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/)

# Using Ant from Gradle

Contents

[Using Ant tasks and types in your build](#4d34og8)

[Importing an Ant build](#2s8eyo1)

[Ant properties and references](#17dp8vu)

[Ant logging](#3rdcrjn)

[API](#26in1rg)

Gradle provides excellent integration with Ant. You can use individual Ant tasks or entire Ant builds in your Gradle builds. In fact, you will find that it’s far easier and more powerful using Ant tasks in a Gradle build script, than it is to use Ant’s XML format. You could even use Gradle simply as a powerful Ant task scripting tool.

Ant can be divided into two layers. The first layer is the Ant language. It provides the syntax for the build.xml file, the handling of the targets, special constructs like macrodefs, and so on. In other words, everything except the Ant tasks and types. Gradle understands this language, and allows you to import your Ant build.xml directly into a Gradle project. You can then use the targets of your Ant build as if they were Gradle tasks.

The second layer of Ant is its wealth of Ant tasks and types, like javac, copy or jar. For this layer Gradle provides integration simply by relying on Groovy, and the fantastic AntBuilder.

Finally, since build scripts are Groovy scripts, you can always execute an Ant build as an external process. Your build script may contain statements like: "ant clean compile".execute().[[1](#lnxbz9)]

You can use Gradle’s Ant integration as a path for migrating your build from Ant to Gradle. For example, you could start by importing your existing Ant build. Then you could move your dependency declarations from the Ant script to your build file. Finally, you could move your tasks across to your build file, or replace them with some of Gradle’s plugins. This process can be done in parts over time, and you can have a working Gradle build during the entire process.

[Using Ant tasks and types in your build](#4d34og8)

In your build script, a property called ant is provided by Gradle. This is a reference to an [AntBuilder](http://docs.google.com/javadoc/org/gradle/api/AntBuilder.html) instance. This AntBuilder is used to access Ant tasks, types and properties from your build script. There is a very simple mapping from Ant’s build.xml format to Groovy, which is explained below.

You execute an Ant task by calling a method on the AntBuilder instance. You use the task name as the method name. For example, you execute the Ant echo task by calling the ant.echo() method. The attributes of the Ant task are passed as Map parameters to the method. Below is an example of the echo task. Notice that we can also mix Groovy code and the Ant task markup. This can be extremely powerful.

[Example: Using an Ant task](#35nkun2)

**build.gradle**

task hello {  
 doLast {  
 String greeting = 'hello from Ant'  
 ant.echo(message: greeting)  
 }  
}

**Output of gradle hello**

> gradle hello  
  
> Task :hello  
[ant:echo] hello from Ant  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

You pass nested text to an Ant task by passing it as a parameter of the task method call. In this example, we pass the message for the echo task as nested text:

[Example: Passing nested text to an Ant task](#1ksv4uv)

**build.gradle**

task hello {  
 doLast {  
 ant.echo('hello from Ant')  
 }  
}

**Output of gradle hello**

> gradle hello  
  
> Task :hello  
[ant:echo] hello from Ant  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

You pass nested elements to an Ant task inside a closure. Nested elements are defined in the same way as tasks, by calling a method with the same name as the element we want to define.

[Example: Passing nested elements to an Ant task](#44sinio)

**build.gradle**

task zip {  
 doLast {  
 ant.zip(destfile: 'archive.zip') {  
 fileset(dir: 'src') {  
 include(name: '\*\*.xml')  
 exclude(name: '\*\*.java')  
 }  
 }  
 }  
}

You can access Ant types in the same way that you access tasks, using the name of the type as the method name. The method call returns the Ant data type, which you can then use directly in your build script. In the following example, we create an Ant path object, then iterate over the contents of it.

[Example: Using an Ant type](#2jxsxqh)

**build.gradle**

task list {  
 doLast {  
 def path = ant.path {  
 fileset(dir: 'libs', includes: '\*.jar')  
 }  
 path.list().each {  
 println it  
 }  
 }  
}

More information about AntBuilder can be found in 'Groovy in Action' 8.4 or at the [Groovy Wiki](http://groovy-lang.org/scripting-ant.html).

[Using custom Ant tasks in your build](#z337ya)

To make custom tasks available in your build, you can use the taskdef (usually easier) or typedef Ant task, just as you would in a build.xml file. You can then refer to the custom Ant task as you would a built-in Ant task.

