SIMULATION X®
Powered by ITI

User Information

Using custom model elements "External Types" in SimulationX 3.7

When do the following explanations apply?

User-defined model libraries which were created under *ExternalTypes* in *SimulationX 3.5* or earlier need to be imported in order for models containing types from these libraries to work in *SimulationX* 3.7. A dedicated dialog guides the user through the automated process and only requires the user to specify a directory and a name.

Custom libraries and types which were created as Modelica packages (i. e. not under *ExternalTypes*) are not affected. They do not require any import procedure.

What is a type?

Types are elements from libraries which can be accessed via the library bar. This can be model elements, functions or fluids, for instance.

Background

Up to version 3.5, it was not only possible to use the standard *SimulationX* libraries, but also to organize custom elements under *ExternalTypes* in the library bar. Since *SimulationX 3.0*, users can generate types and libraries as Modelica packages in line with the Modelica standard. Starting with version 3.6, custom types and libraries are now solely managed in line with the Modelica standard as part of the ongoing development of *SimulationX*. The folder *ExternalTypes* has been removed.

For further information about external types, please refer to the user manual, section 8.1.1.

What is a Modelica package?

A Modelica package is an organization structure in the Modelica language. It can comprise functions and model elements as well as Modelica packages, for example. In the library bar, Modelica packages are treated like other libraries as a structural unit. It is comparable to a folder in the Windows Explorer.

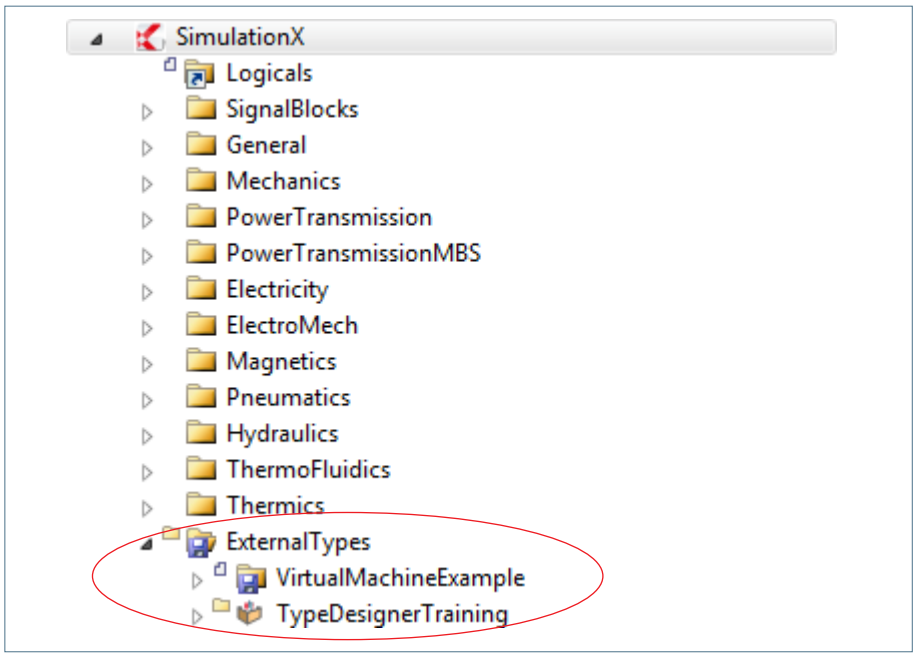
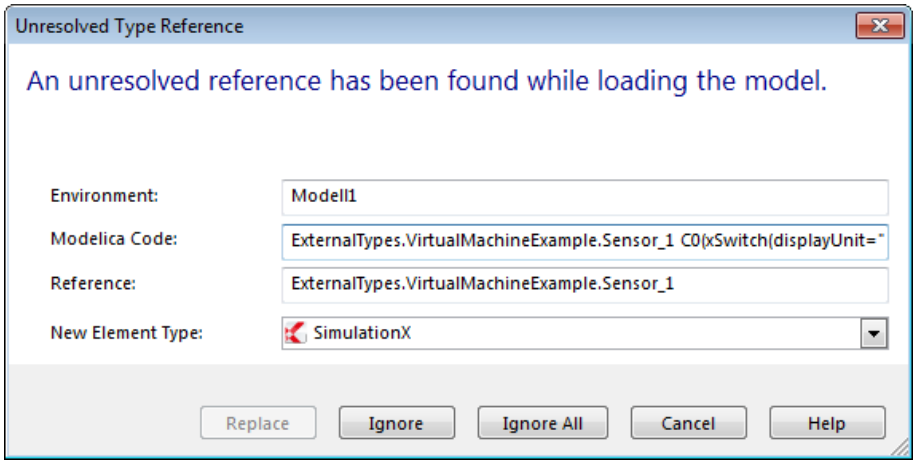


