Managing Bundle Project Errors

Contents

[Bundle Project Errors 1](#_Toc433871090)

[Build Errors 1](#_Toc433871091)

[Life Cycle Errors 1](#_Toc433871092)

[Error Management 1](#_Toc433871093)

[Sample Use Case 1](#_Toc433871094)

[Install and Activate 1](#_Toc433871095)

[Resolve 1](#_Toc433871096)

[Start and Stop 1](#_Toc433871097)

[Update 1](#_Toc433871098)

[Startup and Shutdown 1](#_Toc433871099)

# Bundle Project Errors

Error handling is centered around the term bundle project which refers to both the project and its associated bundle or plug-in. For a quick overview of activating and deactivating bundle projects see [Activating Bundle Projects and the Workspace Region](Dynamic%20Workspace%20Bundles.docx#ActivatingBundleProjects).

Errors occurs in both bundles and projects and are detected by the build system, the Eclipse core runtime (only plug-ins), the OSGi framework and the Java runtime environment. For a description of the differences between a plug-in and a bundle see [Dynamic Bundles in Eclipse](http://javatime.no/blog/dynamic-bundles-in-eclipse/).

## Build Errors

Errors are divided into build time and life cycle errors. Build errors generated by Eclipse are further divided into Java and Bundle build errors. Java build errors include all errors detected by the java compiler and Bundle build errors, in this context, is directed against errors prohibiting bundle operations from being executed. Bundle build errors is a mix of build errors generated by the Eclipse build system and other build time errors detected by an intermediate layer that sits between the builder and the OSGi life cycle module.

Relevant Bundle build time errors include:

* Build state  
  A project must have a build state before the associated bundle can be installed
* Bundle cycles

Direct and indirect circular references between bundles prevent them from being resolved. See [Bundle Dependencies](Dynamic%20Workspace%20Bundles.docx#DepencencyRules) for a description of bundle cycles.

* External duplicates

It is not possible to resolve and start a workspace bundle that have the same symbolic name as an external jar bundle. Note that the bundle version does not have to be the same.

* Workspace duplicates

For workspace bundles, the bundle names may be equal, but they cannot have the same symbolic name and version.

* Project meta files

Invalid project description files and certain errors in manifest files (e.g. error in the Bundle-SymbolicName header) prevent bundles from being installed. Circular references (Import-Package header) between bundles results in resolve errors while others (e.g. error in the Bundle-Activator header) prevents a bundle from being started.

## Life Cycle Errors

Life cycle errors are divided into modular lifecycle (installed, resolved and uninstalled states) and service lifecycle (starting, active and stopping states) errors detected and reported by the OSGi life cycle module. Bundle life cycle errors are critical errors that prevent bundles from being activated (installed, resolved and started) or cause a bundle operation to fail, while it is possible to activate bundles with Java build time errors. Except for runtime errors, which are service errors, modular errors include, among others, resolve errors, duplicates, cycles, bundle file location access failures, security violations, bundle state and state change errors.

Some errors can be characterized as both as Bundle build and OSGi modular errors depending on the state of a bundle. These kind of overlapping errors are detected at build time, right after build or before a bundle operation is executed if possible. For instance, the Eclipse build system detects circular references between projects while OSGi in principle does the same for bundles, although with a more rigorous set of rules. Another example are duplicates between bundles. Workspace and external (jar bundles) duplicates are not detected by the Eclipse build system while only workspace duplicates are detected by the life cycle module. If for instance a cycle is detected during a build or a duplicate right after a build and the bundle is in state INSTALLED, the bundle will not be resolved and if in state RESOLVED the current revision of the bundle is used when updated. It is then possible to start the bundle, using the current revision, even if the bundle is a duplicate or is part of a cycle.

# Error Management

The error management is controlled by an intermediate layer and a validation state machine backing the life cycle of bundles. The intermediate layer analyze errors produced by the build system, detects additional errors and make decisions on what kind of bundle operations to perform based on changes made to source projects. For instance, if new dependencies are added or removed form a project or when projects are added or removed from the workspace. The validation machine detects transition errors and generates errors in case a bundle operation or transition violates the possible bundle transitions given an initial bundle state. Showing errors states in a state machine clutters up the diagram, but briefly illustrates where the different kind of errors are detected. Note that the state machine diagram is slightly different compared to the one usually shown.