[Example: Using a custom Ant task](#3j2qqm3)

**build.gradle**

task check {  
 doLast {  
 ant.taskdef(resource: 'checkstyletask.properties') {  
 classpath {  
 fileset(dir: 'libs', includes: '\*.jar')  
 }  
 }  
 ant.checkstyle(config: 'checkstyle.xml') {  
 fileset(dir: 'src')  
 }  
 }  
}

You can use Gradle’s dependency management to assemble the classpath to use for the custom tasks. To do this, you need to define a custom configuration for the classpath, then add some dependencies to the configuration. This is described in more detail in [Declaring Dependencies](http://docs.google.com/declaring_dependencies.html).

[Example: Declaring the classpath for a custom Ant task](#1y810tw)

**build.gradle**

configurations {  
 pmd  
}  
  
dependencies {  
 pmd group: 'pmd', name: 'pmd', version: '4.2.5'  
}

To use the classpath configuration, use the asPath property of the custom configuration.

[Example: Using a custom Ant task and dependency management together](#4i7ojhp)

**build.gradle**

task check {  
 doLast {  
 ant.taskdef(name: 'pmd',  
 classname: 'net.sourceforge.pmd.ant.PMDTask',  
 classpath: configurations.pmd.asPath)  
 ant.pmd(shortFilenames: 'true',  
 failonruleviolation: 'true',  
 rulesetfiles: file('pmd-rules.xml').toURI().toString()) {  
 formatter(type: 'text', toConsole: 'true')  
 fileset(dir: 'src')  
 }  
 }  
}

[Importing an Ant build](#2s8eyo1)

You can use the ant.importBuild() method to import an Ant build into your Gradle project. When you import an Ant build, each Ant target is treated as a Gradle task. This means you can manipulate and execute the Ant targets in exactly the same way as Gradle tasks.

[Example: Importing an Ant build](#2xcytpi)

**build.gradle**

ant.importBuild 'build.xml'

**build.xml**

<project>  
 <target name="hello">  
 <echo>Hello, from Ant</echo>  
 </target>  
</project>

**Output of gradle hello**

> gradle hello  
  
> Task :hello  
[ant:echo] Hello, from Ant  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

You can add a task which depends on an Ant target:

[Example: Task that depends on Ant target](#1ci93xb)

**build.gradle**

ant.importBuild 'build.xml'  
  
task intro(dependsOn: hello) {  
 doLast {  
 println 'Hello, from Gradle'  
 }  
}

**Output of gradle intro**

> gradle intro  
  
> Task :hello  
[ant:echo] Hello, from Ant  
  
> Task :intro  
Hello, from Gradle  
  
BUILD SUCCESSFUL in 0s  
2 actionable tasks: 2 executed

Or, you can add behaviour to an Ant target:

[Example: Adding behaviour to an Ant target](#3whwml4)

**build.gradle**

ant.importBuild 'build.xml'  
  
hello {  
 doLast {  
 println 'Hello, from Gradle'  
 }  
}

**Output of gradle hello**

> gradle hello  
  
> Task :hello  
[ant:echo] Hello, from Ant  
Hello, from Gradle  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

It is also possible for an Ant target to depend on a Gradle task:

[Example: Ant target that depends on Gradle task](#2bn6wsx)

**build.gradle**

ant.importBuild 'build.xml'  
  
task intro {  
 doLast {  
 println 'Hello, from Gradle'  
 }  
}

**build.xml**

<project>  
 <target name="hello" depends="intro">  
 <echo>Hello, from Ant</echo>  
 </target>  
</project>

**Output of gradle hello**

> gradle hello  
  
> Task :intro  
Hello, from Gradle  
  
> Task :hello  
[ant:echo] Hello, from Ant  
  
BUILD SUCCESSFUL in 0s  
2 actionable tasks: 2 executed

Sometimes it may be necessary to “rename” the task generated for an Ant target to avoid a naming collision with existing Gradle tasks. To do this, use the [AntBuilder.importBuild(java.lang.Object, org.gradle.api.Transformer)](http://docs.google.com/javadoc/org/gradle/api/AntBuilder.html#importBuild-java.lang.Object-org.gradle.api.Transformer-) method.

[Example: Renaming imported Ant targets](#qsh70q)

**build.gradle**

ant.importBuild('build.xml') { antTargetName ->  
 'a-' + antTargetName  
}

**build.xml**

<project>  
 <target name="hello">  
 <echo>Hello, from Ant</echo>  
 </target>  
</project>

**Output of gradle a-hello**

> gradle a-hello  
  
> Task :a-hello  
[ant:echo] Hello, from Ant  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

Note that while the second argument to this method should be a [Transformer](http://docs.google.com/javadoc/org/gradle/api/Transformer.html), when programming in Groovy we can simply use a closure instead of an anonymous inner class (or similar) due to [Groovy’s support for automatically coercing closures to single-abstract-method types](http://mrhaki.blogspot.ie/2013/11/groovy-goodness-implicit-closure.html).

