4.10.1

* Docs  
  [User Manual](http://docs.google.com/userguide/userguide.html)  
  [Guides and Tutorials](https://guides.gradle.org)  
  [DSL Reference](http://docs.google.com/dsl/)  
  [Javadoc](http://docs.google.com/javadoc/)  
  [Release Notes](http://docs.google.com/release-notes.html)
* [Forums](https://discuss.gradle.org/)
* [Training](https://gradle.org/training/)
* [Try Gradle Enterprise](https://gradle.com/enterprise)
* [User Manual Home](http://docs.google.com/userguide/userguide.html)
* [DSL Reference Home](http://docs.google.com/dsl/)
* [Release Notes](http://docs.google.com/release-notes.html)
  + [Properties](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N1FC60)
  + [Methods](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N1FD49)

### Build script blocks

* [allprojects { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:allprojects(groovy.lang.Closure))
* [artifacts { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:artifacts(groovy.lang.Closure))
* [buildscript { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:buildscript(groovy.lang.Closure))
* [configurations { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:configurations(groovy.lang.Closure))
* [dependencies { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:dependencies(groovy.lang.Closure))
* [repositories { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:repositories(groovy.lang.Closure))
* [sourceSets { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:sourceSets(groovy.lang.Closure))
* [subprojects { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:subprojects(groovy.lang.Closure))
* [publishing { }](http://docs.google.com/org.gradle.api.Project.html#org.gradle.api.Project:publishing(groovy.lang.Closure))

### Core types

* [Project](http://docs.google.com/org.gradle.api.Project.html)
* [Task](http://docs.google.com/org.gradle.api.Task.html)
* [Gradle](http://docs.google.com/org.gradle.api.invocation.Gradle.html)
* [Settings](http://docs.google.com/org.gradle.api.initialization.Settings.html)
* [IncludedBuild](http://docs.google.com/org.gradle.api.initialization.IncludedBuild.html)
* [Script](http://docs.google.com/org.gradle.api.Script.html)
* [JavaToolChain](http://docs.google.com/org.gradle.jvm.toolchain.JavaToolChain.html)
* [SourceSet](http://docs.google.com/org.gradle.api.tasks.SourceSet.html)
* [SourceSetOutput](http://docs.google.com/org.gradle.api.tasks.SourceSetOutput.html)
* [SourceDirectorySet](http://docs.google.com/org.gradle.api.file.SourceDirectorySet.html)
* [IncrementalTaskInputs](http://docs.google.com/org.gradle.api.tasks.incremental.IncrementalTaskInputs.html)
* [Configuration](http://docs.google.com/org.gradle.api.artifacts.Configuration.html)
* [ResolutionStrategy](http://docs.google.com/org.gradle.api.artifacts.ResolutionStrategy.html)
* [ArtifactResolutionQuery](http://docs.google.com/org.gradle.api.artifacts.query.ArtifactResolutionQuery.html)
* [ComponentSelection](http://docs.google.com/org.gradle.api.artifacts.ComponentSelection.html)
* [ComponentSelectionRules](http://docs.google.com/org.gradle.api.artifacts.ComponentSelectionRules.html)
* [ExtensionAware](http://docs.google.com/org.gradle.api.plugins.ExtensionAware.html)
* [ExtraPropertiesExtension](http://docs.google.com/org.gradle.api.plugins.ExtraPropertiesExtension.html)
* [PluginDependenciesSpec](http://docs.google.com/org.gradle.plugin.use.PluginDependenciesSpec.html)
* [PluginDependencySpec](http://docs.google.com/org.gradle.plugin.use.PluginDependencySpec.html)
* [PluginManagementSpec](http://docs.google.com/org.gradle.plugin.management.PluginManagementSpec.html)
* [ResourceHandler](http://docs.google.com/org.gradle.api.resources.ResourceHandler.html)
* [TextResourceFactory](http://docs.google.com/org.gradle.api.resources.TextResourceFactory.html)

### Publishing types

* [PublishingExtension](http://docs.google.com/org.gradle.api.publish.PublishingExtension.html)
* [IvyPublication](http://docs.google.com/org.gradle.api.publish.ivy.IvyPublication.html)
* [IvyArtifact](http://docs.google.com/org.gradle.api.publish.ivy.IvyArtifact.html)
* [IvyArtifactSet](http://docs.google.com/org.gradle.api.publish.ivy.IvyArtifactSet.html)
* [IvyModuleDescriptorSpec](http://docs.google.com/org.gradle.api.publish.ivy.IvyModuleDescriptorSpec.html)
* [IvyModuleDescriptorAuthor](http://docs.google.com/org.gradle.api.publish.ivy.IvyModuleDescriptorAuthor.html)
* [IvyModuleDescriptorLicense](http://docs.google.com/org.gradle.api.publish.ivy.IvyModuleDescriptorLicense.html)
* [IvyModuleDescriptorDescription](http://docs.google.com/org.gradle.api.publish.ivy.IvyModuleDescriptorDescription.html)
* [MavenPublication](http://docs.google.com/org.gradle.api.publish.maven.MavenPublication.html)
* [MavenArtifact](http://docs.google.com/org.gradle.api.publish.maven.MavenArtifact.html)
* [MavenArtifactSet](http://docs.google.com/org.gradle.api.publish.maven.MavenArtifactSet.html)
* [MavenPom](http://docs.google.com/org.gradle.api.publish.maven.MavenPom.html)
* [MavenPomCiManagement](http://docs.google.com/org.gradle.api.publish.maven.MavenPomCiManagement.html)
* [MavenPomContributor](http://docs.google.com/org.gradle.api.publish.maven.MavenPomContributor.html)
* [MavenPomContributorSpec](http://docs.google.com/org.gradle.api.publish.maven.MavenPomContributorSpec.html)
* [MavenPomDeveloper](http://docs.google.com/org.gradle.api.publish.maven.MavenPomDeveloper.html)
* [MavenPomDeveloperSpec](http://docs.google.com/org.gradle.api.publish.maven.MavenPomDeveloperSpec.html)
* [MavenPomDistributionManagement](http://docs.google.com/org.gradle.api.publish.maven.MavenPomDistributionManagement.html)
* [MavenPomIssueManagement](http://docs.google.com/org.gradle.api.publish.maven.MavenPomIssueManagement.html)
* [MavenPomLicense](http://docs.google.com/org.gradle.api.publish.maven.MavenPomLicense.html)
* [MavenPomLicenseSpec](http://docs.google.com/org.gradle.api.publish.maven.MavenPomLicenseSpec.html)
* [MavenPomMailingList](http://docs.google.com/org.gradle.api.publish.maven.MavenPomMailingList.html)
* [MavenPomMailingListSpec](http://docs.google.com/org.gradle.api.publish.maven.MavenPomMailingListSpec.html)
* [MavenPomOrganization](http://docs.google.com/org.gradle.api.publish.maven.MavenPomOrganization.html)
* [MavenPomRelocation](http://docs.google.com/org.gradle.api.publish.maven.MavenPomRelocation.html)
* [MavenPomScm](http://docs.google.com/org.gradle.api.publish.maven.MavenPomScm.html)