## Sample Use Case

Different use cases that ends up in an error state can be run on the state machine and a given initial status of projects and the state of their bundles. A sample use case may be to update a bundle A after it has been saved and built (the initial state in the diagram) with new dependencies on a deactivated bundle B in state INSTALLED that already has a reference to bundle A (cycle error). Bundle A is initially activated and is in state ACTIVE when adding a dependency (e.g. adding bundle B to its import directive) on bundle B that already is referencing bundle A in its own import directive.

The intermediate layer determines that bundle A should be stopped, updated, resolved and started (to return to its initial ACTIVE state) after a build and bundle B should be activated (installed and resolved) together with bundle A for resolve to succeed. Before updating A, its dependency graph is constructed showing that bundle A require some capabilities from bundle B which at the same time require capabilities from A. This results in a circular reference between A and B and would terminate with a modular resolve error state if executed. In this case the cycle is detected before the bundle operations on A and B are executed and results in a cycle bundle build error. The cycle must in this case be solved by removing one of the import directives in the source projects before bundle A can be updated after a build. A simpler use case is activating a duplicate bundle where the install transition from state UNINSTALLED to state INSTALLED raise a modular install error, causing the bundle project to return (rollback) to state UNINSTALLED. An alternative is to let the intermediate layer detect the error, before even entering the install transition.

# Install and Activate

Errors in meta data files, duplicates, circular references between bundle projects and lack of build state are build errors that prevents bundle projects from being installed and activated. Java build errors does not prevent a bundle project from being installed, but Bundle build errors do. If a bundle build error occurs in a deactivated workspace all bundles remains in state UNINSTALLED.

# Resolve

In the life cycle of bundles, they are resolved after they are installed and after they are updated and refreshed when in state RESOLVED and ACTIVE (and state STARTING/STOPPING). In short, when a bundle is resolved, all Java classes that the bundle needs, located within the bundle and in all bundles that the bundle to resolve depends on, are available.

The following errors are detected and reported before trying to resolve a bundle:

* Circular references (cycles) result in resolve errors and are detected before bundles are resolved. See [Dependency Rules](Dynamic%20Workspace%20Bundles.docx#DepencencyRules) for an explanation of bundle cycles.
* Dependencies are calculated where bundles providing capabilities to bundles to resolve are automatically included in the resolve process to avoid missing references. Errors are reported if a dependency tree could not be calculated before trying to resolve the bundles.

If the resolver detects any other modular errors, they are reported and the bundles to resolve are deactivated and moved to state INSTALL in an activated workspace and state UNINSTALLED in an deactivated workspace.

# Start and Stop

It is possible to activate a bundle with Java build errors even if this kind of errors may generate a runtime error when the bundle is started. If the runtime flow does not execute code where the build errors are located, no runtime errors occurs due to the build errors. An option is provided to not allow activation (and thus starting a bundle) of bundle projects with Java build errors.

Any service exceptions thrown in start and stop methods are caught and reported. After a service runtime error, the bundle returns to state RESOLVED. It is possible from the bundle menus to terminate a bundle while it is running within its start and stop methods. If stopped manually a service exception is then generated stating that the start or stop operation was incomplete.

# Update

A bundle is usually updated after a build at any activation level (installed, resolved and started). A bundle, together with its dependent bundles, is assigned a unique revision when resolved. Trying to update a resolved or started bundle with a bundle – Eclipse build or life cycle error - error is rejected and the current revision of the involved bundles are used. In such cases the state of the bundle projects is a project in an erroneous state and a running bundle executing code from the last error free build. Bundles with Java build errors may be updated.

# Startup and Shutdown

At shutdown all bundles are uninstalled and refreshed. If there were Eclipse build or modular bundle errors at shut down all bundles are deactivated. If the IDE crashes, the bundles are either deactivated at startup or installed and resolved (not started) depending on the state of the workspace. Some projects may be missing build state and need to be refreshed (e.g. not in sync with the file system) after a crash. It is not possible to activate the workspace if there are bundles with Eclipse build errors. The situation at startup is similar to activating the workspace; - That is installing all workspace bundles and resolving and starting the activated bundles.