String	Diagram's format.png Supported formats are: png, svg, puml and eps.		no
<code>destination</code>	String	Location where the diagram will be saved. It could be a file or directory path.	Same location as current buildfile	no
<code>footer</code>	String	A text to display at the bottom right of the diagram.		no
<code>showTitle</code>	Bool	Should the buildfile's title be displayed in diagram?	true	no
<code>showDescription</code>	Bool	Should the buildfile's description be displayed in diagram?	false	no
<code>direction</code>	String	Use this to change the diagram's layout. Valid values are: horizontal and vertical.	vertical	no
<code>server</code>	String	PlantUML server. Needed by all PlantUML mats except puml.	http://www.plantuml.com/plantuml	no

If you have network connectivity issues, you should try `puml` format. This format doesn't require a PlantUML server (and therefore an internet connection) to generate a diagram.

C.94.1. Examples

Using `VisualizerTask` with default values:

```
<visualizer/>
```

Setting diagram's format to `svg` with horizontal arrows:

```
<visualizer format = "svg" direction = "horizontal"/>
```

Save diagram into `resources/images/` directory:

```
<visualizer destination = "resources/images/" />
```

Display buildfile's description and custom footer text:

```
<visualizer showDescription = "true" footer = "© Copyright 2021" />
```

C.94.2. Limitations

- Special target naming is not interpreted by `VisualizerTask`, targets' names are used as *is*. Please read [Target Overriding](#) for more details.
- As said before, `VisualizerTask` depends on a remote PlantUML server. Even if only buildfile's name and targets' names are sent to server, please be sure you are not sending any sensible information.
- PlantUML limits image width and height to 4096 pixels. Overcoming this limitation will require to configure your own PlantUML server and to configure it according to PlantUML FAQ [<http://plantuml.com/faq>] instructions.

C.94.3. Requirements

To work properly, `VisualizerTask` needs to have the following installed:

- SimpleXML extension [<http://php.net/manual/en/book.simplexml.php>]
- XSL extension [<http://php.net/manual/en/book.xsl.php>]
- Guzzle [<http://docs.guzzlephp.org/en/stable/>]
- `jawira/plantuml-client` [<https://packagist.org/packages/jawira/plantuml-client>]

C.94.4. Advanced HTTP configuration

As said before `VisualizerTask` needs a remote server to generate the diagrams. In order to configure the connection with remote server, several attributes and nested tags are available.

Because `VisualizerTask` relies on an internal Phing's library, these attributes and nested tags are shared among these tasks: `HttpGetTask`, `HttpRequestTask` and `VisualizerTask`.

HTTP attributes

Use the following attributes if your PlantUML server requires an authentication mechanism.

Table C.122: Attributes

Name	Type	Description	Default	Required
<code>authUser</code>	<code>String</code>	The authenticationn/a user name		No
<code>authPassword</code>	<code>String</code>	The authenticationn/a password		No

Name	Type	Description	Default	Required
authScheme	String	The authentication scheme	basic	No

Supported Nested Tags

- `config`

Holds additional config data. See Guzzle documentation [<http://docs.guzzlephp.org/en/stable/request-options.html>] for supported values.

Table C.123: 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.124: Attributes

Name	Type	Description	Default	Required
name	String	Header name	n/a	Yes
value	String	Header value	n/a	Yes

Global configuration

In addition to configuring a particular instance of `Guzzle` via nested `<config>` tags it is also possible to set default configuration values for `HttpGetTask` / `HttpRequestTask` / `VisualizerTask` by setting `phing.http.*` properties.

```
<property name="phing.http.proxy" value="socks5://localhost:1080"/>

<!-- This request will go through the default proxy -->
<visualizer/>

<visualizer>
  <!-- This proxy will be used instead of the default one -->
  <config name="proxy" value="http://foo:bar@proxy.example.org:3128"/>
  <header name="user-agent" value="Phing VisualizerTask"/>
</visualizer>
```

C.95. 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.125: 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.
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.95.1. Example

```
<wikipublish
  apiUrl = "http://localhost/wiki/api.php"
  apiUser = "testUser"
  apiPassword = "testPassword"
  title = "Some Page"
  content = "Some content"
  mode = "prepend"/>
```

C.96. 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.126: 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.96.1. 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>
</xmlint>
```

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" />

<xmlint schema = "${docbook.relaxng}" useRNG = "yes">
  <fileset refid = "sources" />
</xmlint>
```

Validate a set of DocBook files against the DocBook RNG grammar

C.96.2. Supported Nested Tags

- fileset

C.97. XmlPropertyTask

Loads property values from a well-formed xml file. There are no other restrictions than "well-formed".

Table C.127: 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.97.1. 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.98. ZSDTPackTask

The `zsdtPackTask` Create a package with the help of the ZendServer Deployment Tool. The pack options should contain pointers to the application data directory, the package descriptor file, and the package scripts directory.

Table C.128: Attributes

Name	Type	Description	Default	Required
package	String	A directory containing the data and the scriptnone directories, in addition to the package descriptor file.	none	Yes
scripts	String	The directory which contains the package de- none ployment scripts. The Deployment Tool will search this directory for the expected files (as described in section 2.2.1) and then packs them.	none	Yes
descriptor	String	The package descriptor file.	none	Yes
source	String	The directory that contains the application re- none sources (PHP sources, JavaScript, etc.). The directory's internal structure must match the necessary structure for the application to be functional.	none	No

Name	Type	Description	Default	Required
output	String	The directory in which the package is created. The package name will be created as <code>app-name-app-version.zpk</code> .	none	No
lint	Boolean	Performs a PHP lint test on the deployment scripts before creating the package.	false	No
phpbin	String	The PHP executable to use for lint.	none	No (Yes if option lint is set to true)
schema	String	The path to the package descriptor schema used for validation.	none	No

C.98.1. Example

```
<zsdtpack lint = "true"
        schema = "file/to/schema.xml"
        descriptor = "file/to/descriptor.xml"
        scripts = "path/to/scripts/"
        package = "path/to/package/"
        source = "path/to/source/"
        output = "path/to/output/"
        phpbin = "path/to/php" />
```

C.99. ZSDTValidateTask

The `zsdtValidateTask` validates a given Zend package descriptor against the schema file.