### Container types

* [TaskContainer](http://docs.google.com/org.gradle.api.tasks.TaskContainer.html)
* [ConfigurationContainer](http://docs.google.com/org.gradle.api.artifacts.ConfigurationContainer.html)
* [RepositoryHandler](http://docs.google.com/org.gradle.api.artifacts.dsl.RepositoryHandler.html)
* [DependencyHandler](http://docs.google.com/org.gradle.api.artifacts.dsl.DependencyHandler.html)
* [ArtifactHandler](http://docs.google.com/org.gradle.api.artifacts.dsl.ArtifactHandler.html)

### Build Cache types

* [BuildCacheConfiguration](http://docs.google.com/org.gradle.caching.configuration.BuildCacheConfiguration.html)
* [DirectoryBuildCache](http://docs.google.com/org.gradle.caching.local.DirectoryBuildCache.html)
* [HttpBuildCache](http://docs.google.com/org.gradle.caching.http.HttpBuildCache.html)

### Input Normalization types

* [InputNormalizationHandler](http://docs.google.com/org.gradle.normalization.InputNormalizationHandler.html)
* [InputNormalization](http://docs.google.com/org.gradle.normalization.InputNormalization.html)
* [RuntimeClasspathNormalization](http://docs.google.com/org.gradle.normalization.RuntimeClasspathNormalization.html)

### Help Task types

* [TaskReportTask](http://docs.google.com/org.gradle.api.tasks.diagnostics.TaskReportTask.html)
* [ProjectReportTask](http://docs.google.com/org.gradle.api.tasks.diagnostics.ProjectReportTask.html)
* [DependencyReportTask](http://docs.google.com/org.gradle.api.tasks.diagnostics.DependencyReportTask.html)
* [DependencyInsightReportTask](http://docs.google.com/org.gradle.api.tasks.diagnostics.DependencyInsightReportTask.html)
* [PropertyReportTask](http://docs.google.com/org.gradle.api.tasks.diagnostics.PropertyReportTask.html)
* [ComponentReport](http://docs.google.com/org.gradle.api.reporting.components.ComponentReport.html)
* [DependentComponentsReport](http://docs.google.com/org.gradle.api.reporting.dependents.DependentComponentsReport.html)
* [ModelReport](http://docs.google.com/org.gradle.api.reporting.model.ModelReport.html)

### Task types

* [AntlrTask](http://docs.google.com/org.gradle.api.plugins.antlr.AntlrTask.html)
* [BuildEnvironmentReportTask](http://docs.google.com/org.gradle.api.tasks.diagnostics.BuildEnvironmentReportTask.html)
* [Checkstyle](http://docs.google.com/org.gradle.api.plugins.quality.Checkstyle.html)
* [CodeNarc](http://docs.google.com/org.gradle.api.plugins.quality.CodeNarc.html)
* [CompareGradleBuilds](http://docs.google.com/org.gradle.api.plugins.buildcomparison.gradle.CompareGradleBuilds.html)
* [Copy](http://docs.google.com/org.gradle.api.tasks.Copy.html)
* [CreateStartScripts](http://docs.google.com/org.gradle.jvm.application.tasks.CreateStartScripts.html)
* [Delete](http://docs.google.com/org.gradle.api.tasks.Delete.html)
* [Ear](http://docs.google.com/org.gradle.plugins.ear.Ear.html)
* [Exec](http://docs.google.com/org.gradle.api.tasks.Exec.html)
* [FindBugs](http://docs.google.com/org.gradle.api.plugins.quality.FindBugs.html)
* [GenerateIvyDescriptor](http://docs.google.com/org.gradle.api.publish.ivy.tasks.GenerateIvyDescriptor.html)
* [GenerateMavenPom](http://docs.google.com/org.gradle.api.publish.maven.tasks.GenerateMavenPom.html)
* [GenerateBuildDashboard](http://docs.google.com/org.gradle.api.reporting.GenerateBuildDashboard.html)
* [GradleBuild](http://docs.google.com/org.gradle.api.tasks.GradleBuild.html)
* [GroovyCompile](http://docs.google.com/org.gradle.api.tasks.compile.GroovyCompile.html)
* [Groovydoc](http://docs.google.com/org.gradle.api.tasks.javadoc.Groovydoc.html)
* [HtmlDependencyReportTask](http://docs.google.com/org.gradle.api.reporting.dependencies.HtmlDependencyReportTask.html)
* [JacocoReport](http://docs.google.com/org.gradle.testing.jacoco.tasks.JacocoReport.html)
* [JacocoMerge](http://docs.google.com/org.gradle.testing.jacoco.tasks.JacocoMerge.html)
* [JacocoCoverageVerification](http://docs.google.com/org.gradle.testing.jacoco.tasks.JacocoCoverageVerification.html)
* [Jar](http://docs.google.com/org.gradle.api.tasks.bundling.Jar.html)
* [JavaCompile](http://docs.google.com/org.gradle.api.tasks.compile.JavaCompile.html)
* [Javadoc](http://docs.google.com/org.gradle.api.tasks.javadoc.Javadoc.html)
* [JavaExec](http://docs.google.com/org.gradle.api.tasks.JavaExec.html)
* [JDepend](http://docs.google.com/org.gradle.api.plugins.quality.JDepend.html)
* [Pmd](http://docs.google.com/org.gradle.api.plugins.quality.Pmd.html)
* [PublishToIvyRepository](http://docs.google.com/org.gradle.api.publish.ivy.tasks.PublishToIvyRepository.html)
* [PublishToMavenRepository](http://docs.google.com/org.gradle.api.publish.maven.tasks.PublishToMavenRepository.html)
* [ScalaCompile](http://docs.google.com/org.gradle.api.tasks.scala.ScalaCompile.html)
* [ScalaDoc](http://docs.google.com/org.gradle.api.tasks.scala.ScalaDoc.html)
* [InitBuild](http://docs.google.com/org.gradle.buildinit.tasks.InitBuild.html)
* [Sign](http://docs.google.com/org.gradle.plugins.signing.Sign.html)
* [Sync](http://docs.google.com/org.gradle.api.tasks.Sync.html)
* [Tar](http://docs.google.com/org.gradle.api.tasks.bundling.Tar.html)
* [AbstractTestTask](http://docs.google.com/org.gradle.api.tasks.testing.AbstractTestTask.html)
* [Test](http://docs.google.com/org.gradle.api.tasks.testing.Test.html)
* [TestReport](http://docs.google.com/org.gradle.api.tasks.testing.TestReport.html)
* [Upload](http://docs.google.com/org.gradle.api.tasks.Upload.html)
* [War](http://docs.google.com/org.gradle.api.tasks.bundling.War.html)
* [Wrapper](http://docs.google.com/org.gradle.api.tasks.wrapper.Wrapper.html)
* [WriteProperties](http://docs.google.com/org.gradle.api.tasks.WriteProperties.html)
* [Zip](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html)

