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)

[PDF](http://docs.google.com/userguide/userguide.pdf)

* [User Manual Home](http://docs.google.com/userguide/userguide.html)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Installing Gradle](http://docs.google.com/userguide/installation.html)
* [Tutorials](https://guides.gradle.org/)

### Reference

* [Groovy DSL Reference](http://docs.google.com/dsl/)
* [Gradle API Javadoc](http://docs.google.com/javadoc/)
* [Core Plugins](http://docs.google.com/userguide/plugin_reference.html)
* [Gradle & Third-party Tools](http://docs.google.com/userguide/third_party_integration.html)

### Getting Started

* [Creating New Gradle Builds](https://guides.gradle.org/creating-new-gradle-builds/)
* [Creating Build Scans](https://guides.gradle.org/creating-build-scans/)
* [Migrating From Maven](https://guides.gradle.org/migrating-from-maven/)

### Running Gradle Builds

* [Command-Line Interface](http://docs.google.com/userguide/command_line_interface.html)
* [Customizing Execution](#gjdgxs)
  + [Configuring the Build Environment](http://docs.google.com/userguide/build_environment.html)
  + [Configuring the Gradle Daemon](http://docs.google.com/userguide/gradle_daemon.html)
  + [Initialization Scripts](http://docs.google.com/userguide/init_scripts.html)
* [Directory Layout](http://docs.google.com/userguide/directory_layout.html)
* [Executing Multi-Project Builds](http://docs.google.com/userguide/intro_multi_project_builds.html)
* [Gradle Wrapper](http://docs.google.com/userguide/gradle_wrapper.html)
* [Troubleshooting](http://docs.google.com/userguide/troubleshooting.html)
* [Using Build Scans](https://docs.gradle.com/build-scan-plugin)
* [Enabling and Configuring the Build Cache](http://docs.google.com/userguide/build_cache.html)
* [Integrating Separate Gradle Builds (Composite Builds)](http://docs.google.com/userguide/composite_builds.html)

### Authoring Gradle Builds

* [Fundamentals](#30j0zll)
  + [Introducing the Basics of Build Scripts](http://docs.google.com/userguide/tutorial_using_tasks.html)
  + [Working with Tasks](http://docs.google.com/userguide/more_about_tasks.html)
  + [Learning More About Build Scripts](http://docs.google.com/userguide/writing_build_scripts.html)
  + [Working with Files](http://docs.google.com/userguide/working_with_files.html)
  + [Creating Custom Task Types](http://docs.google.com/userguide/custom_tasks.html)
  + [Using Gradle Plugins](http://docs.google.com/userguide/plugins.html)
  + [The Standard Gradle Plugins](http://docs.google.com/userguide/standard_plugins.html)
  + [Understanding the Build Lifecycle](http://docs.google.com/userguide/build_lifecycle.html)
  + [Working with Logging](http://docs.google.com/userguide/logging.html)
  + [Configuring Multi-Project Builds](http://docs.google.com/userguide/multi_project_builds.html)
* [Best Practices](#1fob9te)
  + [Authoring Maintainable Build Scripts](http://docs.google.com/userguide/authoring_maintainable_build_scripts.html)
  + [Organizing Gradle Projects](http://docs.google.com/userguide/organizing_gradle_projects.html)
  + [Optimizing Build Performance](https://guides.gradle.org/performance/)
  + [Using the Build Cache](https://guides.gradle.org/using-build-cache/)
* [Dependency Management](#3znysh7)
  + [Introduction to Dependency Management](http://docs.google.com/userguide/introduction_dependency_management.html)
  + [Dependency Management Terminology](http://docs.google.com/userguide/dependency_management_terminology.html)
  + [Dependency Types](http://docs.google.com/userguide/dependency_types.html)
  + [Repository Types](http://docs.google.com/userguide/repository_types.html)
  + [Declaring Dependencies](http://docs.google.com/userguide/declaring_dependencies.html)
  + [Declaring Repositories](http://docs.google.com/userguide/declaring_repositories.html)
  + [Inspecting Dependencies](http://docs.google.com/userguide/inspecting_dependencies.html)
  + [Managing Dependency Configurations](http://docs.google.com/userguide/managing_dependency_configurations.html)
  + [Managing Transitive Dependencies](http://docs.google.com/userguide/managing_transitive_dependencies.html)
  + [Dependency Locking](http://docs.google.com/userguide/dependency_locking.html)
  + [Troubleshooting Dependency Resolution](http://docs.google.com/userguide/troubleshooting_dependency_resolution.html)
  + [Customizing Dependency Resolution Behavior](http://docs.google.com/userguide/customizing_dependency_resolution_behavior.html)
  + [Dependency Cache Internals](http://docs.google.com/userguide/dependency_cache.html)
  + [Working with Dependencies](http://docs.google.com/userguide/working_with_dependencies.html)
* [Publishing Artifacts](http://docs.google.com/userguide/artifact_management.html)
* [C++ Projects](#2et92p0)
  + [Building Native Software](http://docs.google.com/userguide/native_software.html)
  + [Software Model Concepts](http://docs.google.com/userguide/software_model_concepts.html)
  + [Rule-based Model Configuration](http://docs.google.com/userguide/software_model.html)
  + [Implementing Model Rules in a Plugin](http://docs.google.com/userguide/rule_source.html)
  + [Extending the Software Model](http://docs.google.com/userguide/software_model_extend.html)
* [Java Projects](#tyjcwt)
  + [Building Java & JVM projects](http://docs.google.com/userguide/building_java_projects.html)
  + [Testing Java & JVM projects](http://docs.google.com/userguide/java_testing.html)
* [Advanced Techniques](#3dy6vkm)
  + [Configuring Tasks Lazily](http://docs.google.com/userguide/lazy_configuration.html)
  + [Developing Parallel Tasks](https://guides.gradle.org/using-the-worker-api/)
  + [Testing Your Build with TestKit](http://docs.google.com/userguide/test_kit.html)
  + [Using Ant from Gradle](http://docs.google.com/userguide/ant.html)
* [Sample Gradle builds](#1t3h5sf)
  + [Groovy DSL Samples](https://github.com/gradle/gradle/tree/master/subprojects/docs/src/samples)
  + [Kotlin DSL Samples](https://github.com/gradle/kotlin-dsl/tree/master/samples)

