Dymola

Dynamic Modeling Laboratory

Dymola Release Notes

Dymola 2026x

The information in this document is subject to change without notice.

Document version: 1

Dassault Systèmes AB
Ideon Gateway
Scheelevägen 27 – Floor 9
SE-223 63 Lund
Sweden

Support: https://www.3ds.com/support
URL: https://www.dymola.com/
Phone: +46 46 270 67 00

Contents

1 Important notes on Dymola

Installation on Windows

To translate models on Windows, you must also install a supported compiler. The compiler is not distributed with Dymola. Note that administrator privileges are required for installation. Three types of compilers are supported on Windows in Dymola 2026x:

Microsoft Visual Studio C++

This is the recommended compiler for professional users. Both free and full compiler versions are supported. Refer to section "Compilers" on page 104 for more information. **Notes**:

- From Dymola 2024 Refresh 1, Visual Studio C++ compilers older than version 2015 are no longer supported:
    - From Dymola 2024x Refresh 1, Visual Studio 2012 is not supported anymore.
    - From Dymola 2022x, Visual Studio 2013 is not supported anymore. (Visual Studio 2012 was however still supported until Dymola 2024x, due to the logistics of changing the oldest supported version)

Intel

Important. The support for Intel compilers is discontinued from the previous Dymola 2022 release.

MinGW GCC

Dymola 2026x has limited support for the MinGW GCC compiler, 32-bit and 64-bit. For more information about MinGW GCC, see section "Compilers" on page 104, the section about MinGW GCC compiler. (Note that the support for MinGW GCC 32-bit compiler is deprecated in this version (Dymola 2026x), to be removed in Dymola 2026x Refresh 1. 64-bit GCC will still be supported.)

WSL GCC (Linux cross-compiler)

Dymola 2026x has support for the WSL (Windows Subsystem for Linux) GCC compiler, 64-bit. For more information about WLS GCC, see section "Compilers" on page 104, the section about WSL GCC compiler.

Clang compiler

If you first select to use Visual Studio 2019, Visual Studio 2022, or WSL GCC as compiler, you can then select to use Clang as code generator instead of native Visual Studio/WSL GCC.

Installation on Linux

To translate models, Linux relies on a GCC compiler, which is usually part of the Linux distribution. Refer to section “Supported Linux versions and compilers” on page 108 for more information. Note that you can use Clang as code generator.

2 About this booklet

This booklet covers Dymola 2026x. The disposition is similar to the one in Dymola User Manuals; the same main headings are being used (except for, e.g., Libraries and Documentation).

3 Dymola 2026x

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2026x. In particular, Dymola 2026x provides:

- New library: Sustainable Supply Systems Library (see below)
- Support for the new Modelica Standard Library version 4.1.0 (page 10)
- Table lookup for MSL or Dymola (page 14)
- Modelica Text editor: Options for automatic indentation while typing (page 23)
- Checking bus signal sets for missing signal sources (page 24)
- Model metadata editor improvements (page 25)
- Customization of the quick access bar and the ribbon (page 48)
- List of recommended libraries for on-demand download and installation (page 42)
- Improved support of SSP:
    - SSP import (starting on page 95), containing:
        - Purging resources
        - Unpacking in a temporary directory
        - Providing placement when needed
        - Improved handling of inlined components
    - SSP export:
        - Exporting protected items (page 96)
        - Option to save file without an SSP directory (page 96)
        - Updated SSP export of FMU components (page 96)
        - Exporting an SSV instead of SSP in some cases (page 96)
        - Specifying the SSP version to 2.0 when exporting (*beta feature*) page (72)
        - Option to export SSP parameter bindings as external files (*beta feature*) (page 72)
- Improved support of FMI:
    - FMI export:
        - Shorter default model identifiers (page 96)
        - All solvers supported for analyzing numeric integration (page 97)