Table C.129: Attributes

Name	Type	Description	Default	Required
descriptor	String	The package descriptor file.	none	Yes
schema	String	The path to the package descriptor schema used for validation.	none	No

C.99.1. Example

```
<zsdvalidate schema = "/path/to/schema.xml" descriptor = "/path/to/descriptor.xml" />
```

C.100. ZendCodeAnalyzerTask

The `ZendCodeAnalyzerTask` analyze PHP source files using the Zend Code Analyzer tool that ships with all versions of Zend Studio.

Table C.130: Attributes

Name	Type	Description	Default	Required
analyzerPath	String	Path to Zend Code Analyzer binary	n/a	Yes

Name	Type	Description	Default	Required
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.100.1. 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.

C.100.2. Supported Nested Tags

- fileset

C.101. ZipTask

The ZipTask creates a .zip archive from a fileset or directory.

Table C.131: 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
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
comment	String	Comment to add to the zip archive	none	No
ignorelinks	Boolean	Whether to ignore symlinks or not.	false	No

Important note: using basedir and fileset simultaneously can result in strange contents in the archive.

C.101.1. 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.101.2. Supported Nested Tags

- fileset

Appendix D. Core Types

This appendix contains a reference of the system data types contained in Phing.

D.1. Description

Allows for a description of the project to be specified that will be included in the output of the `phing #projecthelp` command.

D.1.1. Usage Examples

```
<description>
This buildfile is used to build the Foo subproject within
the large, complex Bar project.
</description>
```

D.2. Excludes

Specifies a set of files, classes or methods to be excluded from processing.

This element has no attributes, only nested tags

D.2.1. Nested tags

- `file`
- `class`
- `method`

Table D.1: Common attributes for all *file*, *class*, *method* tags

Name	Type	Description	Default	Required
<code>name</code>	String	The name of the class, method or file. This may also be specified as a pattern.		Yes

D.2.2. 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.3. 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 `<filelist>` tag

Name	Type	Description	Default	Required
dir	String	The directory, to which the paths given in <code>files</code> or <code>listfile</code> are relative.	n/a	Yes
files	String	Comma or space-separated list of files.	n/a	Yes (or <code>listfile</code>)
listfile	String	A text file with one filename per line.	n/a	Yes (or <code>files</code>)

D.3.1. 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.4. FileSet

FileSets offer an easy and straightforward way to include files. The tag supports Selectors and PatternSets. 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 <code>include</code> or <code>exclude</code> are relative to.	n/a	Yes
defaultexcludes	Boolean	Whether default exclusions should be used or not. Default excludes are: <code>*~, #*#, .#*, %*%, CVS, CVS/**, .cvsignore, SCCS, SCCS/**, vssver.scc, .svn, .svn/**, ._*, .DS_Store, .darcs, .darcs/**, .git, .git/**, .gitattributes, .gitignore, .gitmodules</code>	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

Name	Type	Description	Default	Required
erroronmissingdir	Boolean	Specify what happens if the base directory does not exist. If true a build error will happen, if false, the fileset will be ignored/empty.	false	No
includes	String	Comma- or space-separated list of patterns of files that must be included; all files are included when omitted.		No
includesfile	String	The name of a file; each line of this file is taken to be an include pattern.		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.		No
excludesfile	String	The name of a file; each line of this file is taken to be an exclude pattern.		No

D.4.1. 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.4.2. 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`.

```
<fileset id = "files" dir = "${phing.dir}/etc">
  <excludesfile name = "test" />
</fileset>
<target name = "test">
  <echo msg = "${toString:files}" />
</target>
```

This will exclude all files from a file named `test`. Each line of this file is taken to be an exclude pattern.

D.4.3. Nested tags

The tags that are supported by `Fileset` are:

- include
- exclude
- patternset
- any of the selectors

The `<include>` and the `<exclude>` tags must have a `name` attribute that contains the pattern to include/exclude.

D.5. DirSet

A DirSet is a group of directories. These directories can be found in a directory tree starting in a base directory and are matched by patterns taken from a number of PatternSets and Selectors.

PatternSets can be specified as nested `<patternset>` elements. In addition, DirSet holds an implicit PatternSet and supports the nested `<include>`, `<includesfile>`, `<exclude>` and `<excludesfile>` elements of `<patternset>` directly, as well as `<patternset>`'s attributes.

Selectors are available as nested elements within the DirSet. If any of the selectors within the DirSet do not select the directory, it is not considered part of the DirSet. This makes a DirSet equivalent to an `<and>` selector container.