[Ant properties and references](#17dp8vu)

There are several ways to set an Ant property, so that the property can be used by Ant tasks. You can set the property directly on the AntBuilder instance. The Ant properties are also available as a Map which you can change. You can also use the Ant property task. Below are some examples of how to do this.

[Example: Setting an Ant property](#3as4poj)

**build.gradle**

ant.buildDir = buildDir  
ant.properties.buildDir = buildDir  
ant.properties['buildDir'] = buildDir  
ant.property(name: 'buildDir', location: buildDir)

**build.xml**

<echo>buildDir = ${buildDir}</echo>

Many Ant tasks set properties when they execute. There are several ways to get the value of these properties. You can get the property directly from the AntBuilder instance. The Ant properties are also available as a Map. Below are some examples.

[Example: Getting an Ant property](#1pxezwc)

**build.xml**

<property name="antProp" value="a property defined in an Ant build"/>

**build.gradle**

println ant.antProp  
println ant.properties.antProp  
println ant.properties['antProp']

There are several ways to set an Ant reference:

[Example: Setting an Ant reference](#49x2ik5)

**build.gradle**

ant.path(id: 'classpath', location: 'libs')  
ant.references.classpath = ant.path(location: 'libs')  
ant.references['classpath'] = ant.path(location: 'libs')

**build.xml**

<path refid="classpath"/>

There are several ways to get an Ant reference:

[Example: Getting an Ant reference](#2p2csry)

**build.xml**

<path id="antPath" location="libs"/>

**build.gradle**

println ant.references.antPath  
println ant.references['antPath']

[Ant logging](#3rdcrjn)

Gradle maps Ant message priorities to Gradle log levels so that messages logged from Ant appear in the Gradle output. By default, these are mapped as follows:

Table 1. Ant message priority mapping

| **Ant Message Priority** | **Gradle Log Level** |
| --- | --- |
| *VERBOSE* | DEBUG |
| *DEBUG* | DEBUG |
| *INFO* | INFO |
| *WARN* | WARN |
| *ERROR* | ERROR |

[Fine tuning Ant logging](#147n2zr)

The default mapping of Ant message priority to Gradle log level can sometimes be problematic. For example, there is no message priority that maps directly to the LIFECYCLE log level, which is the default for Gradle. Many Ant tasks log messages at the *INFO* priority, which means to expose those messages from Gradle, a build would have to be run with the log level set to INFO, potentially logging much more output than is desired.

Conversely, if an Ant task logs messages at too high of a level, to suppress those messages would require the build to be run at a higher log level, such as QUIET. However, this could result in other, desirable output being suppressed.

To help with this, Gradle allows the user to fine tune the Ant logging and control the mapping of message priority to Gradle log level. This is done by setting the priority that should map to the default Gradle LIFECYCLE log level using the [AntBuilder.setLifecycleLogLevel(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/AntBuilder.html#setLifecycleLogLevel-java.lang.String-) method. When this value is set, any Ant message logged at the configured priority or above will be logged at least at LIFECYCLE. Any Ant message logged below this priority will be logged at most at INFO.

For example, the following changes the mapping such that Ant *INFO* priority messages are exposed at the LIFECYCLE log level.

[Example: Fine tuning Ant logging](#3o7alnk)

**build.gradle**

ant.lifecycleLogLevel = "INFO"  
  
task hello {  
 doLast {  
 ant.echo(level: "info", message: "hello from info priority!")  
 }  
}

**Output of gradle hello**

> gradle hello  
  
> Task :hello  
[ant:echo] hello from info priority!  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

On the other hand, if the lifecycleLogLevel was set to *ERROR*, Ant messages logged at the *WARN* priority would no longer be logged at the WARN log level. They would now be logged at the INFO level and would be suppressed by default.

[API](#26in1rg)

The Ant integration is provided by [AntBuilder](http://docs.google.com/javadoc/org/gradle/api/AntBuilder.html).

[1](#23ckvvd). In Groovy you can execute Strings. To learn more about executing external processes with Groovy have a look in 'Groovy in Action' 9.3.2 or at the Groovy wiki

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/)