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/)

# Maven Publish Plugin

Contents

[Usage](#4d34og8)

[Tasks](#2s8eyo1)

[Publications](#17dp8vu)

[Repositories](#3rdcrjn)

[Publishing to Maven Local](#26in1rg)

[Complete example](#lnxbz9)

[Removal of deferred configuration behavior](#35nkun2)

The Maven Publish Plugin provides the ability to publish build artifacts to an [Apache Maven](http://maven.apache.org/) repository. A module published to a Maven repository can be consumed by Maven, Gradle (see [Declaring Dependencies](http://docs.google.com/declaring_dependencies.html#declaring_dependencies)) and other tools that understand the Maven repository format. You can learn about the fundamentals of publishing in [Publishing Overview](http://docs.google.com/publishing_overview.html#publishing_overview).

[Usage](#4d34og8)

To use the Maven Publish Plugin, include the following in your build script:

[Example: Applying the Maven Publish Plugin](#1ksv4uv)

**build.gradle**

plugins {  
 id 'maven-publish'  
}

The Maven Publish Plugin uses an extension on the project named publishing of type [PublishingExtension](http://docs.google.com/dsl/org.gradle.api.publish.PublishingExtension.html). This extension provides a container of named publications and a container of named repositories. The Maven Publish Plugin works with [MavenPublication](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html) publications and [MavenArtifactRepository](http://docs.google.com/dsl/org.gradle.api.artifacts.repositories.MavenArtifactRepository.html) repositories.

[Tasks](#2s8eyo1)

generatePomFileFor*PubName*Publication — [GenerateMavenPom](http://docs.google.com/dsl/org.gradle.api.publish.maven.tasks.GenerateMavenPom.html)

Creates a POM file for the publication named *PubName*, populating the known metadata such as project name, project version, and the dependencies. The default location for the POM file is *build/publications/$pubName/pom-default.xml*.

publish*PubName*PublicationTo*RepoName*Repository — [PublishToMavenRepository](http://docs.google.com/dsl/org.gradle.api.publish.maven.tasks.PublishToMavenRepository.html)

Publishes the *PubName* publication to the repository named *RepoName*. If you have a repository definition without an explicit name, *RepoName* will be "Maven".

publish*PubName*PublicationToMavenLocal — [PublishToMavenLocal](http://docs.google.com/javadoc/org/gradle/api/publish/maven/tasks/PublishToMavenLocal.html)

Copies the *PubName* publication to the local Maven cache — typically *$USER\_HOME/.m2/repository* — along with the publication’s POM file and other metadata.

publish

*Depends on*: All publish*PubName*PublicationTo*RepoName*Repository tasks

An aggregate task that publishes all defined publications to all defined repositories. It does *not* include copying publications to the local Maven cache.

publishToMavenLocal

*Depends on*: All publish*PubName*PublicationToMavenLocal tasks

Copies all defined publications to the local Maven cache, including their metadata (POM files, etc.).

[Publications](#17dp8vu)

This plugin provides [publications](http://docs.google.com/publishing_overview.html#glossary:publication) of type [MavenPublication](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html). To learn how to define and use publications, see the section on [basic publishing](http://docs.google.com/publishing_overview.html#sec:basic_publishing).

There are four main things you can configure in a Maven publication:

* A [component](http://docs.google.com/publishing_overview.html#glossary:component) — via [MavenPublication.from(org.gradle.api.component.SoftwareComponent)](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html#org.gradle.api.publish.maven.MavenPublication:from(org.gradle.api.component.SoftwareComponent)).
* [Custom artifacts](http://docs.google.com/publishing_overview.html#sec:publishing_custom_artifacts_to_maven) — via the [MavenPublication.artifact(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html#org.gradle.api.publish.maven.MavenPublication:artifact(java.lang.Object)) method. See [MavenArtifact](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenArtifact.html) for the available configuration options for custom Maven artifacts.
* Standard metadata like artifactId, groupId and version.
* Other contents of the POM file — via [MavenPublication.pom(org.gradle.api.Action)](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html#org.gradle.api.publish.maven.MavenPublication:pom(org.gradle.api.Action)).

You can see all of these in action in the [complete publishing example](#lnxbz9). The API documentation for MavenPublication has additional code samples.

[Identity values in the generated POM](#44sinio)

The attributes of the generated POM file will contain identity values derived from the following project properties:

* groupId - [Project.getGroup()](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:group)
* artifactId - [Project.getName()](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:name)
* version - [Project.getVersion()](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:version)

Overriding the default identity values is easy: simply specify the groupId, artifactId or version attributes when configuring the [MavenPublication](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html).