Table D.4: Attributes for the `<dirset>` tag

Name	Type	Description	Default	Required
<code>dir</code>	String	The root of the directory tree of this DirSet.	n/a	Yes
<code>casesensitive</code>	Boolean	Specifies whether case-sensitivity should be applied (<code>true yes on</code> or <code>false no off</code>).	<code>true</code>	No
<code>expandsymboliclinks</code>	Boolean	Whether to expand/dereference (follow) symbolic links - set to 'true' to emulate old Phing behavior.	<code>false</code>	No
<code>includes</code>	String	A comma- or space-separated list of patterns of directories that must be included; all directories are included when omitted.	n/a	No
<code>includesfile</code>	String	The name of a file; each line of this file is taken to be an include pattern. Note: if the file is empty and there are no other patterns defined for the fileset, all directories will be included.	n/a	No
<code>excludes</code>	String	A comma- or space-separated list of patterns of directories that must be excluded; no directories are excluded when omitted.	n/a	No
<code>excludesfile</code>	String	The name of a file; each line of this file is taken to be an exclude pattern.	n/a	No

D.5.1. 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.5.2. Usage Examples

```
<dirset dir = "/etc" >
  <include name = "httpd/**" />
  <include name = "php.ini" />
</dirset>

<dirset dir = "/etc" >
  <patternset>
    <include name = "**/*.php" />
    <exclude name = "**/*Test*" />
  </patternset>
</dirset>
```

This will include the apache configuration and PHP configuration file from `/etc`.

D.5.3. Nested tags

The tags that are supported by Fileset are:

- `include`
- `exclude`
- `patternset`
- any of the selectors

The `<include>` and the `<exclude>` tags must have a `name` attribute that contains the pattern to include/exclude.

D.6. PatternSet

The PatternSet data type defines patterns that can be grouped into sets and nested into FileSets. Patterns can be specified by nested `<include>` or `<exclude>` elements.

Table D.5: Attributes for `<patternset>` tag

Name	Type	Description	Default	Required
<code>includes</code>	String	Comma- or space-separated list of patterns of/n/a files that must be included; all files are included when omitted.		No
<code>includesfile</code>	String	The name of a file; each line of this file is taken/n/a to be an include pattern.		No
<code>excludes</code>	String	comma- or space-separated list of patterns of/n/a files that must be excluded; no files (except default excludes) are excluded when omitted.		No
<code>excludesfile</code>	String	The name of a file; each line of this file is taken/n/a to be an exclude pattern.		No

D.6.1. Usage Example

```
<patternset id = "no.tests">
  <include name = "**/*.php"/>
  <exclude name = "**/*Test*"/>
</patternset>
```

D.6.2. 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.7. Path / Classpath

The Path data type can be used to represent 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"/>
</path>

<target name = "blah">
  <taskdef name = "mytask" path = "MyApp\CustomTask\MyTask">
    <classpath refid = "project.class.path"/>
  </taskdef>
</target>
```

Table D.6: 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.		No

D.7.1. Nested tags

The `<paentry>` tag supports nested `<fileset>` and `<dirset>` tags.

D.8. Regexp

Regexp represents a regular expression.

Table D.7: Attributes for `<regexp>` tag

Name	Type	Description	Default	Required
pattern	String	regular expression pattern	n/a	Yes
refid	String	Makes this regexp a reference to a regexp defined elsewhere. If specified no other attributes or nested elements are allowed.	n/a	No

D.8.1. Examples

```
<regexp id = "myregexp" pattern = "alpha(+)beta" />
```

Defines a regular expression for later use with id "myregexp".

```
<regexp refid = "myregexp" />
```

Use the regular expression with id "myregexp".

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 <code>include_path</code> .	n/a	No
classpathlink:href	String	Reference to classpath to use when including classes. This is added to PHP's <code>include_path</code> .	n/a	No

E.1.1. Nested tags

The `PhingFilterReader` supports nested `<classpaentry>`.

E.1.2. Advanced

In order to support the `<filterreader ... />` syntax, your class must extend the `BaseParamFilterReader` class. Most of the filters that are bundled with Phing can be invoked using this syntax. The notable 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 `<replace-regexp ... />` 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 re-ported. One of error, warning, info, verbose, debug.	info	No

E.3. ConcatFilter

This filter prepends or appends the content file to the filtered files.

```
<filterchain>
  <concatfilter prepend = "license.txt" />
</filterchain>
```

Table E.3: Attributes for the `<concatfilter>` tag

Name	Type	Description	Default	Required
prepend	String	The name of the file which content should be/a prepended to the file.		No

Name	Type	Description	Default	Required
append	String	The name of the file which content should be appended to the file.		No

E.4. 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.4: Attributes for the `<headfilter>` tag

Name	Type	Description	Default	Required
lines	Integer	Number of lines to read.	10	No
skip	Integer	Number of lines to skip (from the beginning).	0	No

E.5. IconvFilter

The `IconvFilter` encodes file from `in` encoding to `out` encoding. Usage example:

```
<filterchain>
  <iconvfilter inputencoding = "UTF-8" outputencoding = "CP1251" />
</filterchain>
```

Table E.5: 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.6. 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>
```

Table E.6: Attributes for the `<linecontains>` filter

Name	Type	Description	Default	Required
negate	Boolean	Whether to select non-matching lines only.	false	No
matchAny	Boolean	If false, then all the strings are expected to be present in the line. If true, then the presence of any of the strings in the line is considered a successful match.	false	No

E.6.1. Nested tags

The `linecontains` tag must contain one or more `contains` tags.

E.7. LineContainsRegexp

This filter is similar to Section E.6, “Line Contains ” but you can specify regular expressions instead of simple strings.

```
<filterchain>
  <linecontainsregexp>
    <regexp pattern = "foo(.*?)bar" />
  </linecontainsregexp>
</filterchain>
```

Table E.7: Attributes for the `<linecontainsregexp>` filter

Name	Type	Description	Default	Required
casesensitive	Boolean	Perform a case sensitive match.	true	No
negate	Boolean	Whether to select non-matching lines only.	false	No
regexp	String	Regular expression to be searched for.	n/a	No - Unless specified, a valid nested regexp element has to be set.

