Activating and Deactivating Bundles

Contents

[How to 1](#_Toc349760034)

[Activating and Deactivating Bundle Projects 1](#_Toc349760035)

[Project Modifications when Activating and Deactivating 2](#_Toc349760036)

[Bundle Status and Activation States 2](#_Toc349760037)

[Activated and Deactivated Bundles 2](#_Toc349760038)

[Activation Summary 2](#_Toc349760039)

[Bundle Project Status 3](#_Toc349760040)

[Creating, Opening, Importing, Deleting and Closing Bundle Projects 3](#_Toc349760041)

[Activating and Deactivating Dependent Bundles 3](#_Toc349760042)

[Dynamic Import and Buddy Policy 3](#_Toc349760043)

[Fragment Bundles 4](#_Toc349760044)

[Bundle Errors 4](#_Toc349760045)

# How to

You activate and deactivate collections of bundle projects in one operation from the Bundle main menu and individual bundles from the pop-up context menu in Package Explorer, the Bundle Details Page View or the Bundle List Page View. Activation and deactivation of individual bundles are also available from local tool bars in the Details and List Page View.

# Activating and Deactivating Bundle Projects

A bundle project is activated when the project is assigned an in-place specific nature and its associated bundle is installed, resolved and possibly started.

The workspace is said to be activated when the first of the set of possible workspace bundle projects is activated. When the workspace is activated, that is when one or more bundle projects are activated, the set of deactivated bundles in the activated workspace are in state INSTALLED and the activated bundles are in state RESOLVED, STARTING/<<LAZY>>, ACTIVE or STOPPING.

When the last activated bundle in the workspace is deactivated and the nature is removed from the project, the workspace as a whole is deactivated, and all bundles in the workspace are in state UNINSTALLED.

Exceptions to this rule are when errors exists in already activated bundle projects at session start up (IDE or OSGi framework) and when activated bundle projects with errors are opened or imported. See [Bundle Project Errors](../concepts/Bundle%20Project%20Errors.htm) for details.

## Project Modifications when Activating and Deactivating

When a project is activated the following modifications are made to the project:

1. An in-place nature, called JavaTime, is added to the .project file.
2. If the [Update Bundle-ClassPath on Activate/Deactivate](Setting%20Bundle%20Options.htm#AddbintoBundleClassPathonActivate) option is switched on (default) the defined output folder of the project is added to the Bundle-Class-Path header in the manifest file of the project. An alternative to update the manifest file is to use the osgi.dev system property.
3. If the [Set Activation Policy to Eager on Activate](Setting%20Bundle%20Options.htm#SetActivationPolicyToEagerOnActivate) is switched on the Bundle-ActivationPolicy header, if any, is removed from the manifest file. This header can be removed (eager activation) and added (lazy activation) from one of the menus in the bundle view (See [Eager Activation Policy](Setting%20Bundle%20Options.htm#EagerActivationPolicy)), from the overview tab in the manifest editor or directly by editing the manifest file.

No modifications are performed on the project while the bundle project stays activated.

When a project is deactivated the in-place nature is removed from the .project file and, as long as the *Update Bundle-ClassPath on Activate/Deactivate* option is on, the added output folder is removed from the Bundle-ClassPath header in the manifest file. If the output folder is the only entry in the Bundle-ClassPath header, the header is removed. Otherwise only the output folder path is removed from the header.

In summary, if the activation policy is not set to eager on activation, and the option for updating the Bundle-ClassPath is the same as it was at activation time, the project will not contain any modifications inserted by InPlace Activator when deactivated.

# Bundle Mode and Activation States

When a workspace bundle is activated it is started and loaded in to memory and executed as any ordinary deployed bundle. You can then control the state of the activated bundle by refreshing, starting, stopping, resetting and updating it. When deactivated, you only have the option to activate the bundle. See [Bundle States and Transitions](../concepts/Dynamic%20Workspace%20Bundles.htm#BundleStatesAndTransitions) for a detailed description of the possible states a bundle may hold.