[Example: customizing the publication identity](#2jxsxqh)

**build.gradle**

publishing {  
 publications {  
 maven(MavenPublication) {  
 groupId = 'org.gradle.sample'  
 artifactId = 'project1-sample'  
 version = '1.1'  
  
 from components.java  
 }  
 }  
}

| **💡** | Certain repositories will not be able to handle all supported characters. For example, the : character cannot be used as an identifier when publishing to a filesystem-backed repository on Windows. |
| --- | --- |

Maven restricts groupId and artifactId to a limited character set ([A-Za-z0-9\_\\-.]+) and Gradle enforces this restriction. For version (as well as the artifact extension and classifier properties), Gradle will handle any valid Unicode character.

The only Unicode values that are explicitly prohibited are \, / and any ISO control character. Supplied values are validated early in publication.

[Customizing the generated POM](#z337ya)

The generated POM file can be customized before publishing. For example, when publishing a library to Maven Central you will need to set certain metadata. The Maven Publish Plugin provides a DSL for that purpose. Please see [MavenPom](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPom.html) in the DSL Reference for the complete documentation of available properties and methods. The following sample shows how to use the most common ones:

[Example: Customizing the POM file](#3j2qqm3)

**build.gradle**

publishing {  
 publications {  
 mavenJava(MavenPublication) {  
 pom {  
 name = 'My Library'  
 description = 'A concise description of my library'  
 url = 'http://www.example.com/library'  
 licenses {  
 license {  
 name = 'The Apache License, Version 2.0'  
 url = 'http://www.apache.org/licenses/LICENSE-2.0.txt'  
 }  
 }  
 developers {  
 developer {  
 id = 'johnd'  
 name = 'John Doe'  
 email = 'john.doe@example.com'  
 }  
 }  
 scm {  
 connection = 'scm:git:git://example.com/my-library.git'  
 developerConnection = 'scm:git:ssh://example.com/my-library.git'  
 url = 'http://example.com/my-library/'  
 }  
 }  
 }  
 }  
}

[Repositories](#3rdcrjn)

This plugin provides [repositories](http://docs.google.com/dependency_management_terminology.html#sub:terminology_repository) of type [MavenArtifactRepository](http://docs.google.com/dslorg.gradle.api.artifacts.repositories.MavenArtifactRepository.html). To learn how to define and use repositories for publishing, see the section on [basic publishing](http://docs.google.com/publishing_overview.html#sec:basic_publishing).

Here’s a simple example of defining a publishing repository:

[Example: Declaring repositories to publish to](#1y810tw)

**build.gradle**

publishing {  
 repositories {  
 maven {  
 // change to point to your repo, e.g. http://my.org/repo  
 url = "$buildDir/repo"  
 }  
 }  
}

The two main things you will want to configure are the repository’s:

* URL (required)
* Name (optional)

You can define multiple repositories as long as they have unique names within the build script. You may also declare one (and only one) repository without a name. That repository will take on an implicit name of "Maven".

You can also configure any authentication details that are required to connect to the repository. See [MavenArtifactRepository](http://docs.google.com/dslorg.gradle.api.artifacts.repositories.MavenArtifactRepository.html) for more details.

[Snapshot and release repositories](#4i7ojhp)

It is a common practice to publish snapshots and releases to different Maven repositories. A simple way to accomplish this is to configure the repository URL based on the project version. The following sample uses one URL for versions that end with "SNAPSHOT" and a different URL for the rest:

[Example: Configuring repository URL based on project version](#2xcytpi)

**build.gradle**

publishing {  
 repositories {  
 maven {  
 def releasesRepoUrl = "$buildDir/repos/releases"  
 def snapshotsRepoUrl = "$buildDir/repos/snapshots"  
 url = version.endsWith('SNAPSHOT') ? snapshotsRepoUrl : releasesRepoUrl  
 }  
 }  
}

Similarly, you can use a [project or system property](http://docs.google.com/build_environment.html#build_environment) to decide which repository to publish to. The following example uses the release repository if the project property release is set, such as when a user runs gradle -Prelease publish:

[Example: Configuring repository URL based on project property](#1ci93xb)

**build.gradle**

publishing {  
 repositories {  
 maven {  
 def releasesRepoUrl = "$buildDir/repos/releases"  
 def snapshotsRepoUrl = "$buildDir/repos/snapshots"  
 url = project.hasProperty('release') ? releasesRepoUrl : snapshotsRepoUrl  
 }  
 }  
}

[Publishing to Maven Local](#26in1rg)

For integration with a local Maven installation, it is sometimes useful to publish the module into the Maven local repository (typically at *$USER\_HOME/.m2/repository*), along with its POM file and other metadata. In Maven parlance, this is referred to as 'installing' the module.