E.7.1. Nested tags

The `LineContains` filter has to contain at least one `regexp` tag if the `regexp` attribute has no pattern set. This must have a `pattern` attribute that is set to a regular expression.

E.8. 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.8: Attributes for the `<prefixlines>` tag

Name	Type	Description	Default	Required
prefix	String	String to prepend to every line.	n/a	Yes

E.9. 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.9: 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.9.1. Nested tags

The `ReplaceTokens` filter must contain one or more `token` tags. These must have a `key` and a `value` attribute.

E.10. 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 = "example1dir/" postfix = ".php" />
  </filterchain>
```

The filter 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.10: Attributes for the `<replacetokenswithfile>` tag

Name	Type	Description	Default	Required
<code>begintoken</code>	String	The string that marks the beginning of a token. <code>##@#</code>		No
<code>endtoken</code>	String	The string that marks the end of a token. <code>##@#</code>		No
<code>prefix</code>	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
<code>postfix</code>	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
<code>dir</code>	String	The directory where to look for the files to use as replacements for the tokens		No
<code>translatehtml</code>	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.10.1. Nested tags

None.

E.11. 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.11.1. 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.11: Attributes for the `<regex>` tag

Name	Type	Description	Default	Required
pattern	String	Regular expression used as needle. Phingn/a relies on Perl-compatible [http://php.net/pcre] regular expressions.		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 mul-false ti-line mode.		No
modifiers	String	Raw regular expression modifiers [http://ph-" p.net/manual/en/ reference.pcre.pattern.modifiers.php]. You can pass several modifiers as single string, and use raw modifiers with <code>ignoreCase</code> and <code>multiline</code> attributes. In case of conflict, value specified by dedicated attribute takes precedence.		No

The previous example (using `modifiers` attribute this time):

```
<filterchain>
  <replaceregexp>
    <regex pattern = "ANT" replace = "Phing" modifiers = "i"/>
    <regex pattern = "(\w+)\.java" replace = "\1.php"/>
  </replaceregexp>
</filterchain>
```

E.12. SortFilter

The sort filter reads all lines and sorts them. The sort order can be reversed.

```
<filterchain>
  <sortfilter reverse = "true" />
</filterchain>
```

Table E.12: Attributes for the `<sortfilter>` filter

Name	Type	Description	Default	Required
reverse	Boolean	whether to reverse the sort order, defaults tofalse false.		No

E.13. StripLineBreaks

The `StripLineBreaks` filter removes all linebreaks from the stream passed through the filter chain.

```
<filterchain>
  <striplinebreaks />
</filterchain>
```

E.14. 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.14.1. 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.15. StripPhpComments

The `StripPhpComments` filter removes all PHP comments from the stream passed through the filter.

```
<filterchain>
  <stripphpcomments />
</filterchain>
```

E.16. 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.17. TabToSpaces

The `TabToSpaces` filter replaces all tab characters with a given count of space characters.

```
<filterchain>
  <tabtospaces tablength = "8" />
</filterchain>
```

Table E.13: Attributes for the `<tabtospaces>` filter

Name	Type	Description	Default	Required
tablength	Integer	The number of space characters that a tab is8 to represent.		No

E.18. TailFilter

Similar to Section E.4, “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.14: Attributes for the `<tailfilter>` tag

Name	Type	Description	Default	Required
lines	Integer	Number of lines from the back to read.	10	No
skip	Integer	Number of lines to be skipped (from the end).0		No

E.19. 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.15: Attributes for the `<tidyfilter>` tag

Name	Type	Description	Default	Required
encoding	String	The expected input encoding of the file.	utf8	No

E.19.1. 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 official Quick Reference [<http://tidy.sourceforge.net/docs/quickref.html>].

E.20. 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.16: Attributes for the `<xincludefilter>` tag

Name	Type	Description	Default	Required
basedir	String	The working directory from which to processProject the Xincludes. Relative pathnames in the in-basedir clude tags are based on this location.		No
resolveexter- nals	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.21. XsltFilter

The `XsltFilter` applies a XSL template to the stream. Though you can use this filter directly, you should use `XsltTask` 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.17: Attributes for the `<xsltfilter>` tag

Name	Type	Description	Default	Required
style	String	The XSLT stylesheet to use for transform-a-tion.	n/a	Yes
html	Boolean	Whether to parse the input as HTML (usinglibxml2 DOMDocument::loadHTML()).	false	No
resolvedocu- mentexternals	Boolean	Whether to resolve entitiesin the XML document. (see this link [http://www.php.net/manual/en/class.domdocument.php#domdocument-props.resolveexternals] for details)	false	No
re- solvestyleshee- textexternals	Boolean	Whether to resolve entities in the stylesheet.	false	No

E.21.1. 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.

E.22. ClassConstants

This filters basic constants defined in a PHP Class, and outputs them in lines composed of the format name=value.