## Activated and Deactivated Bundles

In OSGi terminology a bundle is activated when started (state ACTIVE or <<LAZY>>) and deactivated when stopped (state RESOLVED). For workspace bundles a bundle is activated in the workspace when in state RESOLVED, ACTIVE, STARTING/<<LAZY>>, STOPPING and deactivated when in state INSTALLED or UNINSTALLED.

This means that a bundle is started the first time it is activated and stays activated even if it is stopped and started multiple times. The bundle becomes deactivated when an explicit deactivate command is issued, moving the bundle to one of the two deactivated states. To avoid confusion the terms start and stop bundles are used instead of activate and deactivate for the OSGi Start and Stop operations.

## Activation Summary

* If at least one bundle is activated, the workspace is said to be activated and if no workspace bundles are activated the workspace is deactivated.
* Activated bundles are in state RESOLVED, STARTING/<<LAZY>>, ACTIVE or STOPPING.
* Deactivated bundles are in state UNINSTALLED when the workspace is deactivated and in state INSTALLED when the workspace is activated.

## Bundle Project Status

The status attribute of a bundle project is first dependent on its build state. The build state is independent of the activation mode (deactivated or activated) and the state (UNINSTALLED, INSTALLED, RESOLVED, STARTING or ACTIVE) of a bundle project:

* “Missing Build State” status is flagged when a project has not been built at least once. No bundle operations are performed on bundles when it corresponding project has no build state. If the bundle is, at least in state RESOLVED before the build state is missing the current revision is used.
* “Build Problems” status is flagged when a project has compile time errors. Bundle projects with build errors are not installed or updated. Instead the current revision of the bundle is used.

A deactivated bundle is in state INSTALLED in an activated workspace – that is if one or bundles are activated – and in state UNINSTALLED in a deactivated workspace (no bundles are activated). A deactivated bundle with no build errors

The mode of a bundle project is either activated or deactivated. There is also a status field in the Bundle View flagging build problems, runtime errors and pending operations. Details about build errors are found in the Problem View and for runtime errors the details are found in the Error View. Pending operations are not errors but a reminder on the next logical operation to be performed on a bundle to reach a logical target state. For instance if the Update on Build option is switched on when a bundle is built, the next logical pending operation would be to update the bundle. Before the bundle is updated the compiled version of the bundle project is not synchronized with the running instance of the bundle. The target state is usually a bundle in state ACTIVE.

If there are no errors and no pending operations the bundles status is marked as OK.

The build statuses are used independent of the activation mode:

* “Missing Build State” when the build status is such that it is impossible to install or update a bundle. A re-activation or a new build is usually needed to be able to install or update the bundle.
* “Build Problems” when a project has compile time errors. Bundle projects with build errors are not installed or updated. Instead the current revision of the bundle is used.

Specific statuses for activated bundle projects are:

* “Build Pending” if a project is not built after a saved change of relevant resources in the project.  
  Note that dependent (requiring) projects may also get built when the pending project is built. This is usually the case if the Eclipse “Build Automatically” option is switched off. The “Build Pending” status is not maintained across Eclipse sessions.
* “Bundle problems” when a runtime error related to a bundle occurs while trying to execute a bundle operation or any project CRUD operation affecting activated bundle projects. Runtime errors are reported in the “Error View” and the “Message View” and sent to the console if the “–consoleLog” runtime option or the “eclipse.consoleLog” system property is set to “true”.
* “External Transition” when an OSGi command is executed from an external source as for instance the OSGi Host Console or another bundle.
* “Update Pending” if the activated bundle has been built but not updated. This is the case if the [Update On Build](Setting%20Bundle%20Options.htm#UpdateOnBuild) option is switched off and the bundle has not been updated manually. Dependent (requiring) bundles are also updated when the pending bundle is updated.
* “Refresh Pending” if the bundle has been updated, have requiring bundles and not refreshed yet. This is the case if the [Refresh On Update](Setting%20Bundle%20Options.htm#RefreshOnUpdate) is switched off and the bundle has more than one revision and not refreshed manually. Number of revisions are shown in parenthesis after the status label. Dependent (requiring) bundles are also refreshed when the pending bundle is refreshed.
* “Lazy Loading” when a bundle with a lazy activation policy is started and waits for other bundles to cause classes of the lazy activated bundle to be loaded.