    - FMI import:
        - Delayed unpacking of FMU (page 98)
        - Option to use a standardized package of FMI functions when importing FMUs (*beta feature*) (page 69)
    - Continued implementation of FMI 3 support for terminals and icons (*beta feature*) (page 66)
    - Support for terminals and icons for FMI Layered Standard for Structured Data (*beta feature*) (page 68)
- Improved support of eFMI:
    - Support for embedded code generation for neural networks (page 98)
    - New GALEC built-in functions implementation of Software Production Engineering (page 98)
    - Added Arduino support (page 98)
    - Support for tunable parameters in MATLAB/Simulink exporter (page 99)
    - Checking eFMUs for eFMI Standard compliance with eFMPy (page 99)
    - Preserving multi-dimensional equations of source models (page 99)
- Reading Widows registry keys (page 58)
- Changed color theme for dark mode (page 60)
- Improved support for the 3DEXPERIENCE app “Design with Dymola”:
    - For a model parameter, specifying a file from the 3DEXPERIENCE Platform (page 80)
    - Importing an FMU, an SSP or an SSV from the 3DEXPERIENCE Platform (page 85)

3.1.2 New and updated libraries

New libraries

- Sustainable Supply Systems Library

For more information, see “New libraries” starting on page 101.

Updated libraries

The following libraries have been updated:

Note. The below list build on what is displayed when using **File > Libraries**. The name of the package in the package browser may be different, and it may be that more than one package is opened, due to help packages, underlying library combinations etc.

- Aviation Systems Library, version 1.7.0
- Battery Library, version 2.8.2
- Brushless DC Drives Library, version 1.4.4
- ClaRa DCS Library, version 1.8.0
- ClaRa Grid Library, version 1.8.0

- ClaRa Plus Library, version 1.8.0
- Claytex Library, version 2025.2
- Claytex Fluid Library, version 2025.2
- Cooling Library, version 1.5.5
- DataFiles, version 1.1.2
- Dassault Systemes Library, version 1.15.0
- Design Library, version 1.2.4
- Dymola Commands Library, version 1.20
- Dymola Embedded Library, version 1.0.6
- Dymola Models Library, version 1.11.0
- eFMI Library, version 1.0.2
- eFMI_TestCases, version 1.0.2
- Electric Power Systems Library, version 1.7.1
- Electrified Powertrains Library (ETPL), version 1.12.0
- External Interfaces Library, version 1.7
- Flight Dynamics Library, version 1.0.5
- Fluid Dynamics Library, version 2.20.0
- Fluid Power Library, version 2025.2
- FTire Interface Library, version 1.3.3
- Human Comfort Library, version 2.20.0
- HVAC (Heating, Ventilation, and Air Conditioning) Library, version 3.5.0
- Hydrogen Library, version 1.4.3
- Model Management Library, version 1.4.2
- Modelica Standard Library, version 4.1.0
- Modelica_LinearSystems2, version 3.0.1
- Modelica_StateGraph2, version 2.1.1
- ModelicaServices, version 4.1.0
- Multiflash Media Library, see Thermodynamics Connector Library
- Optimization Library, version 2.2.8
- Plot3D, version 1.1.5
- Pneumatic Systems Library, version 1.7.3
- Process Modeling Library, version 1.4.0. **Note**. This library is currently not supported on Linux.
- SDF, version 0.4.6
- Testing Library, version 1.11.0
- TIL Suite 2025.2:
    - TIL 2025.2
    - TIL Automotive 2025.2

    - TIL Heat Storage 2025.2
    - TIL PCM Storages 2025.2
    - TIL NTU 2025.2
    - TIL Hydrogen Energy Systems 2025.2
    - TIL Adsorption 2025.2
- Thermodynamics Connector Library (previously named Multiflash Media Library), version 1.3.1. **Note**. This library is currently not supported on Linux.
- UserInteraction, version 0,71
- Vehicle Interfaces, version 2.0.2
- VeSyMA (Vehicle Systems Modeling and Analysis) Library, version 2025.2
- VeSyMA - Engines Library, version 2025.2
- VeSyMA - Powertrain Library, version 2025.2
- VeSyMA - Suspensions Library, version 2025.2
- VeSyMA2ETPL Library, version 2025.2
- Visa2Base, version 1.19
- Visa2Paper, version 1.19
- Visa2Steam, version 1.19
- Wind Power Library, version 1.1.7

For more information about the updated libraries, please see the Release Notes section in the documentation for each library, respectively.