### Reporting types

* [CustomizableHtmlReport](http://docs.google.com/org.gradle.api.reporting.CustomizableHtmlReport.html)
* [SingleFileReport](http://docs.google.com/org.gradle.api.reporting.SingleFileReport.html)
* [DirectoryReport](http://docs.google.com/org.gradle.api.reporting.DirectoryReport.html)
* [FindBugsXmlReport](http://docs.google.com/org.gradle.api.plugins.quality.FindBugsXmlReport.html)
* [Report](http://docs.google.com/org.gradle.api.reporting.Report.html)
* [Reporting](http://docs.google.com/org.gradle.api.reporting.Reporting.html)
* [ReportContainer](http://docs.google.com/org.gradle.api.reporting.ReportContainer.html)
* [ReportingExtension](http://docs.google.com/org.gradle.api.reporting.ReportingExtension.html)

### Eclipse/IDEA model types

* [EclipseModel](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseModel.html)
* [EclipseProject](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseProject.html)
* [EclipseClasspath](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseClasspath.html)
* [EclipseJdt](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseJdt.html)
* [EclipseWtp](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseWtp.html)
* [EclipseWtpComponent](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseWtpComponent.html)
* [EclipseWtpFacet](http://docs.google.com/org.gradle.plugins.ide.eclipse.model.EclipseWtpFacet.html)
* [IdeaModel](http://docs.google.com/org.gradle.plugins.ide.idea.model.IdeaModel.html)
* [IdeaProject](http://docs.google.com/org.gradle.plugins.ide.idea.model.IdeaProject.html)
* [IdeaModule](http://docs.google.com/org.gradle.plugins.ide.idea.model.IdeaModule.html)
* [IdeaWorkspace](http://docs.google.com/org.gradle.plugins.ide.idea.model.IdeaWorkspace.html)
* [XmlFileContentMerger](http://docs.google.com/org.gradle.plugins.ide.api.XmlFileContentMerger.html)
* [FileContentMerger](http://docs.google.com/org.gradle.plugins.ide.api.FileContentMerger.html)

### Eclipse/IDEA task types

* [GenerateEclipseProject](http://docs.google.com/org.gradle.plugins.ide.eclipse.GenerateEclipseProject.html)
* [GenerateEclipseClasspath](http://docs.google.com/org.gradle.plugins.ide.eclipse.GenerateEclipseClasspath.html)
* [GenerateEclipseJdt](http://docs.google.com/org.gradle.plugins.ide.eclipse.GenerateEclipseJdt.html)
* [GenerateEclipseWtpComponent](http://docs.google.com/org.gradle.plugins.ide.eclipse.GenerateEclipseWtpComponent.html)
* [GenerateEclipseWtpFacet](http://docs.google.com/org.gradle.plugins.ide.eclipse.GenerateEclipseWtpFacet.html)
* [GenerateIdeaModule](http://docs.google.com/org.gradle.plugins.ide.idea.GenerateIdeaModule.html)
* [GenerateIdeaProject](http://docs.google.com/org.gradle.plugins.ide.idea.GenerateIdeaProject.html)
* [GenerateIdeaWorkspace](http://docs.google.com/org.gradle.plugins.ide.idea.GenerateIdeaWorkspace.html)

