[Lookup Rules](http://book.xogeny.com/components/packages/lookup/)

[Modelica Standard Library](http://book.xogeny.com/components/packages/msl/)



[http://book.xogeny.com/_static/TitleHeading.png](http://book.xogeny.com/)

Release: v0.3.0-12-g226adba-Early Access

[Packages](http://book.xogeny.com/components/packages/)

**Importing**

*Table Of Contents*

* [Packages](http://book.xogeny.com/components/packages/)
  + [Importing](http://book.xogeny.com/components/packages/importing/)

*Importing*

As we saw previously, there are basically three forms of importing. In all cases, the import statement creates an “alias” within the definition that refers to a type defined elsewhere.

The first form simply imports a definition by its fully qualified name, e.g.:

**import** **Modelica.SIunits.Temperature**;

The result of such an import is that references to the name Temperature are mapped to the fully qualified name Modelica.SIunits.Temperature. In other words, the alias introduced by the import statement is Temperature and it maps to the definition found at Modelica.SIunits.Temperature. With this form of import, the name of the alias always matches the last element in the name of what is being imported (*e.g.,* Temperature).

In some cases, we want the alias that is introduced to be different from the last element of the imported name. In this case, we can explicitly introduce an alternative name for the alias, *e.g.,*

**import** DegK = **Modelica.SIunits.Temperature**; *// Kelvin*

After such an import, we can use the alias DegK to refer to Modelica.SIunits.Temperature. Providing alternative names avoids name collisions or simply makes the model more readable.

Finally, it is possible to import all definitions within a package into the current scope. This is done with a wildcard import. For example, to import all the definitions in the Modelica.SIunits package, we would use the following import statement:

**import** **Modelica.SIunits**.\*;

Such an import would create as many aliases as there are definitions in Modelica.SIunits. The only option available is for each alias to be named the same as the definition in the imported package (*i.e.,* it isn’t possible to assign an alternative name for the alias).

**Wildcards considered harmful**

These types of wildcard imports are dangerous because, as mentioned, there is no option to rename a type. As a consequence, two or more wildcard imports in the same model could create name clashes. Furthermore, explicit imports (or fully qualified types) make it very easy to backtrack and locate the definition associated with a given type. Wildcards make this very difficult because it is not clear what types are imported from what packages.