3.2 Developing a model

3.2.1 New version of Modelica Standard Library

General

Modelica Standard Library (MSL) version 4.1.0 is included in this Dymola 2026x distribution.

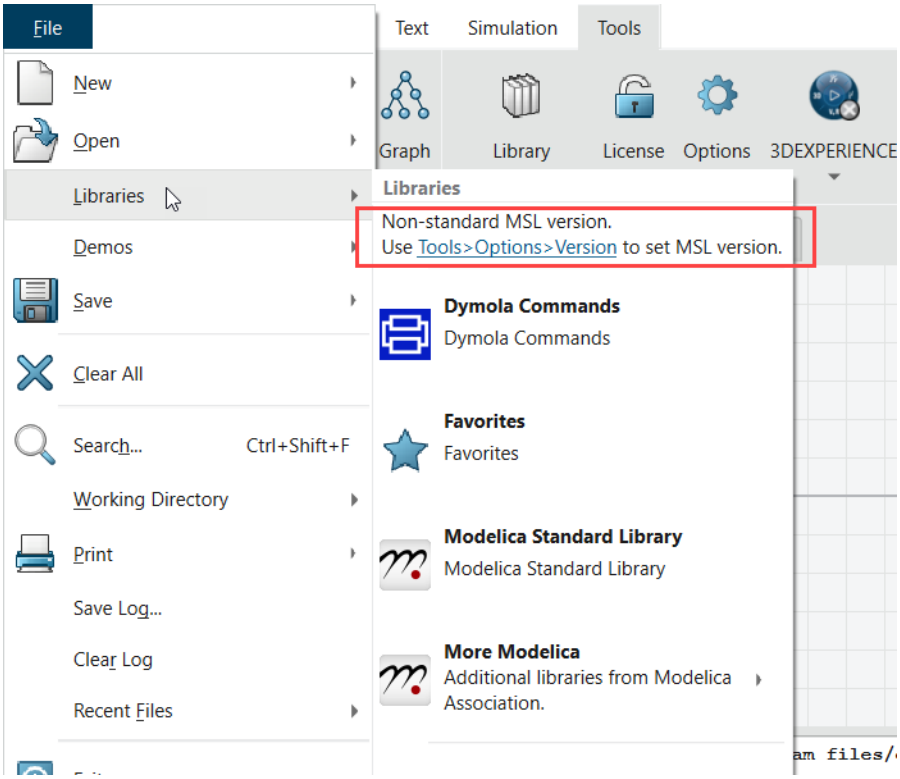
By default, Dymola will start with MSL 4.1.0 when opening Dymola 2026x.

All libraries in the Dymola distribution support MSL 4.1.0.