Fig. 1: Library tree in *SimulationX 3.5*

What happens when I do not import External Types?

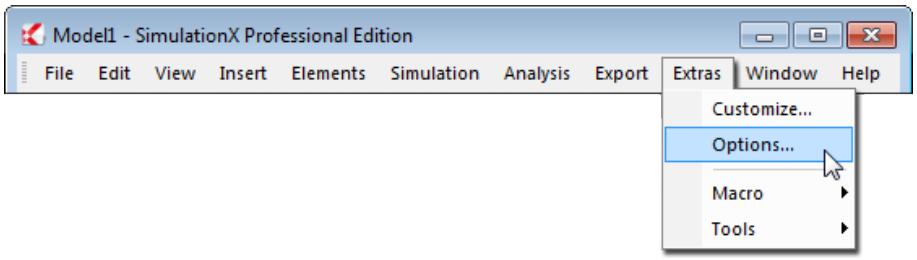
If models are loaded in *SimulationX* 3.7 which include user-defined types that were organized under *ExternalTypes*, they cannot be found without prior import. Loading such a model causes a message to pop up stating "Unresolved Type Reference". In this case, the model must be closed and reloaded after a successful import.



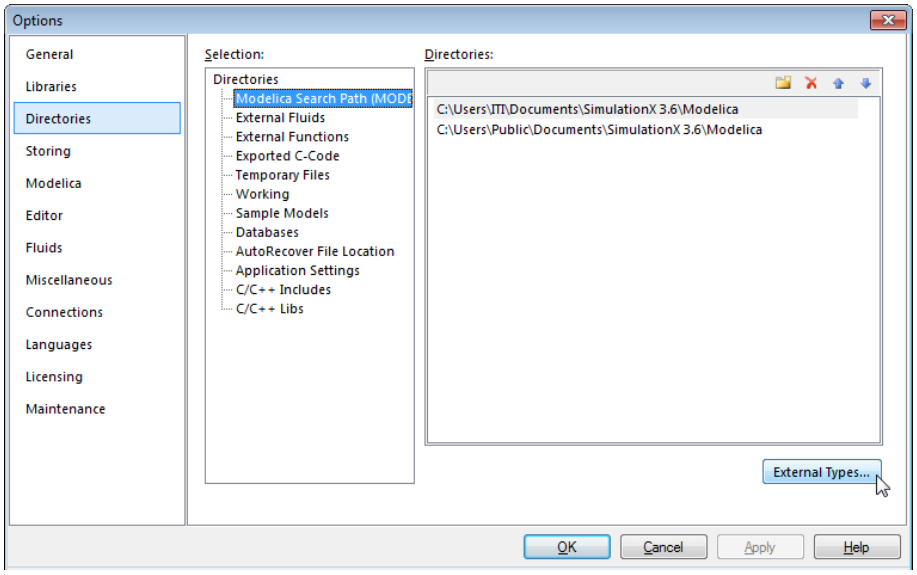
Additional information:
If all *ExternalTypes* are located in the same directory, the import only needs to be executed once after the installation of *SimulationX* 3.7. If there are different directories for user-defined elements that are organized under *ExternalTypes*, the import procedure needs to be repeated accordingly. When loading the model that contains user-defined types from *ExternalTypes*, *SimulationX* reroutes all corresponding references automatically to the newly created Modelica package.

