Update and Refresh Bundles

Contents

[How to 1](#_Toc347699344)

[Refresh 1](#_Toc347699345)

[Restoring State after Refresh 1](#_Toc347699346)

[Refreshing with build Errors 2](#_Toc347699347)

[Refresh Options 2](#_Toc347699348)

[Update 2](#_Toc347699349)

[Restoring State after Update 2](#_Toc347699350)

[Update and Build 2](#_Toc347699351)

[Build Errors 2](#_Toc347699352)

[Automatic Build 2](#_Toc347699353)

[Update and Stale References 2](#_Toc347699354)

[Update Bundles Manually 2](#_Toc347699355)

# How to

A bundle is started when first [activated](Activate%20and%20Deactivate%20Bundles.htm) . When activated, you can then refresh multiple bundles from the Bundle main menu and individual bundles from the context pop-up menus in the Package Explorer, Bundle Detail Page View and the Bundle List Page View. There are also local tool bars in the Details and List pages for refreshing bundles individually.

If the [Update on Build](Setting%20Bundle%20Options.htm#UpdateOnBuild) option is switched on, update becomes an implicit command and automatically executed each time one or more activated bundle projects are built. If the option is off an update command is available from the main and local pop-up menus and tool bars given that there are bundle projects that have not been updated since the last build.

# Refresh

Refresh calculates the requiring dependency closure which is all bundles that directly and indirectly require capabilities from the set of bundles to refresh. The set of calculated bundles are stopped and unresolved forcing a replacement or removal of packages exported by the refreshed bundle(s). When unresolved no wiring will exist among the bundles. This makes the bundle wirings available for garbage collection. Then an attempt is made to resolve the bundles by creating new wirings based on the dependencies between the bundles.

## Restoring State after Refresh

If you refresh a lazy activated bundle that is in state ACTIVE, STARTING/<<LAZY>> or RESOLVED it will return to same state after refresh. Eagerly activated bundles also returns to their pre refresh RESOLVED or ACTIVE state.

## Refreshing with build Errors

If you refresh an ACTIVE or RESOLVED bundle with build errors the current revision and the wirings of the bundle is used. The bundles are updated after a successful build.

## Refresh Options

By default a refresh command is issued on a bundle being updated and its requiring bundles. This will guarantee that older revisions of the bundle are released. See the [Refresh on Update](Setting%20Bundle%20Options.htm#RefreshOnUpdate) option for a detailed description.

# Update

An activated bundle is by default automatically updated after the associated project has been built. The updated version of the bundle is read from the input stream and installed. The bundle is then refreshed if the [Refresh on Update](Setting%20Bundle%20Options.htm#RefreshOnUpdate) option is on and only resolved if not. Lastly the bundle is started if it was ACTIVE before the update command was executed. The new compiled code is loaded and becomes the current active code in the updated bundle at the time the start method of the activator is invoked.

Deactivated bundles does not participate explicit in the update process, but may be activated implicit, and thus updated, due to code changes involving importing packages from deactivated bundles in activated bundles to update.

## Restoring State after Update

After an update operation is complete the bundle is moved to the same state as it was in before the update operation. An exception is a bundle in state RESOLVED, with a lazy activation policy that is refreshed after update. In this case the state after update is STARTING/<<LAZY>>.

## Update and Build

### Build Errors

If an activated bundle project with errors is built, update is skipped and the current revision and wires are used. In other words, an activated bundle always reflects the last successful build as long as the [Update on Build](Setting%20Bundle%20Options.htm#UpdateOnBuild) option is on.

### Automatic Build

When the Build Automatically option is on, projects are built according to default or custom build rules defined for projects. This implies that decisions when to build, which projects to build and kind of build (full or incremental) varies and bundle updates are fully controlled by these rules.

## Update and Stale References

An activated bundle is stopped (if in state ACTIVE) and unresolved before resolved and started again during the update process. This happens every time a project is built and can cause stale references if resources are not released (e.g. in the stop method of the activator of a bundle) when a bundle is stopped or unresolved during the development of a bundle.

## Update Bundles Manually

Bundles are by default updated when built. It is possible to disable this feature by turning the [Update on Build](Setting%20Bundle%20Options.htm#UpdateOnBuild) option off. You can then control the update of bundles manually.

When you update bundles from the main menu all bundles that have been built since last update are updated. When you update a single bundle from one of the local menus or toolbars the partial graph of the bundle is first calculated. See [How to interpret the different Dependency Options](Setting%20Bundle%20Options.htm#HowToInterpretTheDifferentDependencyOpti) for an example of a partial graph. In short, the partial graph is constructed by traversing the bundle dependencies in both directions until it is not possible to extend the graph further. All built bundles are updated and the members of the partial graph of the selected bundle(s) to update are resolved or refreshed if the [Refresh on Update](Setting%20Bundle%20Options.htm#RefreshOnUpdate) is switched on.