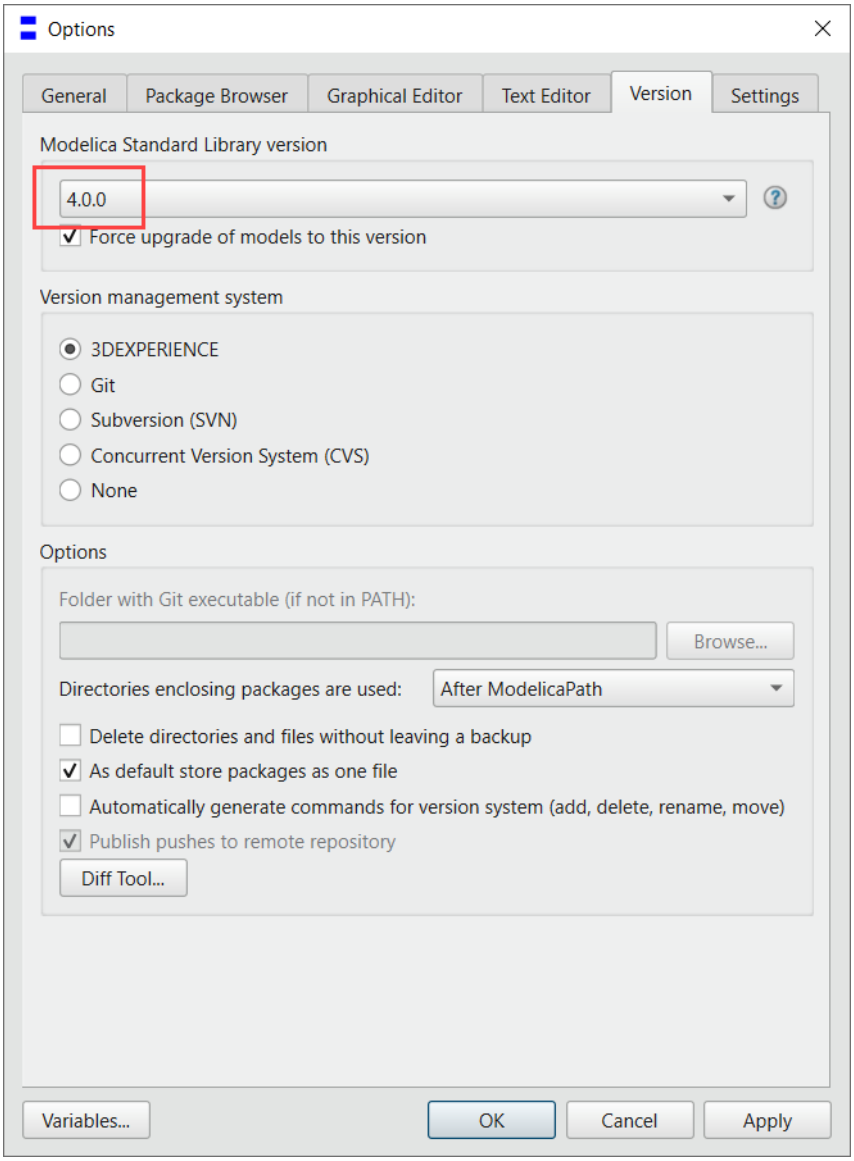
Improved information when not using the default version of Modelica Standard Library

If you are using the library Modelica Standard Library (MSL) with a version lower than the default version, currently 4.1.0, you cannot see all libraries when using the command **File > Libraries**, since the default MSL version is demanded for a number of libraries.

From Dymola 2026x, in such a case you get information about this when using the command **File > Libraries**:



Clicking the link, you may see:



In this case, you have to select 4.1.0 instead of the present 4.0.0. It is usually a good idea to also activate **Force upgrade of models to this version**.

Handling MSL 4.1.0 rotational friction models using tables, for example, clutches

Some rotational friction models, for example the clutch and the one-way clutch, use Modelica tables for the parameters. Especially for analytic Jacobian and Rosenbrock solvers, such models have non-differentiable functions inside conditions that cause a failure to differentiate.

This can be avoided by evaluating the specific conditions (assuming the parameter has an appropriate value). That functionality is controlled by the new flag:

```
Advanced.Translation.SmartConditionalJacobian
```

The possible values of this flag are:

- 0 – **Disable**. (The new functionality is not activated – corresponds to earlier versions.)
- 1 – **When Jacobian evaluation fails**, activate the new functionality.
- 2 – **Before evaluating Jacobian**, activate the new functionality (this is the default value of the flag).

Note that it is normally recommended to set the flag `Evaluate = true` when using Rosenbrock solvers in a realtime setting – this eliminates the need for using the new flag.

Handling error corrections that may effect old models

In the library ModelicaServices 4.1.0, errors in machine constants have been corrected, this can however effect some old models. In such cases, you may get warnings for `Modelica.Constants`, but you may just get odd results – or errors. To handle this, you can instead use the previous values of the constants by setting the flag:

```
Advanced.Modelica.CompatibilityConstants = true
```

(The default value of the flag is `false`.)

Using MSL 4.1.0 without updating the models

For certain cases, you don´t want to upgrade your models to fully use MSL 4.1.0, you want to use MSL 4.1.0 instead of MSL 4.0.0 but without upgrading your models. If the differences between MSL 4.1.0 and MSL 4.0.0 are not effecting the package or library where you use it, this is possible by setting the flag:

```
Advanced.Editor.DelayConversionIfNotNeeded = true
```

(The flag is by default `false`.) The flag is general for using an older version of a library without updating the models, and was available in earlier Dymola versions. The flag is not saved between sessions.