### Extending Gradle

* [Writing Custom Plugins](http://docs.google.com/userguide/custom_plugins.html)
* [Plugin Development Guides](https://gradle.org/guides/?q=Plugin+Development)

[Edit this page](https://github.com/gradle/gradle/edit/master/subprojects/docs/src/docs/userguide/)

# The JaCoCo Plugin

Contents

[Getting Started](#4d34og8)

[Configuring the JaCoCo Plugin](#2s8eyo1)

[JaCoCo Report configuration](#17dp8vu)

[Enforcing code coverage metrics](#3rdcrjn)

[JaCoCo specific task configuration](#26in1rg)

[Tasks](#lnxbz9)

[Dependency management](#35nkun2)

| **✨** | The JaCoCo plugin is currently [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle). Please be aware that the DSL and other configuration may change in later Gradle versions. |
| --- | --- |

The JaCoCo plugin provides code coverage metrics for Java code via integration with [JaCoCo](http://www.eclemma.org/jacoco/).

[Getting Started](#4d34og8)

To get started, apply the JaCoCo plugin to the project you want to calculate code coverage for.

[Example: Applying the JaCoCo plugin](#1ksv4uv)

**build.gradle**

apply plugin: "jacoco"

If the Java plugin is also applied to your project, a new task named jacocoTestReport is created that depends on the test task. The report is available at *$buildDir*/reports/jacoco/test. By default, a HTML report is generated.

[Configuring the JaCoCo Plugin](#2s8eyo1)

The JaCoCo plugin adds a project extension named jacoco of type [JacocoPluginExtension](http://docs.google.com/dsl/org.gradle.testing.jacoco.plugins.JacocoPluginExtension.html), which allows configuring defaults for JaCoCo usage in your build.

[Example: Configuring JaCoCo plugin settings](#44sinio)

**build.gradle**

jacoco {  
 toolVersion = "0.8.1"  
 reportsDir = file("$buildDir/customJacocoReportDir")  
}

Table 1. Gradle defaults for JaCoCo properties

| **Property** | **Gradle default** |
| --- | --- |
| reportsDir | *$buildDir*/reports/jacoco |

[JaCoCo Report configuration](#17dp8vu)

The [JacocoReport](http://docs.google.com/dsl/org.gradle.testing.jacoco.tasks.JacocoReport.html) task can be used to generate code coverage reports in different formats. It implements the standard Gradle type [Reporting](http://docs.google.com/dsl/org.gradle.api.reporting.Reporting.html) and exposes a report container of type [JacocoReportsContainer](http://docs.google.com/javadoc/org/gradle/testing/jacoco/tasks/JacocoReportsContainer.html).

[Example: Configuring test task](#2jxsxqh)

**build.gradle**

jacocoTestReport {  
 reports {  
 xml.enabled false  
 csv.enabled false  
 html.destination file("${buildDir}/jacocoHtml")  
 }  
}



[Enforcing code coverage metrics](#3rdcrjn)

| **✨** | This feature requires the use of JaCoCo version 0.6.3 or higher. |
| --- | --- |

The [JacocoCoverageVerification](http://docs.google.com/dsl/org.gradle.testing.jacoco.tasks.JacocoCoverageVerification.html) task can be used to verify if code coverage metrics are met based on configured rules. Its API exposes the method [JacocoCoverageVerification.violationRules(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/testing/jacoco/tasks/JacocoCoverageVerification.html#violationRules-org.gradle.api.Action-) which is used as main entry point for configuring rules. Invoking any of those methods returns an instance of [JacocoViolationRulesContainer](http://docs.google.com/javadoc/org/gradle/testing/jacoco/tasks/rules/JacocoViolationRulesContainer.html) providing extensive configuration options. The build fails if any of the configured rules are not met. JaCoCo only reports the first violated rule.