### Native software types

* [PrebuiltLibrary](http://docs.google.com/org.gradle.nativeplatform.PrebuiltLibrary.html)
* [PrebuiltSharedLibraryBinary](http://docs.google.com/org.gradle.nativeplatform.PrebuiltSharedLibraryBinary.html)
* [PrebuiltStaticLibraryBinary](http://docs.google.com/org.gradle.nativeplatform.PrebuiltStaticLibraryBinary.html)
* [NativeComponentSpec](http://docs.google.com/org.gradle.nativeplatform.NativeComponentSpec.html)
* [NativeExecutableSpec](http://docs.google.com/org.gradle.nativeplatform.NativeExecutableSpec.html)
* [NativeLibrarySpec](http://docs.google.com/org.gradle.nativeplatform.NativeLibrarySpec.html)
* [NativeTestSuiteSpec](http://docs.google.com/org.gradle.nativeplatform.test.NativeTestSuiteSpec.html)
* [CUnitTestSuiteSpec](http://docs.google.com/org.gradle.nativeplatform.test.cunit.CUnitTestSuiteSpec.html)
* [GoogleTestTestSuiteSpec](http://docs.google.com/org.gradle.nativeplatform.test.googletest.GoogleTestTestSuiteSpec.html)
* [NativeBinarySpec](http://docs.google.com/org.gradle.nativeplatform.NativeBinarySpec.html)
* [NativeExecutableBinarySpec](http://docs.google.com/org.gradle.nativeplatform.NativeExecutableBinarySpec.html)
* [NativeLibraryBinarySpec](http://docs.google.com/org.gradle.nativeplatform.NativeLibraryBinarySpec.html)
* [SharedLibraryBinarySpec](http://docs.google.com/org.gradle.nativeplatform.SharedLibraryBinarySpec.html)
* [StaticLibraryBinarySpec](http://docs.google.com/org.gradle.nativeplatform.StaticLibraryBinarySpec.html)
* [NativeTestSuiteBinarySpec](http://docs.google.com/org.gradle.nativeplatform.test.NativeTestSuiteBinarySpec.html)
* [CUnitTestSuiteBinarySpec](http://docs.google.com/org.gradle.nativeplatform.test.cunit.CUnitTestSuiteBinarySpec.html)
* [GoogleTestTestSuiteBinarySpec](http://docs.google.com/org.gradle.nativeplatform.test.googletest.GoogleTestTestSuiteBinarySpec.html)
* [NativePlatform](http://docs.google.com/org.gradle.nativeplatform.platform.NativePlatform.html)
* [BuildType](http://docs.google.com/org.gradle.nativeplatform.BuildType.html)
* [Flavor](http://docs.google.com/org.gradle.nativeplatform.Flavor.html)
* [Gcc](http://docs.google.com/org.gradle.nativeplatform.toolchain.Gcc.html)
* [Clang](http://docs.google.com/org.gradle.nativeplatform.toolchain.Clang.html)
* [VisualCpp](http://docs.google.com/org.gradle.nativeplatform.toolchain.VisualCpp.html)
* [AssemblerSourceSet](http://docs.google.com/org.gradle.language.assembler.AssemblerSourceSet.html)
* [CSourceSet](http://docs.google.com/org.gradle.language.c.CSourceSet.html)
* [CppSourceSet](http://docs.google.com/org.gradle.language.cpp.CppSourceSet.html)
* [ObjectiveCSourceSet](http://docs.google.com/org.gradle.language.objectivec.ObjectiveCSourceSet.html)
* [ObjectiveCppSourceSet](http://docs.google.com/org.gradle.language.objectivecpp.ObjectiveCppSourceSet.html)
* [WindowsResourceSet](http://docs.google.com/org.gradle.language.rc.WindowsResourceSet.html)
* [VisualStudioProject](http://docs.google.com/org.gradle.ide.visualstudio.VisualStudioProject.html)
* [VisualStudioSolution](http://docs.google.com/org.gradle.ide.visualstudio.VisualStudioSolution.html)
* [NativeExecutable](http://docs.google.com/org.gradle.nativeplatform.NativeExecutable.html)
* [NativeLibrary](http://docs.google.com/org.gradle.nativeplatform.NativeLibrary.html)
* [NativeBinary](http://docs.google.com/org.gradle.nativeplatform.NativeBinary.html)
* [NativeExecutableBinary](http://docs.google.com/org.gradle.nativeplatform.NativeExecutableBinary.html)
* [SharedLibraryBinary](http://docs.google.com/org.gradle.nativeplatform.SharedLibraryBinary.html)
* [StaticLibraryBinary](http://docs.google.com/org.gradle.nativeplatform.StaticLibraryBinary.html)

### Native component task types

* [CppCompile](http://docs.google.com/org.gradle.language.cpp.tasks.CppCompile.html)
* [CCompile](http://docs.google.com/org.gradle.language.c.tasks.CCompile.html)
* [Assemble](http://docs.google.com/org.gradle.language.assembler.tasks.Assemble.html)
* [ObjectiveCCompile](http://docs.google.com/org.gradle.language.objectivec.tasks.ObjectiveCCompile.html)
* [ObjectiveCppCompile](http://docs.google.com/org.gradle.language.objectivecpp.tasks.ObjectiveCppCompile.html)
* [WindowsResourceCompile](http://docs.google.com/org.gradle.language.rc.tasks.WindowsResourceCompile.html)
* [LinkExecutable](http://docs.google.com/org.gradle.nativeplatform.tasks.LinkExecutable.html)
* [LinkSharedLibrary](http://docs.google.com/org.gradle.nativeplatform.tasks.LinkSharedLibrary.html)
* [CreateStaticLibrary](http://docs.google.com/org.gradle.nativeplatform.tasks.CreateStaticLibrary.html)
* [InstallExecutable](http://docs.google.com/org.gradle.nativeplatform.tasks.InstallExecutable.html)
* [RunTestExecutable](http://docs.google.com/org.gradle.nativeplatform.test.tasks.RunTestExecutable.html)

Zip

**Table of Contents**

[Properties](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N1FC60)[Methods](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N1FD49)[Script blocks](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N1FF04)[Property details](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N1FF09)[Method details](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#N20150)

| API Documentation: | [Zip](http://docs.google.com/javadoc/org/gradle/api/tasks/bundling/Zip.html) |
| --- | --- |

| Known Subtypes: | | [Jar](http://docs.google.com/dsl/org.gradle.jvm.tasks.Jar.html) | | --- | |
| --- | --- | --- |

Assembles a ZIP archive. The default is to compress the contents of the zip.

Properties

| Property | Description |
| --- | --- |
| [appendix](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:appendix) | The appendix part of the archive name, if any. |
| [archiveName](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:archiveName) | The archive name. If the name has not been explicitly set, the pattern for the name is: [baseName]-[appendix]-[version]-[classifier].[extension] |
| [archivePath](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:archivePath) | The path where the archive is constructed. The path is simply the destinationDir plus the archiveName. |
| [baseName](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:baseName) | The base name of the archive. |
| [caseSensitive](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:caseSensitive) | Specifies whether case-sensitive pattern matching should be used. |
| [classifier](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:classifier) | The classifier part of the archive name, if any. |
| [destinationDir](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:destinationDir) | The directory where the archive is generated into. |
| [dirMode](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:dirMode) | The Unix permissions to use for the target directories. null means that existing permissions are preserved. It is dependent on the copy action implementation whether these permissions will actually be applied. |
| [duplicatesStrategy](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:duplicatesStrategy) | The strategy to use when trying to copy more than one file to the same destination. |
| [entryCompression](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:entryCompression) | The compression level of the entries of the archive. If set to [ZipEntryCompression.DEFLATED](http://docs.google.com/javadoc/org/gradle/api/tasks/bundling/ZipEntryCompression.html#DEFLATED) (the default), each entry is compressed using the DEFLATE algorithm. If set to [ZipEntryCompression.STORED](http://docs.google.com/javadoc/org/gradle/api/tasks/bundling/ZipEntryCompression.html#STORED) the entries of the archive are left uncompressed. |
| [excludes](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:excludes) | The set of exclude patterns. |
| [extension](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:extension) | The extension part of the archive name. |
| [fileMode](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:fileMode) | The Unix permissions to use for the target files. null means that existing permissions are preserved. It is dependent on the copy action implementation whether these permissions will actually be applied. |
| [includeEmptyDirs](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:includeEmptyDirs) | Tells if empty target directories will be included in the copy. |
| [includes](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:includes) | The set of include patterns. |
| [metadataCharset](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:metadataCharset) | Incubating  The character set used to encode ZIP metadata like file names. Defaults to the platform's default character set. |
| [preserveFileTimestamps](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:preserveFileTimestamps) | Incubating  Specifies whether file timestamps should be preserved in the archive. |
| [reproducibleFileOrder](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:reproducibleFileOrder) | Incubating  Specifies whether to enforce a reproducible file order when reading files from directories. |
| [source](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:source) | The source files for this task. |
| [version](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:version) | The version part of the archive name, if any. |
| [zip64](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:zip64) | Incubating  Whether the zip can contain more than 65535 files and/or support files greater than 4GB in size. |