Notes:

- The library or package will be made read-only, to prevent changing it. You can use the package context menu entry **Upgrade** to fully upgrade it, which allows editing.
- If the changes between the old version of the library and the new version *do* affect the package or library where you use it, you will get a warning when loading it, and the loaded library will be fully upgraded anyway. Important: This is not the case for MSL 4.1.0, it can always be used instead of MSL 4.0.0 without full upgrade.
- If the process is successful (as always for MSL 4.1.0), it is silently implemented in Dymola 2026x. If you want to have a message about the success (that you must acknowledge), you can set the flag `Advanced.Editor.PromptDelayedConversion = true`. (The flag is by default `false`. The flag is saved between sessions.)

Installing and working with MSL 4.0.0 instead of MSL 4.1.0

If you are working with MSL 4.1.0 and need to work with an older Modelica model that has not been converted to the new MSL version, you can yourself install MSL 4.0.0 libraries separately. They are included in the *media* (note, not the distribution) as `extra\CompatibilityLibrares MSL 4.0.0.zip`. To use them, unpack them in, typically `…Program Files\Dymola 2026x\Modelica\Library`.

Notes:

- Commercial libraries are not included; the process only covers MSL and related libraries, e.g. ModelicaServices.
- You need administrator privileges to install.
- The zip folder contains few files; most libraries can use both MSL 4.1.0 and 4.0.0.
- After installation, don´t forget to apply the command **Tools > Options**, select the **Version** tab, select the Modelica version you want, and tick **Force upgrade of models to this version** to select the proper version.
- Remember to restart after having changed the version as above, to make sure that you handle also the already loaded libraries, if any.

3.2.2 Table lookup for MSL or Dymola

To better support handling of 3D and multi-dimensional tables (ND), `DataFiles.TableND` (that is, the block `TableND` in the library `DataFiles`) has been improved:

- The table is now not read during translations. (If you still want to, there is a new block `TableNDStatic` available for this.)
- It is easier to select files.

(The library can be opened by the command **File > Libraries > More Dymola > Data Files**, or you can type the command **import DataFiles** in the command input line of the command window.)

3.2.3 Minor improvements

Handling non-physical units as display units

You can now declare non-physical units, for example, currencies. This allows currencies as display units, and conversion between different units.

The new annotation `__Dymola_UnitDefinitons` defines a new unit. For conversion, the present `defineUnitConversion` can be used in the annotation, but you must use named arguments. An example with the currency "XXX" is:

In a package, define XXX and the conversion by an annotation:

```
annotation(__Dymola_UnitDefinitions(BaseUnit(symbol="XXX", quantity="currency"),
 UnitConversion(defineUnitConversion(from_unit="XXX", to_unit="USD", scale=1.2))));
```