Code coverage requirements can be specified for a project as a whole, for individual files, and for particular JaCoCo-specific types of coverage, e.g., lines covered or branches covered. The following example describes the syntax.

[Example: Configuring violation rules](#z337ya)

**build.gradle**

jacocoTestCoverageVerification {  
 violationRules {  
 rule {  
 limit {  
 minimum = 0.5  
 }  
 }  
  
 rule {  
 enabled = false  
 element = 'CLASS'  
 includes = ['org.gradle.\*']  
  
 limit {  
 counter = 'LINE'  
 value = 'TOTALCOUNT'  
 maximum = 0.3  
 }  
 }  
 }  
}

| **✨** | The code for this example can be found at samples/testing/jacoco/quickstart in the ‘-all’ distribution of Gradle. |
| --- | --- |

The [JacocoCoverageVerification](http://docs.google.com/dsl/org.gradle.testing.jacoco.tasks.JacocoCoverageVerification.html) task is not a task dependency of the check task provided by the Java plugin. There is a good reason for it. The task is currently not incremental as it doesn’t declare any outputs. Any violation of the declared rules would automatically result in a failed build when executing the check task. This behavior might not be desirable for all users. Future versions of Gradle might change the behavior.

[JaCoCo specific task configuration](#26in1rg)

The JaCoCo plugin adds a [JacocoTaskExtension](http://docs.google.com/dsl/org.gradle.testing.jacoco.plugins.JacocoTaskExtension.html) extension to all tasks of type [Test](http://docs.google.com/dsl/org.gradle.api.tasks.testing.Test.html). This extension allows the configuration of the JaCoCo specific properties of the test task.

[Example: Configuring test task](#3j2qqm3)

**build.gradle**

test {  
 jacoco {  
 append = false  
 destinationFile = file("$buildDir/jacoco/jacocoTest.exec")  
 classDumpDir = file("$buildDir/jacoco/classpathdumps")  
 }  
}

| **✨** | Using the configuration append = true (the default) causes the JaCoCo agent to append to a shared output file that may be left over from a different test execution. If append = true, Gradle disables caching for the Test task since it cannot guarantee the same results each time. |
| --- | --- |

**Default values of the JaCoCo Task extension**

test {  
 jacoco {  
 append = true  
 enabled = true  
 destPath = "$buildDir/jacoco"  
 includes = []  
 excludes = []  
 excludeClassLoaders = []  
 includeNoLocationClasses = false  
 sessionId = "<auto-generated value>"  
 dumpOnExit = true  
 classDumpDir = null  
 output = Output.FILE  
 address = "localhost"  
 port = 6300  
 jmx = false  
 }  
}

While all tasks of type [Test](http://docs.google.com/dsl/org.gradle.api.tasks.testing.Test.html) are automatically enhanced to provide coverage information when the java plugin has been applied, any task that implements [JavaForkOptions](http://docs.google.com/javadoc/org/gradle/process/JavaForkOptions.html) can be enhanced by the JaCoCo plugin. That is, any task that forks Java processes can be used to generate coverage information.

For example you can configure your build to generate code coverage using the application plugin.

[Example: Using application plugin to generate code coverage data](#1y810tw)

**build.gradle**

apply plugin: "application"  
apply plugin: "jacoco"  
  
mainClassName = "org.gradle.MyMain"  
  
jacoco {  
 applyTo run  
}  
  
task applicationCodeCoverageReport(type:JacocoReport){  
 executionData run  
 sourceSets sourceSets.main  
}

| **✨** | The code for this example can be found at samples/testing/jacoco/application in the ‘-all’ distribution of Gradle. |
| --- | --- |

**Coverage reports generated by applicationCodeCoverageReport**

.  
└── build  
 ├── jacoco  
 │   └── run.exec  
 └── reports  
 └── jacoco  
 └── applicationCodeCoverageReport  
 └── html  
 └── index.html

[Tasks](#lnxbz9)

For projects that also apply the Java Plugin, the JaCoCo plugin automatically adds the following tasks:

jacocoTestReport — [JacocoReport](http://docs.google.com/dsl/org.gradle.testing.jacoco.tasks.JacocoReport.html)

Generates code coverage report for the test task.

jacocoTestCoverageVerification — [JacocoCoverageVerification](http://docs.google.com/dsl/org.gradle.testing.jacoco.tasks.JacocoCoverageVerification.html)

Verifies code coverage metrics based on specified rules for the test task.

[Dependency management](#35nkun2)

The JaCoCo plugin adds the following dependency configurations:

Table 2. JaCoCo plugin - dependency configurations

| **Name** | **Meaning** |
| --- | --- |
| jacocoAnt | The JaCoCo Ant library used for running the JacocoReport, JacocoMerge and JacocoCoverageVerification tasks. |
| jacocoAgent | The JaCoCo agent library used for instrumenting the code under test. |

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/)