Methods

| Method | Description |
| --- | --- |
| [eachFile](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:eachFile(groovy.lang.Closure))(closure) | Adds an action to be applied to each file as it about to be copied into its destination. The given closure is called with a [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) as its parameter. Actions are executed in the order added, and are inherited from the parent spec. |
| [eachFile](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:eachFile(org.gradle.api.Action))(action) | Adds an action to be applied to each file as it is about to be copied into its destination. The action can change the destination path of the file, filter the contents of the file, or exclude the file from the result entirely. Actions are executed in the order added, and are inherited from the parent spec. |
| [exclude](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:exclude(groovy.lang.Closure))(excludeSpec) | Adds an exclude spec. This method may be called multiple times to append new specs.The given closure is passed a [FileTreeElement](http://docs.google.com/javadoc/org/gradle/api/file/FileTreeElement.html) as its parameter. The closure should return true or false. Example: |
| [exclude](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:exclude(java.lang.Iterable))(excludes) | Adds an ANT style exclude pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed. |
| [exclude](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:exclude(java.lang.String%5B%5D))(excludes) | Adds an ANT style exclude pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed. |
| [exclude](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:exclude(org.gradle.api.specs.Spec))(excludeSpec) | Adds an exclude spec. This method may be called multiple times to append new specs. If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed. |
| [expand](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:expand(java.util.Map))(properties) | Expands property references in each file as it is copied. More specifically, each file is transformed using Groovy's [SimpleTemplateEngine](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/text/SimpleTemplateEngine.html). This means you can use simple property references, such as $property or ${property} in the file. You can also include arbitrary Groovy code in the file, such as ${version ?: 'unknown'} or ${classpath\*.name.join(' ')} |
| [filesMatching](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filesMatching(java.lang.Iterable,%20org.gradle.api.Action))(patterns, action) | Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path matches any of the specified Ant-style patterns. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path. |
| [filesMatching](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filesMatching(java.lang.String,%20org.gradle.api.Action))(pattern, action) | Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path matches the specified Ant-style pattern. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path. |
| [filesNotMatching](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filesNotMatching(java.lang.Iterable,%20org.gradle.api.Action))(patterns, action) | Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path does not match any of the specified Ant-style patterns. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path. |
| [filesNotMatching](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filesNotMatching(java.lang.String,%20org.gradle.api.Action))(pattern, action) | Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path does not match the specified Ant-style pattern. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path. |
| [filter](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filter(groovy.lang.Closure))(closure) | Adds a content filter based on the provided closure. The Closure will be called with each line (stripped of line endings) and should return a String to replace the line or null to remove the line. If every line is removed, the result will be an empty file, not an absent one. |
| [filter](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filter(java.lang.Class))(filterType) | Adds a content filter to be used during the copy. Multiple calls to filter, add additional filters to the filter chain. Each filter should implement java.io.FilterReader. Include org.apache.tools.ant.filters.\* for access to all the standard Ant filters. |
| [filter](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filter(java.util.Map,%20java.lang.Class))(properties, filterType) | Adds a content filter to be used during the copy. Multiple calls to filter, add additional filters to the filter chain. Each filter should implement java.io.FilterReader. Include org.apache.tools.ant.filters.\* for access to all the standard Ant filters. |
| [filter](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:filter(org.gradle.api.Transformer))(transformer) | Adds a content filter based on the provided transformer. The Closure will be called with each line (stripped of line endings) and should return a String to replace the line or null to remove the line. If every line is removed, the result will be an empty file, not an absent one. |
| [from](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:from(java.lang.Object,%20groovy.lang.Closure))(sourcePath, c) | Specifies the source files or directories for a copy and creates a child CopySourceSpec. The given source path is evaluated as per [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D)) . |
| [from](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:from(java.lang.Object,%20org.gradle.api.Action))(sourcePath, configureAction) | Specifies the source files or directories for a copy and creates a child CopySpec. The given source path is evaluated as per [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D)) . |
| [from](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:from(java.lang.Object%5B%5D))(sourcePaths) | Specifies source files or directories for a copy. The given paths are evaluated as per [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D)). |
| [include](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:include(groovy.lang.Closure))(includeSpec) | Adds an include spec. This method may be called multiple times to append new specs. The given closure is passed a [FileTreeElement](http://docs.google.com/javadoc/org/gradle/api/file/FileTreeElement.html) as its parameter. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns or specs to be included. |
| [include](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:include(java.lang.Iterable))(includes) | Adds an ANT style include pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns to be processed. |
| [include](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:include(java.lang.String%5B%5D))(includes) | Adds an ANT style include pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns to be processed. |
| [include](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:include(org.gradle.api.specs.Spec))(includeSpec) | Adds an include spec. This method may be called multiple times to append new specs. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns or specs to be included. |
| [into](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:into(java.lang.Object))(destPath) | Specifies the destination directory \*inside\* the archive for the files. The destination is evaluated as per [Project.file(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:file(java.lang.Object)). Don't mix it up with [AbstractArchiveTask.getDestinationDir()](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html#org.gradle.api.tasks.bundling.AbstractArchiveTask:destinationDir) which specifies the output directory for the archive. |
| [into](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:into(java.lang.Object,%20groovy.lang.Closure))(destPath, configureClosure) | Creates and configures a child CopySpec with a destination directory \*inside\* the archive for the files. The destination is evaluated as per [Project.file(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:file(java.lang.Object)). Don't mix it up with [AbstractArchiveTask.getDestinationDir()](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html#org.gradle.api.tasks.bundling.AbstractArchiveTask:destinationDir) which specifies the output directory for the archive. |
| [into](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:into(java.lang.Object,%20org.gradle.api.Action))(destPath, copySpec) | Creates and configures a child CopySpec with a destination directory \*inside\* the archive for the files. The destination is evaluated as per [Project.file(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:file(java.lang.Object)). Don't mix it up with [AbstractArchiveTask.getDestinationDir()](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html#org.gradle.api.tasks.bundling.AbstractArchiveTask:destinationDir) which specifies the output directory for the archive. |
| [rename](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:rename(groovy.lang.Closure))(closure) | Renames a source file. The closure will be called with a single parameter, the name of the file. The closure should return a String object with a new target name. The closure may return null, in which case the original name will be used. |
| [rename](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:rename(java.lang.String,%20java.lang.String))(sourceRegEx, replaceWith) | Renames files based on a regular expression. Uses java.util.regex type of regular expressions. Note that the replace string should use the '$1' syntax to refer to capture groups in the source regular expression. Files that do not match the source regular expression will be copied with the original name. |
| [rename](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:rename(java.util.regex.Pattern,%20java.lang.String))(sourceRegEx, replaceWith) | Renames files based on a regular expression. See [CopyProcessingSpec.rename(java.lang.String, java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/CopyProcessingSpec.html#rename-java.lang.String,%20java.lang.String-). |
| [rename](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:rename(org.gradle.api.Transformer))(renamer) | Renames a source file. The function will be called with a single parameter, the name of the file. The function should return a new target name. The function may return null, in which case the original name will be used. |
| [with](http://docs.google.com/org.gradle.api.tasks.bundling.Zip.html#org.gradle.api.tasks.bundling.Zip:with(org.gradle.api.file.CopySpec%5B%5D))(sourceSpecs) | Adds the given specs as a child of this spec. |