The Maven Publish Plugin makes this easy to do by automatically creating a [PublishToMavenLocal](http://docs.google.com/javadoc/org/gradle/api/publish/maven/tasks/PublishToMavenLocal.html) task for each [MavenPublication](http://docs.google.com/dsl/org.gradle.api.publish.maven.MavenPublication.html) in the publishing.publications container. The task name follows the pattern of publish*PubName*PublicationToMavenLocal. Each of these tasks is wired into the publishToMavenLocal aggregate task. You do not need to have mavenLocal() in your publishing.repositories section.

[Complete example](#lnxbz9)

The following example demonstrates how to sign and publish a Java library including sources, Javadoc, and a customized POM:

[Example: Publishing a Java library](#3whwml4)

**build.gradle**

plugins {  
 id 'java-library'  
 id 'maven-publish'  
 id 'signing'  
}  
  
group = 'com.example'  
version = '1.0'  
  
task sourcesJar(type: Jar) {  
 from sourceSets.main.allJava  
 classifier = 'sources'  
}  
  
task javadocJar(type: Jar) {  
 from javadoc  
 classifier = 'javadoc'  
}  
  
publishing {  
 publications {  
 mavenJava(MavenPublication) {  
 artifactId = 'my-library'  
 from components.java  
 artifact sourcesJar  
 artifact javadocJar  
 pom {  
 name = 'My Library'  
 description = 'A concise description of my library'  
 url = 'http://www.example.com/library'  
 licenses {  
 license {  
 name = 'The Apache License, Version 2.0'  
 url = 'http://www.apache.org/licenses/LICENSE-2.0.txt'  
 }  
 }  
 developers {  
 developer {  
 id = 'johnd'  
 name = 'John Doe'  
 email = 'john.doe@example.com'  
 }  
 }  
 scm {  
 connection = 'scm:git:git://example.com/my-library.git'  
 developerConnection = 'scm:git:ssh://example.com/my-library.git'  
 url = 'http://example.com/my-library/'  
 }  
 }  
 }  
 }  
 repositories {  
 maven {  
 // change URLs to point to your repos, e.g. http://my.org/repo  
 def releasesRepoUrl = "$buildDir/repos/releases"  
 def snapshotsRepoUrl = "$buildDir/repos/snapshots"  
 url = version.endsWith('SNAPSHOT') ? snapshotsRepoUrl : releasesRepoUrl  
 }  
 }  
}  
  
signing {  
 sign publishing.publications.mavenJava  
}  
  
  
javadoc {  
 if(JavaVersion.current().isJava9Compatible()) {  
 options.addBooleanOption('html4', true)  
 }  
}

The result is that the following artifacts will be published:

* The POM: my-library-1.0.pom
* The primary JAR artifact for the Java component: my-library-1.0.jar
* The sources JAR artifact that has been explicitly configured: my-library-1.0-sources.jar
* The Javadoc JAR artifact that has been explicitly configured: my-library-1.0-javadoc.jar

The [Signing Plugin](http://docs.google.com/signing_plugin.html#signing_plugin) is used to generate a signature file for each artifact. In addition, checksum files will be generated for all artifacts and signature files.

[Removal of deferred configuration behavior](#35nkun2)

| **✨** | Gradle 5.0 will change the behavior of the publishing {} block. Read on to find out how you can make your build compatible today. |
| --- | --- |

Prior to Gradle 4.8, the publishing {} block was implicitly treated as if all the logic inside it was executed after the project is evaluated. This caused quite a bit of confusion, because it was the only block that behaved that way. As part of the stabilization effort in Gradle 4.8, we are deprecating this behavior and asking all users to migrate their build.

The new, stable behavior can be switched on by adding the following to your settings file:

enableFeaturePreview('STABLE\_PUBLISHING')

We recommend doing a test run with a local repository to see whether all artifacts still have the expected coordinates. In most cases everything should work as before and you are done.

If the coordinates change unexpectedly, you may have some logic inside your publishing block or in a plugin that is depending on the deferred configuration behavior. For instance, the following logic assumes that the subprojects will be evaluated when the artifactId is set:

subprojects {  
 publishing {  
 publications {  
 mavenJava {  
 from components.java  
 artifactId = jar.baseName  
 }  
 }  
 }  
}

This kind of logic must be wrapped in an afterEvaluate {} block to make it work going forward.

subprojects {  
 publishing {  
 publications {  
 mavenJava {  
 from components.java  
 afterEvaluate {  
 artifactId = jar.baseName  
 }  
 }  
 }  
 }  
}

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/)