I. Importing External Types

1 Go to *Extras* → *Options* in the menu bar

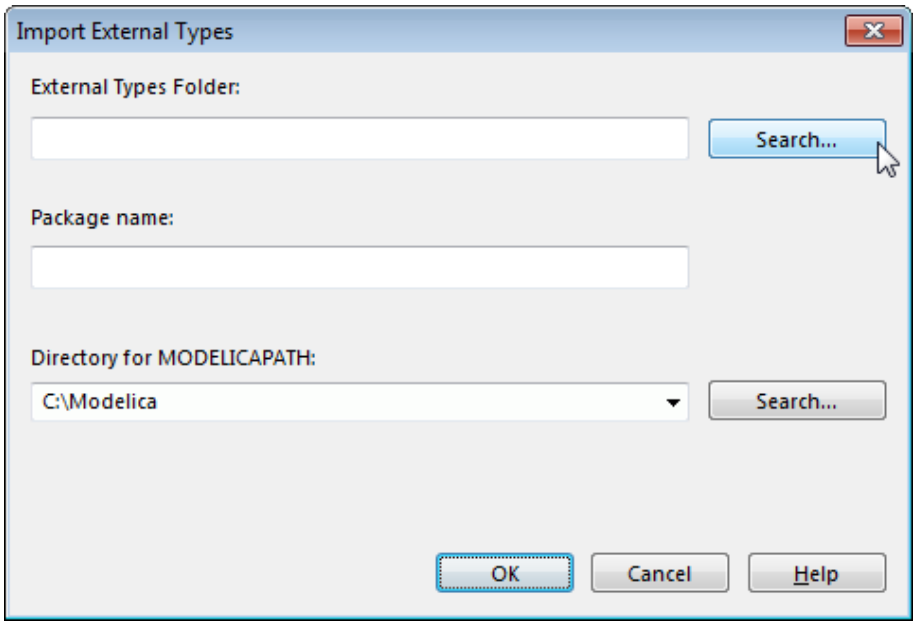


2 Under *Directories* → *Modelica Search Path*, click on *External Types*



Additional information:
SimulationX 3.7 creates internal references for types from the imported *ExternalTypes* package. References from old models starting with *ExternalTypes* can thus still be resolved after importing the corresponding directory. For rare cases, when this is not sufficient, as for relative references between former external types, there is a dialog for interactive type correction that can be used to make the necessary adjustments (see section II).

3 Click the *Search* button to select a directory that contains user-defined types which were created under *External Types* in *SimulationX 3.5* or earlier.