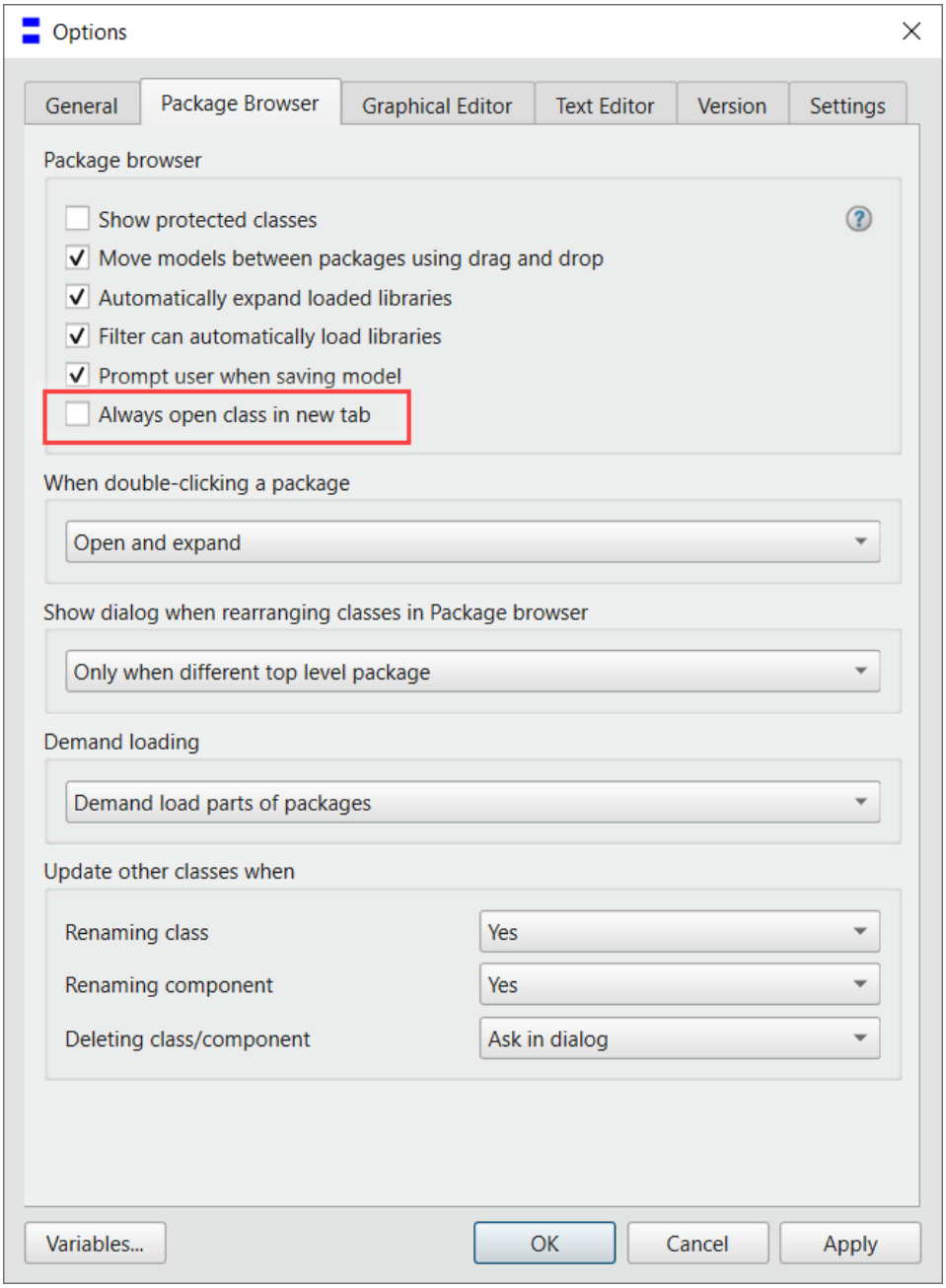
In a model M in that package, you can define a parameter with this display unit:

```
model M
  parameter Real x(unit="XXX")=2;
end M;
```

Note that if you try the above to test the feature, you must save the package, unload it and load it again – the feature works when loading the package (which is the usual case), but currently not when adding the package internally in Dymola.

Opening a class from the package browser in a new tab

When you open a class from the package browser, by default you open it in the current tab in Dymola main window. To open it in a new tab instead, you can activate the option **Always open class in new tab**. You reach this option by the command **Tools > Options**, the **Package Browser** tab:



By default, this option is not activated, that is, you open the class in the current tab. The setting is saved between sessions.