```
<property file = "constants.php">
  <filterchain>
    <classconstants />
  </filterchain>
</property>
```

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. ChainedMapper

This mapper implementation can contain multiple nested mappers. File mapping is performed by passing the source filename to the first nested mapper, its results to the second, and so on. The target filenames generated by the last nested mapper comprise the ultimate results of the mapping operation. The `to` and `from` attributes are ignored.

F.2.1. Examples

```
<mapper type = "chained">
  <mapper type = "flatten"/>
  <mapper type = "glob" from = "*.php" to = "new/path/*.php"/>
  <mapper>
    <mapper type = "glob" from = "*.php" to = "*.php1"/>
    <mapper type = "glob" from = "*.php" to = "*.php2"/>
  </mapper>
</mapper>
```

Applying the mapper, you will get the following results from the following filenames:

Table F.2: Result of mapping

From	To
foo/bar/a.php	new/path/a.php1 and new/path/a.php2
foo/bar/b.php	new/path/b.php1 and new/path/b.php2

F.3. CompositeMapper

This mapper implementation can contain multiple nested mappers. File mapping is performed by passing the source filename to each nested `<mapper>` in turn, returning all results. The `to` and `from` attributes are ignored.

```
<copy todir = "testbuild">
  <fileset dir = "${project.basedir}" />
</copy>
```

This code will copy all files in the fileset to `/tmp`. All files will be in the target directory.

F.3.1. Examples

```
<mapper type = "composite">
  <mapper type = "glob" from = "*.xsl" to = "*.from.xsl" />
  <mapper type = "glob" from = "*.xml" to = "*.from.xml" />
  <mapper type = "glob" from = "*.php" to = "*.from.php" />
</mapper>
```

Applying the mapper, you will get the following results from the following filenames:

Table F.3: Result of mapping

From	To
test.php	./tmp/test.from.php
test.xml	./tmp/test.from.xml
test.xsl	./tmp/test.from.xsl

F.4. FirstMatchMapper

This mapper supports an arbitrary number of nested mappers and returns the results of the first mapper that matches. This is different from composite mapper which collects the results of all matching children.

F.4.1. Examples

```
<mapper type = "firstmatch">
  <mapper type = "glob" from = "*.txt" to = "*.bak" />
  <mapper type = "glob" from = "*.php" to = "*.php" />
</mapper>
```

Applying the mapper, you will get the following results from the following filenames:

Table F.4: Result of mapping

From	To
foo/bar/A.txt	foo/bar/A.bak
foo/bar/A.php	foo/bar/A.php

F.5. CutDirsMapper

The `CutDirsMapper` strips a configured number of leading directories from the source file name.

F.5.1. Examples

```
<mapper type = "cutdirs" to = "1"/>
```

The mapper as above will do the following mappings:

Table F.5: Result of mapping

From	To
foo/bar/A.txt	bar/A.txt

F.6. 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.6.1. Examples

```
<mapper type = "flatten" />
```

Applying the mapper, you will get the following results from the following filenames:

Table F.6: Result of mapping

From	To
test.txt	test.txt
./foo/bar/test.bak	test.bak

F.7. 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.

Table F.7: The globmapper mapper can take the following extra attributes.

Name	Type	Description	Default	Required
handledirsep	String	If this is specified, the mapper will ignore the difference between the normal directory separator characters - \ and /. This attribute is useful for cross-platform build files.	false	No
casesensitive	Boolean	If this is false, the mapper will ignore case when matching the glob pattern.	true	No

F.7.1. Examples

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

Applying the mapper, you will get the following results from the following filenames:

Table F.8: 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.8. IdentityMapper

The IdentityMapper will not change anything on the source filenames.

F.9. MergeMapper

The MergeMapper changes all source filenames to the same filename.

F.9.1. Examples

```
<mapper type = "merge" to = "test.tar"/>
```

Applying the mapper, you will get the following results from the following filenames:

Table F.9: Result of mapping

From	To
test.txt	test.tar
./foo/bar/test.txt	test.tar

From	To
mytxt	test.tar
SomeClass.php	test.tar

F.10. 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.

Table F.10: The regexp mapper can take the following extra attributes.

Name	Type	Description	Default	Required
handledirsep	String	If this is specified, the mapper will ignore the difference between the normal directory separator characters - \ and /. This attribute is useful for cross-platform build files.	false	No
casesensitive	Boolean	If this is false, the mapper will ignore case when matching the glob pattern.	true	No

F.10.1. Examples

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

The mapper as above will do the following mappings:

Table F.11: 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 TypesFileSet` (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
- `<Readable>` - Select files if they are readable
- `<Writable>` - Select files if they are writable
- `<Executable>` - Select files if they are executable
- `<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
- `<Different>` - Select files that are different from those elsewhere
- `<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
- `<Symlink>` - Select files if they are symlink.
- `<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
casesensitive	Whether to pay attention to case when looking for the string in the text attribute.	true	No
ignorewhitespace	Whether to eliminate whitespace before checking for the string in the text attribute.	false	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 shouldn't be in a format parsable by PHP's <code>strtotime()</code> [http://www.php.net/strtotime] function.		One of the two
seconds	The number of seconds since Midnight Jan 1 1970 (Unix epoch) that should be tested for.	n/a	
millis	The number of milliseconds since Midnight Jan 1 1970 (Unix epoch) that should be tested for. Note: It will be internally converted to seconds.		
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: <ul style="list-style-type: none"> <code>before</code> - select files whose last modified date is before the indicated date <code>after</code> - select files whose last modified date is after the indicated date <code>equal</code> - select files whose last modified date is this exact date 		No
granularity	The number of seconds leeway to use when comparing file modification times.	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 <code>mapper</code> element, if any.		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.		No

G.4. Depth

The `<depth>` 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 `<depth>` 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.		One of the two
max	The maximum number of directory levels below the base directory that a file can be and still be selected.		

G.5. Different

The `<different>` selector will select a file if it is deemed to be 'different' from an equivalent file in another location. The rules for determining difference between the two files are as follows:

- If a file is only present in the resource collection you apply the selector to but not in targetdir (or after applying the mapper) the file is selected.
- If a file is only present in targetdir (or after applying the mapper) it is ignored.
- Files with different lengths are different.
- If ignoreFileTimes is turned off, then differing file timestamps will cause files to be regarded as different.
- Unless ignoreContents is set to true, a byte-for-byte check is run against the two files.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not just a timestamp. For example: anything fetched from a web site, or the output of some program. To reduce the amount of checking, when using this task inside a `<copy>` task, set `preserveLastModified` to true to propagate the timestamp from the source file to the destination file.

The `<different>` selector supports the use of a contained `<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.