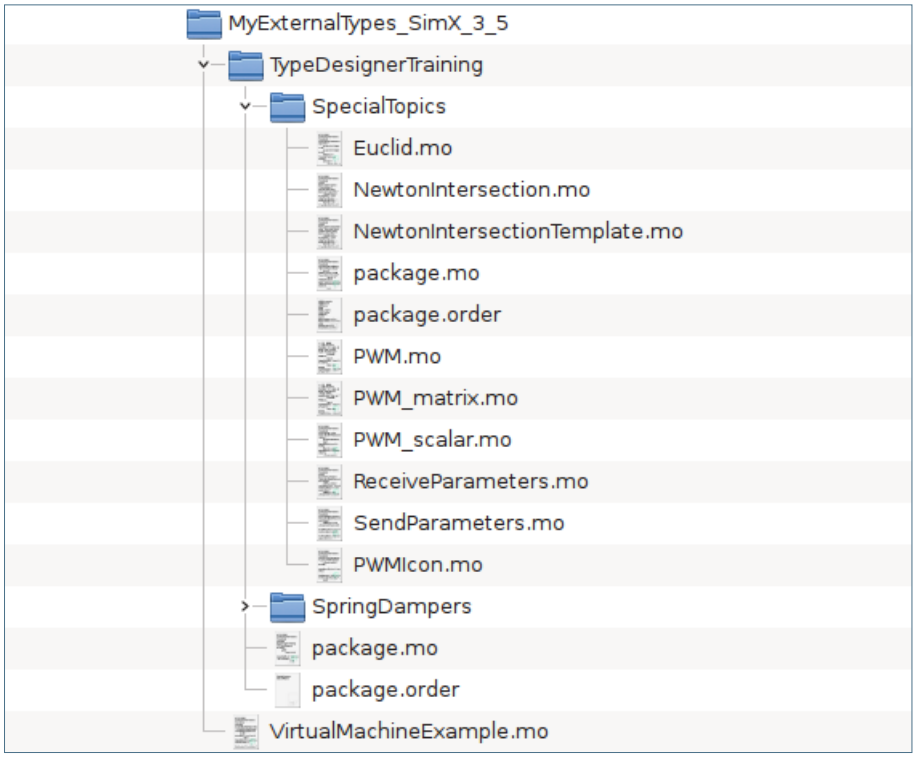
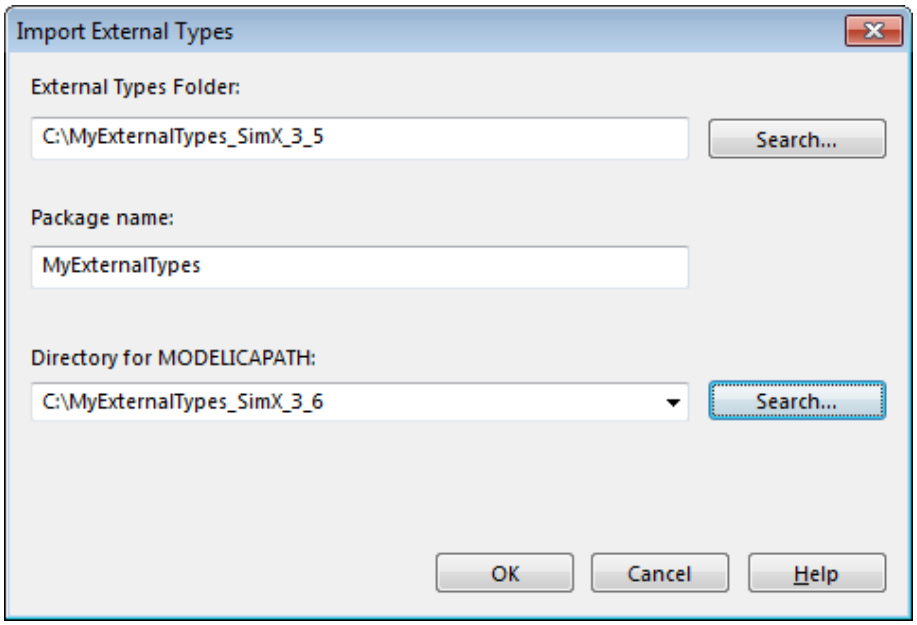


Fig. 2: Example of a directory with user-defined types organized under *ExternalTypes*

4 Enter a name for the package at *Package name* under which the *External Types* from the imported folder should be managed (e.g. My*ExternalTypes*).

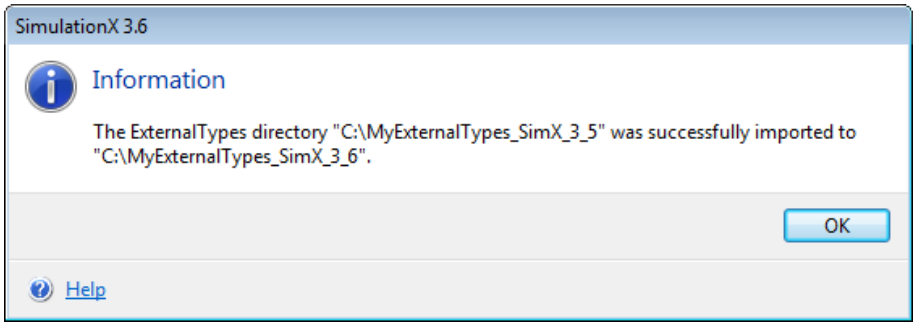
Please note!