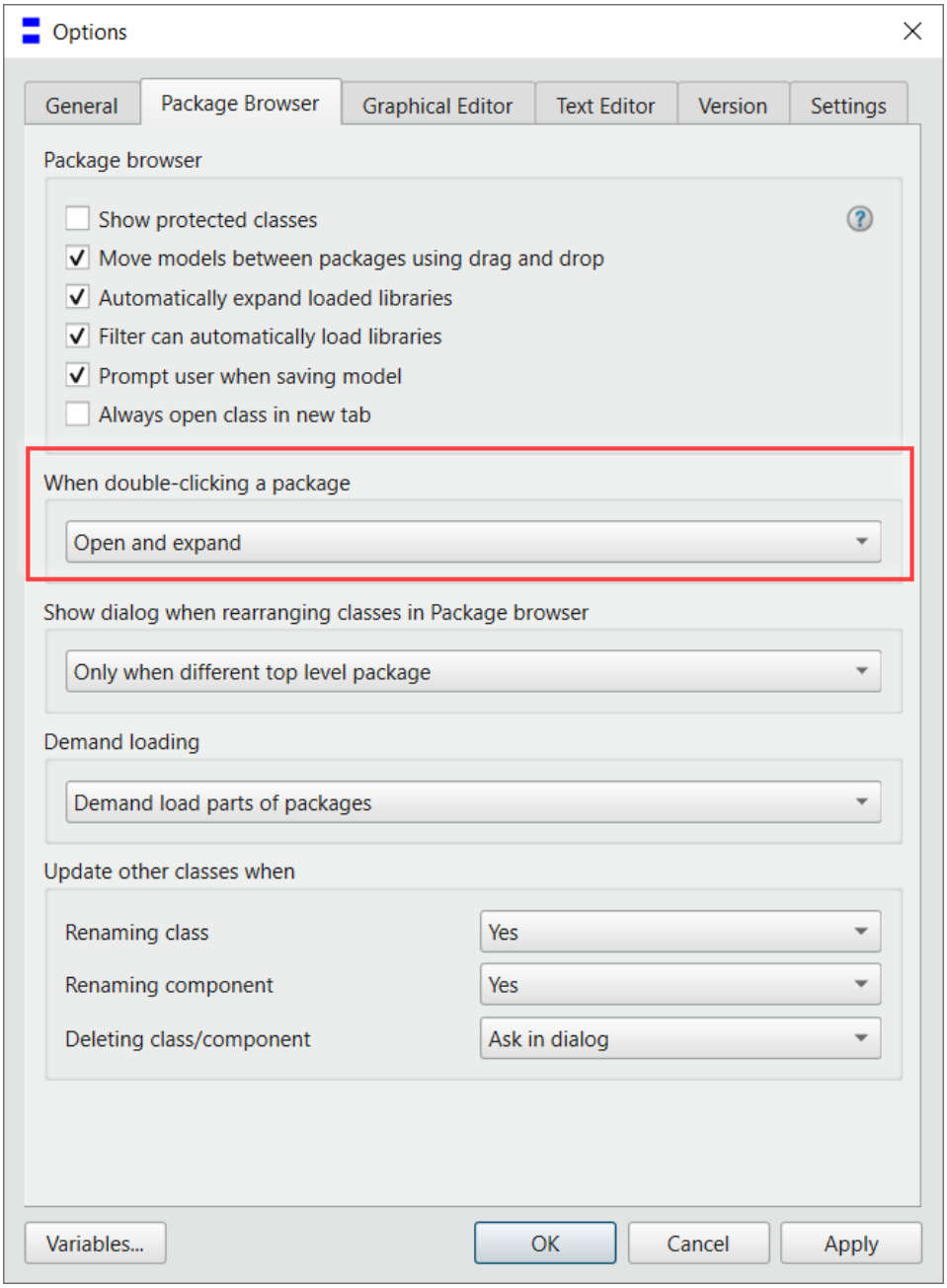
There is a corresponding flag `Advanced.Editor.OpenClassInNewTab` that by default is `false`.

If you want to keep the default behavior of opening a class in the current tab, but still sometimes easily open a class in a new tab, you can use the new shortcut **Shift+Enter** that always opens the selected class in the package browser in a new tab. (Actually, you can also use **Shift+Double-click** on the class in the package browser to do this.)

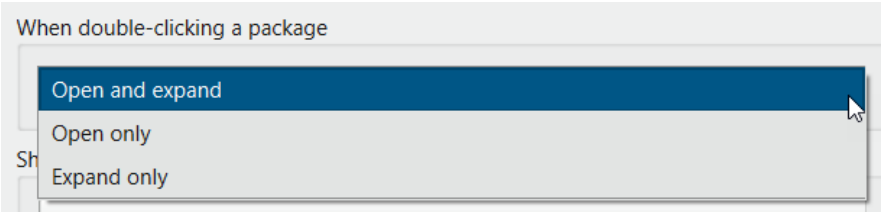
Note. For scripting, you can use the argument `newTab` in the built-in function `openModelFile` to specify how a new class should be opened. This argument was introduced in the previous Dymola version.