# Creating, Opening, Importing, Deleting and Closing Bundle Projects

When a new project is created the bundle project is by default in a deactivated position. If you open or import a project, the bundle is activated if it was in an activated position when the project was closed or exported. A bundle is uninstalled (not deactivated) before a project is closed or deleted.

# Activating and Deactivating Dependent Bundles

If you activate a bundle or import or open an earlier activated project bundle that is dependent on a deactivated bundle (e.g. the deactivated bundle appears in the import header of the newly activated bundle), the deactivated (or providing) bundle is activated implicit. If you on the other hand deactivate, close or delete an activated project bundle that other bundles depend on, the dependent (or requiring) bundles are deactivated implicit. Implicit activation also happens when you change the bundle dependency relationships to other bundle projects (e.g. add a deactivated bundle to the import header of an activated bundle), and then update the bundle with the new dependencies.

As a result of dependencies between bundles, activation, updating and deactivation of bundles may implicit cause other bundles to be activated or deactivated. This is transitive with a cascading effect activating or deactivating a chain of bundles from an initial bundle that directly or indirectly has dependency relationships to other bundles.

# Dynamic Import and Buddy Policy

When using the DynamicImport-Package or the Eclipse-BuddyPolicy header, classes to load in other bundles is not known in advance. In such use cases potential classes in bundles to load should be activated manually on beforehand.

The DynamicImport-Package header bypass the static dependency rules and looks for exported packages when needed and does not know in advance the class name it may request to load. A bundle may successfully resolve if a suitable optional package is present, but to assure that exported packages in candidate bundles for dynamic import are loaded they should be activated manually before activating the bundle with the DynamicImport-Package header.

The Eclipse-BuddyPolicy header does not require an import header to load classes from other bundles. By activating candidate bundles or the whole workspace manually, you are certain that those bundles will be available at runtime.

# Fragment Bundles

Fragment bundles are attached to and use the same class loader as their host bundle. The fragment bundle is part of the host bundle and has requirements on the host bundle. The host bundle is therefore activated implicit when the fragment is activated (installed and resolved). A fragment bundle has no activator and can therefore not be started (ACTIVE).

Because the host has no requirements on the fragment, the fragment is not activated when the host is activated. To enable automatic activation of the fragment when the host is activated select the *Requiring and Providing* option on Activate in the Partial Dependencies dialog.

Also note that the other requirements and capabilities in a fragment bundle never become part of the fragment’s Bundle Wiring; - they are treated as part of the host’s requirements and capabilities when the fragment is attached to that host.

If you resolve (e.g. using activate, update or reset) a fragment after the host bundle has been activated you need to refresh the host bundle manually for the newly resolved fragment bundle to attach to the host bundle. It is by design to not automatic refresh the host bundle after the fragment is resolved.

# Bundle Errors

It is not possible to activate a bundle, or any bundle requiring capabilities from a bundle, with errors. When errors occur in an already activated bundle, the bundle and the requiring bundles are not updated and the current revisions with the existing wires are used. When errors are corrected and the bundle project is built, the bundle and its requiring bundles are updated automatically.

Duplicate bundle projects with the same symbolic name and version is not considered as a compile time error by the build system but it is not possible to activate the workspace if two or more bundles have the same symbolic name and version

For a more complete description of activation and deactivation of bundles with errors see [Bundle Project Errors](../concepts/Bundle%20Project%20Errors.htm).