Script blocks

No script blocks

Property details

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) appendix

The appendix part of the archive name, if any.

Default with java plugin:null

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) archiveName

The archive name. If the name has not been explicitly set, the pattern for the name is: [baseName]-[appendix]-[version]-[classifier].[extension]

Default with java plugin:*${baseName}*-*${appendix}*-*${version}*-*${classifier}*.*${extension}*

[**File**](http://download.oracle.com/javase/8/docs/api/java/io/File.html) archivePath (read-only)

The path where the archive is constructed. The path is simply the destinationDir plus the archiveName.

Default with java plugin:*${destinationDir}*/*${archiveName}*

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) baseName

The base name of the archive.

Default with java plugin:project.archivesBaseName

boolean caseSensitive

Specifies whether case-sensitive pattern matching should be used.

Default with java plugin:true

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) classifier

The classifier part of the archive name, if any.

Default with java plugin:null

[**File**](http://download.oracle.com/javase/8/docs/api/java/io/File.html) destinationDir

The directory where the archive is generated into.

Default with java plugin:project.distsDir

[**Integer**](http://download.oracle.com/javase/8/docs/api/java/lang/Integer.html) dirMode

The Unix permissions to use for the target directories. null means that existing permissions are preserved. It is dependent on the copy action implementation whether these permissions will actually be applied.

Default with java plugin:null

[**DuplicatesStrategy**](http://docs.google.com/javadoc/org/gradle/api/file/DuplicatesStrategy.html) duplicatesStrategy

The strategy to use when trying to copy more than one file to the same destination.

The value can be set with a case insensitive string of the enum value (e.g. 'exclude' for [DuplicatesStrategy.EXCLUDE](http://docs.google.com/javadoc/org/gradle/api/file/DuplicatesStrategy.html#EXCLUDE)).

This strategy can be overridden for individual files by using [CopySpec.eachFile(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html#eachFile-org.gradle.api.Action-) or [CopySpec.filesMatching(java.lang.String, org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html#filesMatching-java.lang.String,%20org.gradle.api.Action-).

Default with java plugin:null

[**ZipEntryCompression**](http://docs.google.com/javadoc/org/gradle/api/tasks/bundling/ZipEntryCompression.html) entryCompression

The compression level of the entries of the archive. If set to [ZipEntryCompression.DEFLATED](http://docs.google.com/javadoc/org/gradle/api/tasks/bundling/ZipEntryCompression.html#DEFLATED) (the default), each entry is compressed using the DEFLATE algorithm. If set to [ZipEntryCompression.STORED](http://docs.google.com/javadoc/org/gradle/api/tasks/bundling/ZipEntryCompression.html#STORED) the entries of the archive are left uncompressed.

Default:ZipEntryCompression.DEFLATED

[**Set**](http://download.oracle.com/javase/8/docs/api/java/util/Set.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> excludes

The set of exclude patterns.

Default with java plugin:[]

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) extension

The extension part of the archive name.

[**Integer**](http://download.oracle.com/javase/8/docs/api/java/lang/Integer.html) fileMode

The Unix permissions to use for the target files. null means that existing permissions are preserved. It is dependent on the copy action implementation whether these permissions will actually be applied.

Default with java plugin:null

boolean includeEmptyDirs

Tells if empty target directories will be included in the copy.

Default with java plugin:true

[**Set**](http://download.oracle.com/javase/8/docs/api/java/util/Set.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> includes

The set of include patterns.

Default with java plugin:[]

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) metadataCharset

Note: This property is [incubating](http://docs.google.com/userguide/feature_lifecycle.html) and may change in a future version of Gradle.

The character set used to encode ZIP metadata like file names. Defaults to the platform's default character set.

Default:Platform default encoding

boolean preserveFileTimestamps

Note: This property is [incubating](http://docs.google.com/userguide/feature_lifecycle.html) and may change in a future version of Gradle.

Specifies whether file timestamps should be preserved in the archive.

If false this ensures that archive entries have the same time for builds between different machines, Java versions and operating systems.