1. The package name must be different from the name of the directory from which the *External Types* are imported
2. The package name must be different from existing libraries
3. The name "*ExternalTypes*" is not permitted



5 Under *Directory for MODELICAPATH*, select an existing Modelica directory or choose a storage location where *SimulationX* may create a new directory (e.g. My*ExternalTypes*_SimX_3_6).

6 Click *OK* and close the dialog. *SimulationX* takes care of all further steps in the background automatically.



The import of External Types is now complete. All models from *SimulationX 3.5* and earlier versions with already imported user-defined types can now be loaded in *SimulationX* 3.7. References to imported types are recognized by *SimulationX* 3.7 automatically.

Please note!

When saving the model, *SimulationX* replaces all references to *ExternalTypes* in the Modelica code that includes the references to the imported types.

Once saved, such models cannot necessarily be loaded in older versions of *SimulationX* any more. (Backwards compatibility between different *SimulationX* versions cannot be guaranteed.)

Additional information:

The conversion of types that were formerly organized under *ExternalTypes* into Modelica packages is based on a copy. The original files selected for conversion remain intact at their storage location. This allows *SimulationX 3.5* or previous versions to still access these types.

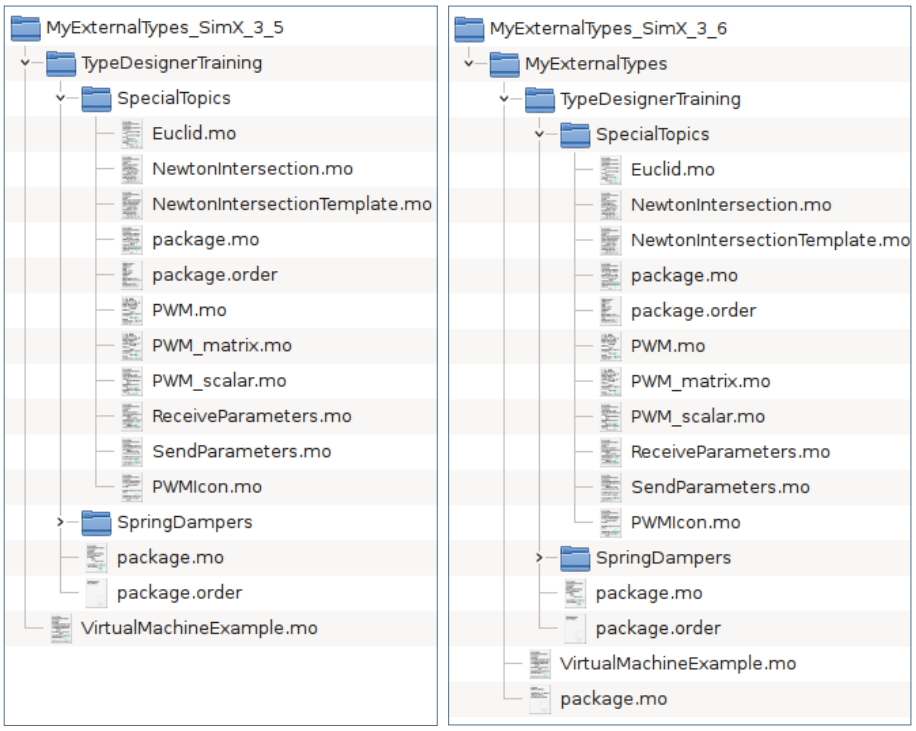


Fig. 3: Comparison between the local or network folder structures of *ExternalTypes* created in *SimulationX 3.5* or earlier (on the left) and the imported directory for these user-defined types in *SimulationX* 3.7 (on the right).

Additional information:

During the import of external types, *SimulationX* creates a directory in the specified Modelica search path (under the given name of the package), and all files and folders from the former External Types folder are copied there.

Additionally, a file *package.mo* is generated to comply with the Modelica requirements for the directory structure. This file has an annotation telling *SimulationX* that it includes types that used to be organized under *ExternalTypes*.

Important: The original directory for ExternalTypes is not changed during import!