```
<fileset dir = "${phing.1.5}/classes" includes = "**/*.php">
  <different targetdir = "${phing.1.4.1}/classes"
    ignoreFileTimes = "true"/>
</fileset>
```

Table G.5: Attributes for the `<different>` 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 <code>mapper</code> element, if any.		Yes
ignoreFileTimes	Whether to use file times in the comparison or not.	true	No
ignoreContents	Whether to do a byte per byte compare.	false	No

G.6. 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.6: Attributes for the `<filename>` selector

Name	Description	Default	Required
name	The name of files to select. The name parameter can contain the standard Phing wildcard characters.		Exactly one of the two
regex	The regular expression matching files to select.	n/a	

Name	Description	Default	Required
casesensitive	Whether to pay attention to case when looking at file names.	true	No
negate	Whether to reverse the effects of this filename selection, therefore emulating an exclude rather than include tag.	false	No

G.7. Present

The `<present>` tag selects files that have an equivalent file in another directory tree.

The `<present>` tag supports the use of a contained 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.7: 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 <code><mapper></code> 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: <code>srconly</code> - select files only if they are in the src directory tree but not in the target directory tree <code>both</code> - select files only if they are present both in the src and target directory trees	both	No

G.8. 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.8: Attributes for the `<containsregexp>` selector

Name	Description	Default	Required
expression	Specifies the regular expression that must match true in every file.	n/a	Yes

Name	Description	Default	Required
casesensitive	Perform a case sensitive match.	true	No
multiline	Perform a multi line match.	false	No

G.9. Size

The `<size>` tag selects files matching a specified size limit.

```
<fileset dir = "${src}">
  <size value = "4M" when = "more"/>
</fileset>
```

Table G.9: Attributes for the `<size>` selector

Name	Description	Default	Required
value	The size of the file which should be tested for.n/a Examples: 250M, 10G, 1T.		Yes
when	Indicates how to interpret the size, whether theequal files to be selected should be larger, smaller, or equal to that value. Accepted values are: <ul style="list-style-type: none"> less - select files less than the indicated size more - select files greater than the indicated size equal - select files this exact size 		No



Note

File size can be written using IEC and SI suffixes, bytes are assumed when suffix is not specified. The following suffixes (case-insensitive) are supported:

Table G.10: Supported file size suffixes

Standard	Suffixes	Equivalence
	B.	1 byte
	K, Ki, KiB, kibi, kibibyte.	1024 bytes
IEC	M, Mi, MiB, mebi, mebibyte.	1024 kibibytes
	G, Gi, GiB, gibi, gibibyte.	1024 mebibytes
	T, Ti, TiB, tebi, tebibyte.	1024 gibibytes
	kB, kilo, kilobyte.	1000 bytes
	MB, mega, megabyte.	1000 kilobytes
SI	GB, giga, gigabyte.	1000 megabytes
	TB, tera, terabyte.	1000 gigabytes

G.10. Type

The `<type>` tag selects files of a certain type: directory or regular.

```
<fileset dir = "${src}">
  <type type = "dir"/>
</fileset>
```

Table G.11: Attributes for the `<type>` selector

Name	Description	Default	Required
type	The type of file which should be tested for. Ei-n/a ther file or dir.		Yes

G.11. 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.12. 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.12: Attributes for the `<majority>` selector container

Name	Description	Default	Required
allowtie	Whether files should be selected if there are antrue even number of selectors selecting them as are not selecting them.		No

G.13. Modified

The `<modified>` selector computes a value for a file, compares that to the value stored in a cache and select the file, if these two values differ.

Because this selector is highly configurable the order in which the selection is done is:

1. get the absolute path for the file
2. get the cached value from the configured cache (absolute path as key)
3. get the new value from the configured algorithm
4. compare these two values with the configured comparator
5. update the cache if needed and requested
6. do the selection according to the comparison result

The comparison, computing of the hashvalue and the store is done by implementation of special interfaces. Therefore they may provide additional parameters.

Table G.13: Attributes for the *<modified>* selector

Name	Description	Default	Required
algorithm	The type of algorithm should be used. Acceptable values are (further information see later): <ul style="list-style-type: none"> • hashfile • lastmodified 		No
cache	The type of cache should be used. Acceptable values are (further information see later): <ul style="list-style-type: none"> • propertyfile 	propertyfile	No
comparator	The type of comparator should be used. Acceptable values are: <ul style="list-style-type: none"> • equal 	equal	No
algorithmclass	Classname of custom algorithm implementation. Lower priority than algorithm.	n/a	No
cacheclass	Classname of custom cache implementation. Lower priority than cache.	n/a	No
comparatorclass	Classname of custom comparator implementation. Lower priority than comparator.	n/a	No
update	Should the cache be updated when values differ? (boolean)	true	No
seldirs	Should directories be selected? (boolean)	true	No
delayupdate	If set to "true", the storage of the cache will be delayed until the next finished BuildEvent; task finished, target finished or build finished, whichever comes first. This is provided for increased performance. If set to "false", the storage of the cache will happen with each change. This attribute depends upon the update attribute. (boolean)	true	No

G.13.1. Parameters specified as nested elements

All attributes of a *<modified>* selector can be set with nested *<param/>* tags. Additional values can be set with *<param/>* tags according to the rules below.

Table G.14: *algorithm*

Name	Description	Default	Required
hashfile	This Algorithm supports the following attribute: n/a <ul style="list-style-type: none"> algorithm.algorithm (optional): Name of the hashfile algorithm (e.g. "MD5" or "SHA"); default is "MD5" 	n/a	No
lastmodified	Uses the lastModified property of a file. No additional configuration is required.	n/a	No

G.13.2. Examples

Here are some examples of how to use the Modified Selector:

```
<copy todir = "dest">
  <fileset dir = "src">
    <modified/>
  </fileset>
</copy>
```

This will copy all files from src to dest which content has changed. Using an updating PropertyfileCache with cache.properties and MD5-FilehashAlgorithm.

```
<copy todir = "dest">
  <fileset dir = "src">
    <modified update = "true"
      seldirs = "true"
      cache = "propertyfile"
      algorithm = "digest"
      comparator = "equal">
      <param name = "cache.cachefile" value = "cache.properties"/>
      <param name = "algorithm.algorithm" value = "md5"/>
    </modified>
  </fileset>
</copy>
```

G.14. 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.15. 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.16. 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.17. Readable

The `<readable>` selector selects only files that are readable.

```
<fileset dir = "${src}" includes = "**/*.php">
  <readable>
</fileset>
```

G.18. Writable

The `<writable>` selector selects only files that are writable.

```
<fileset dir = "${src}" includes = "**/*.php">
  <writable>
</fileset>
```

G.19. Executable

The `<executable>` selector selects only files that are executable.

```
<fileset dir = "${src}" includes = "**/*.php">
  <executable>
</fileset>
```

G.20. 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.

G.21. Symlink Selector

The `<symlink>` selector selects only files that are symbolic links.

G.22. PosixPermissions Selector

The `<posixpermissions>` selector selects only files that have the given POSIX permissions.

Table G.15: *Attributes for the `<posixpermissions>` selector*

Name	Description	Default	Required
permissions	POSIX permissions in string (rwxrwxrwx) or true octal (777) format	true	Yes

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 tasks and 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>
```

Phing allows declaring tasks outside targets. Note that these tasks are evaluated before any targets are executed.

H.1.3. 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 explicitly specified when calling Phing		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
strict	Boolean	Enables the strict-mode for the project build process. If enabled, a warning would be considered as an error, and the build will be aborted.	false	No

H.2. Targets and Extension-Points

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
<code>depends</code>	String	One or more names of targets that have to be executed before this target can be executed.	n/a	No
<code>description</code>	String	A free text description of the target.	n/a	No
<code>if</code>	String	The name of the property that is to be set if then/a target is to be executed.		No
<code>name</code>	String	The name of the target	n/a	Yes
<code>unless</code>	String	The name of the property that is to be set if then/a target is not to be executed.		No
<code>hidden</code>	Boolean	Whether or not to include this target in the list of targets generated by <code>phing -l</code>	False	No
<code>logskipped</code>	Boolean	Whether to log message as INFO instead of VERBOSE if target is skipped	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.

H.2.3. Extension-Points

Extension-Points are similar to targets in that they have a name and a depends list and can be executed from the command line. Just like targets they represent a state during the build process.

Unlike targets they don't contain any tasks, their main purpose is to collect targets that contribute to the desired state in their depends list.

Targets can add themselves to an extension-point's depends list via their `extensionOf` attribute. The targets that add themselves will be added after the targets of the explicit depends attribute of the extension-point, if multiple targets add themselves, their relative order is not defined.

The main purpose of an extension-point is to act as an extension point for build files designed to be imported. In the imported file, an extension-point defines a state that must be reached and targets from other build files can join the depends list of said extension-point in order to contribute to that state.

For example your imported build file may need to compile code, it might look like:

```
<target name = "create-directory-layout">
  ...
</target>

<extension-point name = "ready-to-compile"
  depends = "create-directory-layout"/>

<target name = "compile" depends = "ready-to-compile">
  ...
</target>
```

```
Call-Graph: create-directory-layout -> 'empty slot' -> compile
```

And you need to generate some source before compilation, then in your main build file you may use something like

```
<target name = "generate-sources"
  extensionOf = "ready-to-compile">
  ...
</target>
```

```
Call-Graph: create-directory-layout -> generate-sources -> compile
```

This will ensure that the "generate-sources" target is executed before the "compile" target.

Don't rely on the order of the depends list, if "generate-sources" depends on "create-directory-layout" then it must explicitly depend on it via its own depends attribute.

Appendix I. Loggers and Listeners

Phing has two related features to allow the build process to be monitored: listeners and loggers.

I.1. Listeners

A listener is alerted of the following events.

- build started
- build finished
- target started
- target finished
- task started
- task finished
- message logged

These are used internally for various recording and housekeeping operations, however new listeners may be registered on the command line through the `-listener` argument.

I.2. Loggers

Loggers extend the capabilities of listeners and add the following features:

- Receives a handle to the standard output and error print streams and therefore can log information to the console or the `-logfile` specified file.
- Logging level (`-quiet`, `-verbose`, `-debug`) aware
- Emacs-mode aware

I.3. DefaultLogger

Simply run Phing normally, or: `phing -logger "Phing\Listener\DefaultLogger"`

I.4. AnsiColorLogger

The `AnsiColorLogger` adds color to the standard Phing output by prefixing and suffixing ANSI color code escape sequences to it. It is just an extension of `DefaultLogger` and hence provides all features that `DefaultLogger` does.

