Identifying Package Dependencies

In a model, packages can be associated with other packages or model elements in a project scope or outside the project scope. In other words, packages can have relations with external elements in used projects, shared libraries, profiles, etc. An element depends on the used project when at least one of its metaproperties reference elements from at least one share of that used project. In this case, the element depends on that used project.

To avoid any violations on editing a package, especially on removing or exporting a package to another project, it is always good to know all of the package dependencies. Dependency analysis allows users to detect cyclic dependencies that impede reusing the package in the current or other projects because it can cause an infinite recursion. This analysis manages the impact of changes to other related elements.

In the modeling tool, the Dependency Checker allows you to analyze package dependencies in the whole project scope or analyze dependencies between the selected package/model and shared packages. The dependency severity level status shows how a package depends on a particular element. There are three severity levels:

- **Info**. The lowest severity.
- **Warning**. Conflicts that require your attention are displayed.
- **Error**. The highest severity.

> **Severity levels**
>
> While checking dependencies, conflicts of the selected severity (and higher) are displayed in the **Package Dependencies** panel.

In the **Project Options** dialog, you can define the default values for checking dependencies. You can specify the following:

- The **Check for Cyclic Dependencies among Used Projects** option. Click to select *true* if you want the cyclic dependencies to be checked while exporting or sharing the packages.
- The **Dependency Checker Severity Level** option. Specify the severity level. The dependencies of the specified severity (and higher) will be displayed.
- The **Ignore Standard/System Profiles** option. Click to select *true* if you want to exclude the standard/system profiles from checking for cyclic dependencies in used projects.

Each time you start the dependency checking, you can change the default settings.

In the following procedures, learn how to work with the Dependency Checker:

- How to set default values for the Dependency Checker
- How to analyze package dependencies of the whole project or between the selected package /model and shared packages
- How to analyze package dependencies while exporting packages
- How to analyze package dependencies while sharing packages

To set default values for the Dependency Checker

---

1. On the main menu, click **Options** > **Project**. The **Project Options** dialog opens.
2. Expand the **General** option in the option group list.
3. Click the **Dependency Checker** option. The list of options opens.
4. In the Dependency Checker options list, set the desired values and click **OK** when you are done.

> **Description of an option**
>
> Each option has a short description. Click the option to see the description in the description area of the dialog.

To analyze package dependencies of the whole project or between the selected package/model and shared packages

---

1. Choose one of the following:
   - To analyze package dependencies of the whole project, on the main menu, click **Analyze** > **Dependency Checker**.
   - To analyze dependencies between the selected package/model and shared packages, right-click the package or model in the Model Browser or on the diagram pane, point to **Tools** and then click **Dependency Checker**.

Related pages

- Package Dependencies panel
- Analyzing dependencies among elements
- Unresolved dependencies

2. The **Dependency Checker** dialog opens. Select the **Minimal Severity** from the list, check ignore or not standard and system profiles while checking and click **OK** to execute the dependency checker. If some dependencies are found, they are displayed in the **Package Dependencies** panel.

To identify package dependencies while exporting packages

1. From the package/model shortcut menu, choose **Project Usages** > **Export Packages to New Project**. The **Export Packages to New Project** dialog appears.
2. Select the packages you want to export. For more information, see Exporting packages to new projects.
3. Click **OK**.
4. In the question dialog, click **Yes**.

Notes

- Dependencies between the selected package/model and shared packages will be analyzed and shown in the opened **Package Dependency** panel.
- If you want to discover *cyclic dependencies*, select the **Check for cyclic dependencies among used projects** check box. This enables for the dependencies which have **Error** and **Warning** severity levels to be displayed.

To identify package dependencies while sharing packages

1. From the package/model shortcut menu, choose **Project Usages** > **Share Packages**. The **Shared Packages** dialog appears.
2. Select the packages you want to share. For more information, see Sharing project data.
3. Click **OK**.
4. In the question dialog, click **Yes**.

Notes

- Dependencies between the selected package/model and shared packages will be analyzed and shown in the opened **Package Dependency** panel.
- If you want to discover *cyclic dependencies*, select the **Check for cyclic dependencies among used projects** check box. This enables for the dependencies which have **Error** and **Warning** severity levels to be displayed.

Cyclic dependency

If there is a chain of dependencies such that A M(1), M(1) M(2), M(2) M(3), ..., M(X) M(A), where:

- A is an element from the used project M(A)
- M(1..X) are other used projects
- A M(x) is element A dependency on used project M(x)
- M(y) M(x) is a dependency of at least one element in used project M(y) on used project M(x),

then this chain is called a cyclic dependency and every atomic dependency in this chain is considered as part of cyclic dependency.