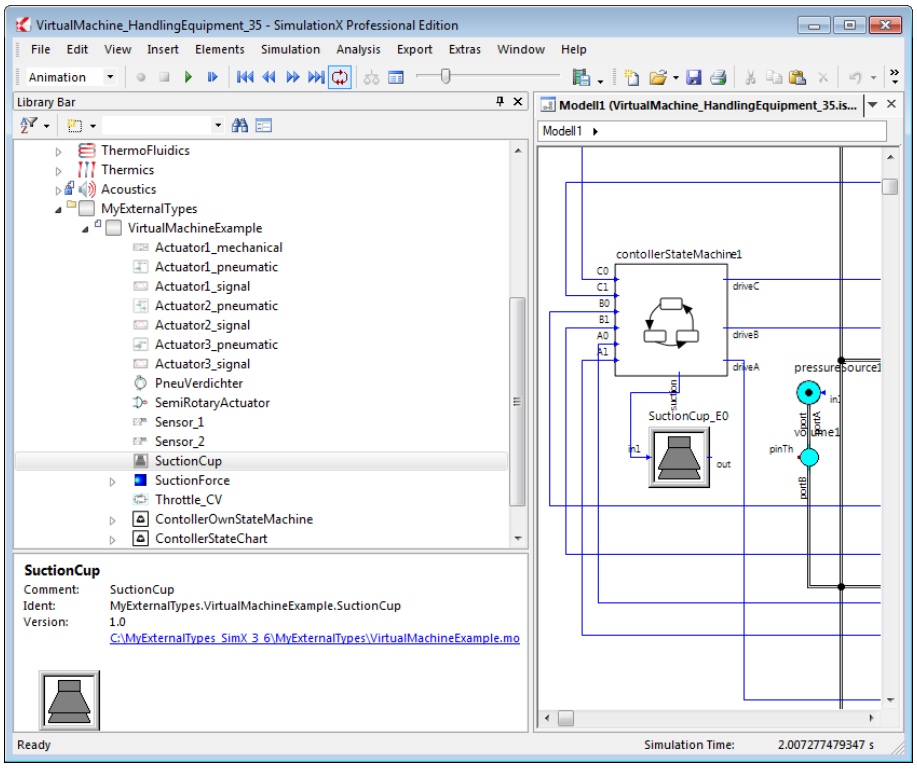


Fig. 4: Tree view in the library bar of *SimulationX* 3.7 with imported user-defined types

Additional information:

The conversion of types that were formerly organized under *ExternalTypes* into Modelica packages is based on a copy. The original files selected for conversion remain intact at their storage location. This allows *SimulationX 3.5* or previous versions to still access these types.

In the unlikely event that references cannot be resolved regardless of a prior import of External Types, a dialog is shown which is explained in section II.

II. Interactive Type Correction when loading models

Additional information:
SimulationX is not able to resolve a reference if there is no matching type for the Modelica ident.

On rare occasions, it is possible that *SimulationX* cannot resolve type references when loading a model. This may occur, for example, if the file structure of user-defined types was changed, or for particular constellations during the import of *External Types*.

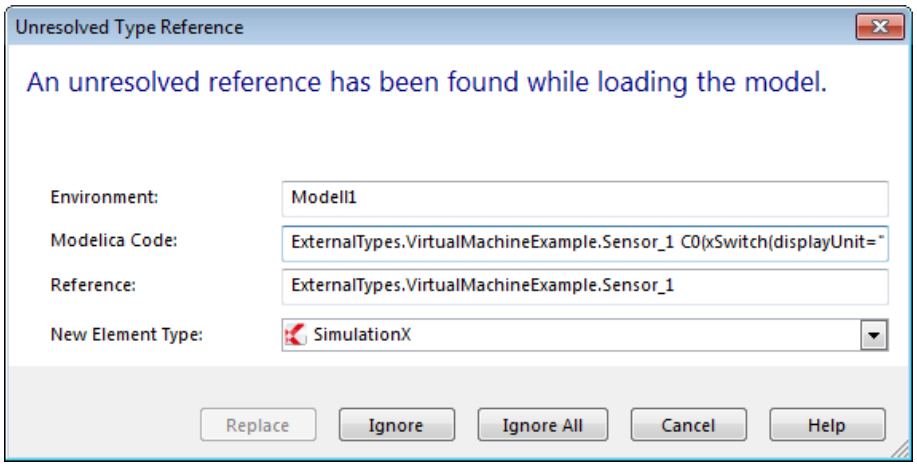
In such a case, *SimulationX* 3.7 shows a dialog for *interactive Type Correction* whenever a type reference cannot be resolved when loading a model. This dialog helps to select an alternative type or point *SimulationX* to the directory that holds the missing type.