Default with java plugin:true

boolean reproducibleFileOrder

Note: This property is [incubating](http://docs.google.com/userguide/feature_lifecycle.html) and may change in a future version of Gradle.

Specifies whether to enforce a reproducible file order when reading files from directories.

Gradle will then walk the directories on disk which are part of this archive in a reproducible order independent of file systems and operating systems. This helps Gradle reliably produce byte-for-byte reproducible archives.

Default with java plugin:false

[**FileCollection**](http://docs.google.com/javadoc/org/gradle/api/file/FileCollection.html) source (read-only)

The source files for this task.

Default with java plugin:[]

[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) version

The version part of the archive name, if any.

Default with java plugin:project.version

boolean zip64

Note: This property is [incubating](http://docs.google.com/userguide/feature_lifecycle.html) and may change in a future version of Gradle.

Whether the zip can contain more than 65535 files and/or support files greater than 4GB in size.

The standard zip format has hard limits on file size and count. The [Zip64 format extension](http://en.wikipedia.org/wiki/Zip_(file_format)#ZIP64) practically removes these limits and is therefore required for building large zips.

However, not all Zip readers support the Zip64 extensions. Notably, the [ZipInputStream](http://download.oracle.com/javase/8/docs/api/java/util/zip/ZipInputStream.html) JDK class does not support Zip64 for versions earlier than Java 7. This means you should not enable this property if you are building JARs to be used with Java 6 and earlier runtimes.

Default:false

Method details

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) eachFile([**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) closure)

Adds an action to be applied to each file as it about to be copied into its destination. The given closure is called with a [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) as its parameter. Actions are executed in the order added, and are inherited from the parent spec.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) eachFile([**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**FileCopyDetails**](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html)> action)

Adds an action to be applied to each file as it is about to be copied into its destination. The action can change the destination path of the file, filter the contents of the file, or exclude the file from the result entirely. Actions are executed in the order added, and are inherited from the parent spec.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) exclude([**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) excludeSpec)

Adds an exclude spec. This method may be called multiple times to append new specs.The given closure is passed a [FileTreeElement](http://docs.google.com/javadoc/org/gradle/api/file/FileTreeElement.html) as its parameter. The closure should return true or false. Example:

copySpec {  
 from 'source'  
 into 'destination'  
 //an example of excluding files from certain configuration:  
 exclude { it.file in configurations.someConf.files }  
}

If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) exclude([**Iterable**](http://download.oracle.com/javase/8/docs/api/java/lang/Iterable.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> excludes)

Adds an ANT style exclude pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) exclude([**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)... excludes)

Adds an ANT style exclude pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) exclude([**Spec**](http://docs.google.com/javadoc/org/gradle/api/specs/Spec.html)<[**FileTreeElement**](http://docs.google.com/javadoc/org/gradle/api/file/FileTreeElement.html)> excludeSpec)

Adds an exclude spec. This method may be called multiple times to append new specs. If excludes are not provided, then no files will be excluded. If excludes are provided, then files must not match any exclude pattern to be processed.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) expand([**Map**](http://download.oracle.com/javase/8/docs/api/java/util/Map.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html), ?> properties)

Expands property references in each file as it is copied. More specifically, each file is transformed using Groovy's [SimpleTemplateEngine](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/text/SimpleTemplateEngine.html). This means you can use simple property references, such as $property or ${property} in the file. You can also include arbitrary Groovy code in the file, such as ${version ?: 'unknown'} or ${classpath\*.name.join(' ')}

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filesMatching([**Iterable**](http://download.oracle.com/javase/8/docs/api/java/lang/Iterable.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> patterns, [**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**FileCopyDetails**](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html)> action)

Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path matches any of the specified Ant-style patterns. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filesMatching([**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) pattern, [**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**FileCopyDetails**](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html)> action)

Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path matches the specified Ant-style pattern. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filesNotMatching([**Iterable**](http://download.oracle.com/javase/8/docs/api/java/lang/Iterable.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> patterns, [**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**FileCopyDetails**](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html)> action)

Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path does not match any of the specified Ant-style patterns. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filesNotMatching([**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) pattern, [**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**FileCopyDetails**](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html)> action)

Configure the [FileCopyDetails](http://docs.google.com/javadoc/org/gradle/api/file/FileCopyDetails.html) for each file whose path does not match the specified Ant-style pattern. This is equivalent to using eachFile() and selectively applying a configuration based on the file's path.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filter([**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) closure)

Adds a content filter based on the provided closure. The Closure will be called with each line (stripped of line endings) and should return a String to replace the line or null to remove the line. If every line is removed, the result will be an empty file, not an absent one.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filter([**Class**](http://download.oracle.com/javase/8/docs/api/java/lang/Class.html)<? extends [**FilterReader**](http://download.oracle.com/javase/8/docs/api/java/io/FilterReader.html)> filterType)

Adds a content filter to be used during the copy. Multiple calls to filter, add additional filters to the filter chain. Each filter should implement java.io.FilterReader. Include org.apache.tools.ant.filters.\* for access to all the standard Ant filters.

Examples:

filter(StripJavaComments)  
filter(com.mycompany.project.CustomFilter)

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filter([**Map**](http://download.oracle.com/javase/8/docs/api/java/util/Map.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html), ?> properties, [**Class**](http://download.oracle.com/javase/8/docs/api/java/lang/Class.html)<? extends [**FilterReader**](http://download.oracle.com/javase/8/docs/api/java/io/FilterReader.html)> filterType)

Adds a content filter to be used during the copy. Multiple calls to filter, add additional filters to the filter chain. Each filter should implement java.io.FilterReader. Include org.apache.tools.ant.filters.\* for access to all the standard Ant filters.

Filter properties may be specified using groovy map syntax.

Examples:

filter(HeadFilter, lines:25, skip:2)  
filter(ReplaceTokens, tokens:[copyright:'2009', version:'2.3.1'])

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) filter([**Transformer**](http://docs.google.com/javadoc/org/gradle/api/Transformer.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html), [**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> transformer)

Adds a content filter based on the provided transformer. The Closure will be called with each line (stripped of line endings) and should return a String to replace the line or null to remove the line. If every line is removed, the result will be an empty file, not an absent one.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) from([**Object**](http://download.oracle.com/javase/8/docs/api/java/lang/Object.html) sourcePath, [**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) c)

Specifies the source files or directories for a copy and creates a child CopySourceSpec. The given source path is evaluated as per [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D)) .