`AnsiColorLogger` differentiates the output by assigning different colors depending upon the type of the message.

If used with the `-logfile` option, the output file will contain all the necessary escape codes to display the text in colorized mode when displayed in the console using applications like `cat`, `more`, etc.

This is designed to work on terminals that support ANSI color codes.

If the user wishes to override the default colors with custom ones, a file containing zero or more of the custom color key-value pairs must be created. The recognized keys and their default values are shown below:

```
AnsiColorLogger.ERROR_COLOR=01;31
AnsiColorLogger.WARNING_COLOR=01;35
AnsiColorLogger.INFO_COLOR=00;36
AnsiColorLogger.VERBOSE_COLOR=00;32
AnsiColorLogger.DEBUG_COLOR=01;34
```

Each key takes as value a color combination defined as "Attribute;Foreground;Background". In the above example, background value has not been used.

This file must be specified as the value of a system variable named `phing.logger.defaults` and passed as an argument using the `-D` option to the `php` command that invokes the Phing application. An easy way to achieve this is to add `-Dphing.logger.defaults=/path/to/your/file`

```
phing -logger "Phing\Listener\AnsiColorLogger"
```

I.5. MailLogger

The MailLogger captures all output logged through DefaultLogger (standard Phing output) and will send success and failure messages to unique e-mail lists, with control for turning off success or failure messages individually.

Table I.1: Properties controlling the operation of MailLogger:

Property	Description	Required
<code>phing.log.mail.from</code>	Mail "from" address	Yes, if mail needs to be sent
<code>phing.log.mail.replyto</code>	Mail "replyto" address(es), comma-separated	No
<code>phing.log.mail.properties.file</code>	Filename of properties file that will override other values.	No
<code>phing.log.mail.success.cc</code>	Address to send success messages to carbon copy (cc)	No
<code>phing.log.mail.failure.cc</code>	Address to send failure messages to carbon copy (cc)	No
<code>phing.log.mail.success.bcc</code>	Address to send success messages to blind carbon copy (bcc)	No
<code>phing.log.mail.failure.bcc</code>	Address to send failure messages to blind carbon copy (bcc)	No

Property	Description	Required
phing.log.mail.success.body	fixed text of mail body for a successful build, default is to send the logfile	No
phing.log.mail.failure.body	fixed text of mail body for a failed build, default is to send the logfile	No
phing.log.mail.success.subject	Subject of successful build	No - default to Build Success
phing.log.mail.failure.subject	Subject of failed build	No - default to Build Failure
phing.log.mail.success.to	Address to send success messages to	required if success mail to be sent
phing.log.mail.failure.to	Address to send failure messages to	required if failure mail to be sent
phing.log.mail.success.notify	Send build success e-mails?	No - default to true
phing.log.mail.failure.notify	Send build failure e-mails?	No - default to true

```
phing -logger "Phing\Listener\MailLogger"
```

I.6. NoBannerLogger

Removes output of empty target output. `phing -logger "Phing\Listener\NoBannerLogger"`

I.7. ProfileLogger

This logger stores the time needed for executing a task, target and the whole build and prints these information. The output contains a timestamp when entering the build, target or task and a timestamp and the needed time when exiting.

I.8. StatisticsListener

A phing BuildListener which can be used to gather statistics while a phing build is executed. Statistics on the targets and tasks executed are written to the console after the build completes. Some of the statistics captured are: - the number of times a target / task is called - the average processing time

spent on a target / task - the total processing time spent on a target / task - the total processing time spent on a target / task expressed as a percentage

I.9. TimestampedLogger

Acts like the default logger, except that the final success/failure message also includes the time that the build completed.

I.10. SilentLogger

A logger which logs nothing but build failure and what task might output.

I.11. MonologListener

Listener which sends events to Monolog.

To use the MonologListener, start Phing with this command: `phing -listener "Phing\\Listener\\MonologListener"`

Configuration is not (yet) implemented, but you could easily extend the existing MonologListener with your own implementation. Or better yet, send us a PR :-)

I.12. DisguiseLogger

A logger which masks passwords in URI strings - i.e: `[echo] hide password from URI http://foo:*****@example.com/baz?one=two`

To use the DisguiseLogger, start Phing with this command: `phing -logger "Phing\\Listener\\DisguiseLogger"`

Appendix J. File Formats

J.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.buildidir and so on

J.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}

```

Property files may also be formatted in YAML format:

```

# Property files contain key/value pairs
key: value

```

```
# Nested values will be available as concatenated strings after import. E.g.,
# you may access these values with keys in the form of "myapp.window.hsize".
myapp:
  window:
    hsize: 300
    vsize: 200
    xpos: 10
    ypos: 100

# You can refer to values of other properties by enclosing their
# keys in "${}".
text:
  width: "${myapp.window.hsize}"
```

Property files may also be formatted in XML format:

```
<myapp>
  <window>
    <hsize>300</hsize>
    <vsize>200</hsize>
    <xpos>10</hsize>
    <ypos>100</hsize>
  </window>
</myapp>

myapp.window.hsize=300
myapp.window.vsize=200
myapp.window.xpos=10
myapp.window.ypos=100
```

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

[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> .

[php] *The PHP homepage - PHP Hypertext Preprocessor.* <http://www.php.net> .

[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/> .