The next section describes how to use this dialog. For further details, please refer to the user manual, section 8.1.2.

Please note!
In contrast to the *Import of External Types*, the *correction of unresolved references* only applies to the selected model.

Exception: The corrected reference is located in the library bar under a type that is used by the active model. Saving the corrected type applies all changes to each model that uses this particular type.

1 The following dialog is shown when *SimulationX* loads a model with missing references:



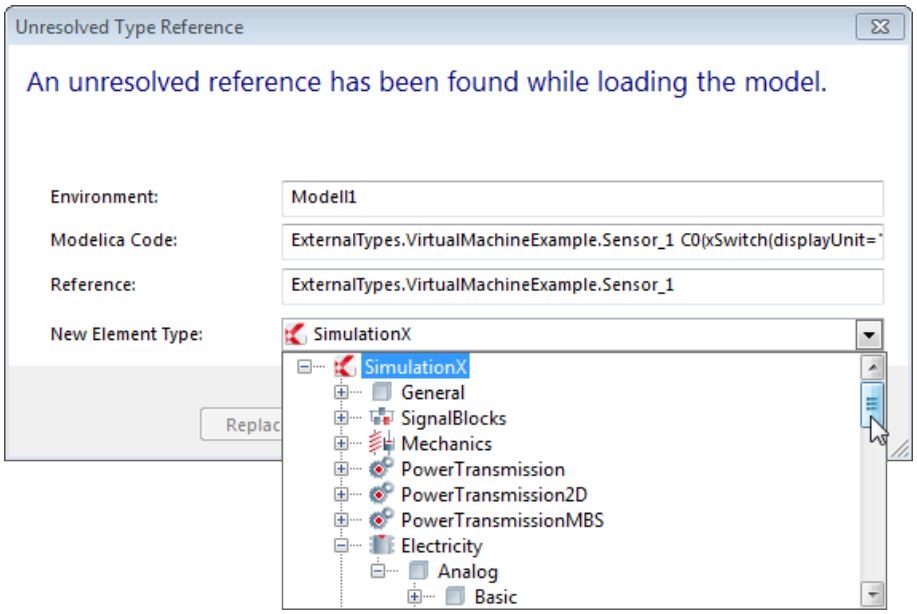
The fields *Environment* and *Modelica Code* indicate at which point the reference cannot be resolved. *Environment* states the user-defined library, compound or model in which the reference is missing. *Modelica Code* provides a snippet of the Modelica code where the corresponding reference is missing.

Reference specifies the *Ident* of the missing type.

What is an ident?
The *ident* specifies a type or component. In order to avoid confusion (*Ground*, for instance, can be used for several types in different libraries), the ident includes both the name and information about its classification within the structure.

Example:
A type named *Ground* can be found both in the *Magnetics* library and the *Electricity* library. Both types have characteristics based on their corresponding domains. The ident *Magnetics.Basic.Ground* makes this type *Ground* distinguishable from the other one in the Electricity library (Ident: Electricity.Analog.Basic.Ground).

2 In the field *New Element Type*, it is possible to specify a type to be used instead of the missing reference.



Changes are applied by clicking *Replace*. The missing reference is replaced by the selected type which is then used when the model is loaded. Saving the model preserves all changes for later use.

ITI Global ITI Headquarters • Schweriner Straße 1 • 01067 Dresden/Germany
www.itisim.com • sales@itisim.com • T + 49 (0) 351.260 50 - 0 • F + 49 (0) 351.260 50 - 155
SimulationX For your local representative please visit: www.simulationx.com/global