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) from([**Object**](http://download.oracle.com/javase/8/docs/api/java/lang/Object.html) sourcePath, [**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**CopySpec**](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html)> configureAction)

Specifies the source files or directories for a copy and creates a child CopySpec. The given source path is evaluated as per [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D)) .

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) from([**Object**](http://download.oracle.com/javase/8/docs/api/java/lang/Object.html)... sourcePaths)

Specifies source files or directories for a copy. The given paths are evaluated as per [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D)).

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) include([**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) includeSpec)

Adds an include spec. This method may be called multiple times to append new specs. The given closure is passed a [FileTreeElement](http://docs.google.com/javadoc/org/gradle/api/file/FileTreeElement.html) as its parameter. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns or specs to be included.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) include([**Iterable**](http://download.oracle.com/javase/8/docs/api/java/lang/Iterable.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> includes)

Adds an ANT style include pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns to be processed.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) include([**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)... includes)

Adds an ANT style include pattern. This method may be called multiple times to append new patterns and multiple patterns may be specified in a single call. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns to be processed.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) include([**Spec**](http://docs.google.com/javadoc/org/gradle/api/specs/Spec.html)<[**FileTreeElement**](http://docs.google.com/javadoc/org/gradle/api/file/FileTreeElement.html)> includeSpec)

Adds an include spec. This method may be called multiple times to append new specs. If includes are not provided, then all files in this container will be included. If includes are provided, then a file must match at least one of the include patterns or specs to be included.

[**AbstractArchiveTask**](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html) into([**Object**](http://download.oracle.com/javase/8/docs/api/java/lang/Object.html) destPath)

Specifies the destination directory \*inside\* the archive for the files. The destination is evaluated as per [Project.file(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:file(java.lang.Object)). Don't mix it up with [AbstractArchiveTask.getDestinationDir()](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html#org.gradle.api.tasks.bundling.AbstractArchiveTask:destinationDir) which specifies the output directory for the archive.

[**AbstractArchiveTask**](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html) into([**Object**](http://download.oracle.com/javase/8/docs/api/java/lang/Object.html) destPath, [**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) configureClosure)

Creates and configures a child CopySpec with a destination directory \*inside\* the archive for the files. The destination is evaluated as per [Project.file(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:file(java.lang.Object)). Don't mix it up with [AbstractArchiveTask.getDestinationDir()](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html#org.gradle.api.tasks.bundling.AbstractArchiveTask:destinationDir) which specifies the output directory for the archive.

[**CopySpec**](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html) into([**Object**](http://download.oracle.com/javase/8/docs/api/java/lang/Object.html) destPath, [**Action**](http://docs.google.com/javadoc/org/gradle/api/Action.html)<? super [**CopySpec**](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html)> copySpec)

Creates and configures a child CopySpec with a destination directory \*inside\* the archive for the files. The destination is evaluated as per [Project.file(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:file(java.lang.Object)). Don't mix it up with [AbstractArchiveTask.getDestinationDir()](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.AbstractArchiveTask.html#org.gradle.api.tasks.bundling.AbstractArchiveTask:destinationDir) which specifies the output directory for the archive.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) rename([**Closure**](http://docs.groovy-lang.org/2.4.15/html/gapi/groovy/lang/Closure.html) closure)

Renames a source file. The closure will be called with a single parameter, the name of the file. The closure should return a String object with a new target name. The closure may return null, in which case the original name will be used.

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) rename([**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) sourceRegEx, [**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) replaceWith)

Renames files based on a regular expression. Uses java.util.regex type of regular expressions. Note that the replace string should use the '$1' syntax to refer to capture groups in the source regular expression. Files that do not match the source regular expression will be copied with the original name.

Example:

rename '(.\*)\_OEM\_BLUE\_(.\*)', '$1$2'

would map the file 'style\_OEM\_BLUE\_.css' to 'style.css'

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) rename([**Pattern**](http://download.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html) sourceRegEx, [**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html) replaceWith)

Renames files based on a regular expression. See [CopyProcessingSpec.rename(java.lang.String, java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/CopyProcessingSpec.html#rename-java.lang.String,%20java.lang.String-).

[**AbstractCopyTask**](http://docs.google.com/dsl/org.gradle.api.tasks.AbstractCopyTask.html) rename([**Transformer**](http://docs.google.com/javadoc/org/gradle/api/Transformer.html)<[**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html), [**String**](http://download.oracle.com/javase/8/docs/api/java/lang/String.html)> renamer)

Renames a source file. The function will be called with a single parameter, the name of the file. The function should return a new target name. The function may return null, in which case the original name will be used.

[**CopySpec**](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html) with([**CopySpec**](http://docs.google.com/javadoc/org/gradle/api/file/CopySpec.html)... sourceSpecs)

Adds the given specs as a child of this spec.

def contentSpec = copySpec {  
 from("content") {  
 include "\*\*/\*.txt"  
 }  
}  
  
task copy(type: Copy) {  
 into "$buildDir/copy"  
 with contentSpec  
}

**Docs**

* [User Manual](http://docs.google.com/userguide/userguide.html)
* [DSL Reference](http://docs.google.com/dsl/)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Javadoc](http://docs.google.com/javadoc/)

**News**

* [Blog](https://blog.gradle.org/)
* [Newsletter](https://newsletter.gradle.com/)
* [Twitter](https://twitter.com/gradle)

**Products**

* [Build Scans](https://gradle.com/build-scans)
* [Build Cache](https://gradle.com/build-cache)
* [Enterprise Docs](https://gradle.com/enterprise/resources)

**Get Help**

* [Forums](https://discuss.gradle.org/c/help-discuss)
* [GitHub](https://github.com/gradle/)
* [Training](https://gradle.org/training/)
* [Services](https://gradle.org/services/)

Subscribe for important Gradle updates and news

Subscribe

By entering your email, you agree to our [Terms](https://gradle.org/terms/) and [Privacy Policy](https://gradle.org/privacy/), including receipt of emails. You can unsubscribe at any time.

© [Gradle Inc.](https://gradle.com)2018 All rights reserved.

[Careers](https://gradle.com/careers) | [Privacy](https://gradle.org/privacy) | [Terms of Service](https://gradle.org/terms) | [Contact](https://gradle.org/contact/)