Option to prevent expanding the package browser when double-clicking a package

You now have three options to select from to decide what will happen when you double-click a package in the package browser. You can use a setting in **Tools > Options**, the **Package Browser** tab:



Using the arrow, you have three options:



By default, you open and expand the package when double-clicking it. The setting is saved between sessions.

The option of only opening the package is new for Dymola 2026x; the others were available in previous versions as well, using a setting **Double-click on package opens it in the editor, in addition to expanding the tree** in the above **Package Browser** tab. That setting is now removed from the GUI.

There is a new flag corresponding to the options in the figure above:

```
Advanced.UI.DoubleClickPackage
```

The flag can have any of the values:

- 0 – Open and expand (default value)
- 1 – Only open package
- 2 – Only expand package

Having now this new flag, it covers also the old flag `Advanced.Editor.DisplayPackage` that controlled if the package was opened in the editor when double-clicking (it corresponds to the removed GUI setting above). That flag is now seen as deprecated but is still present.

(There was also an old flag `Advanced.UI.SingleClickOpensTree` that was supposed to control opening the tree in the package brower by a single click. Not working since long, it has now been removed.)

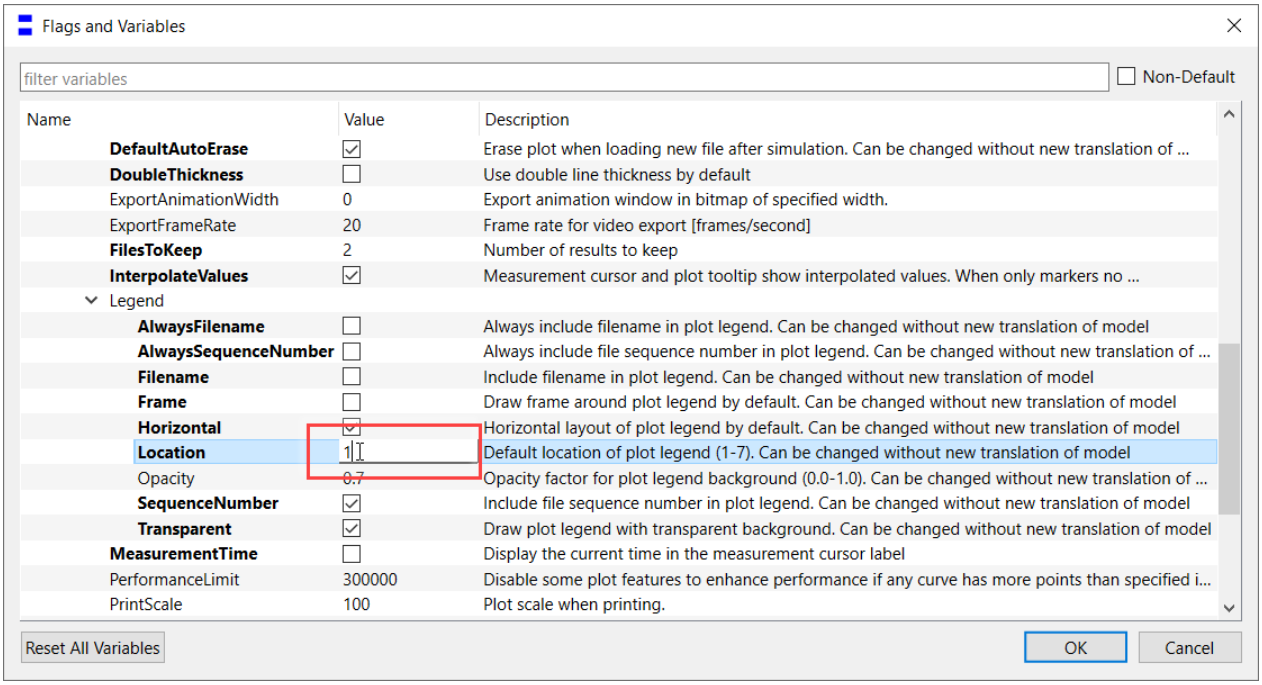
Minor improvement in package browser handling

If you select a class and press **Enter**, you open it.

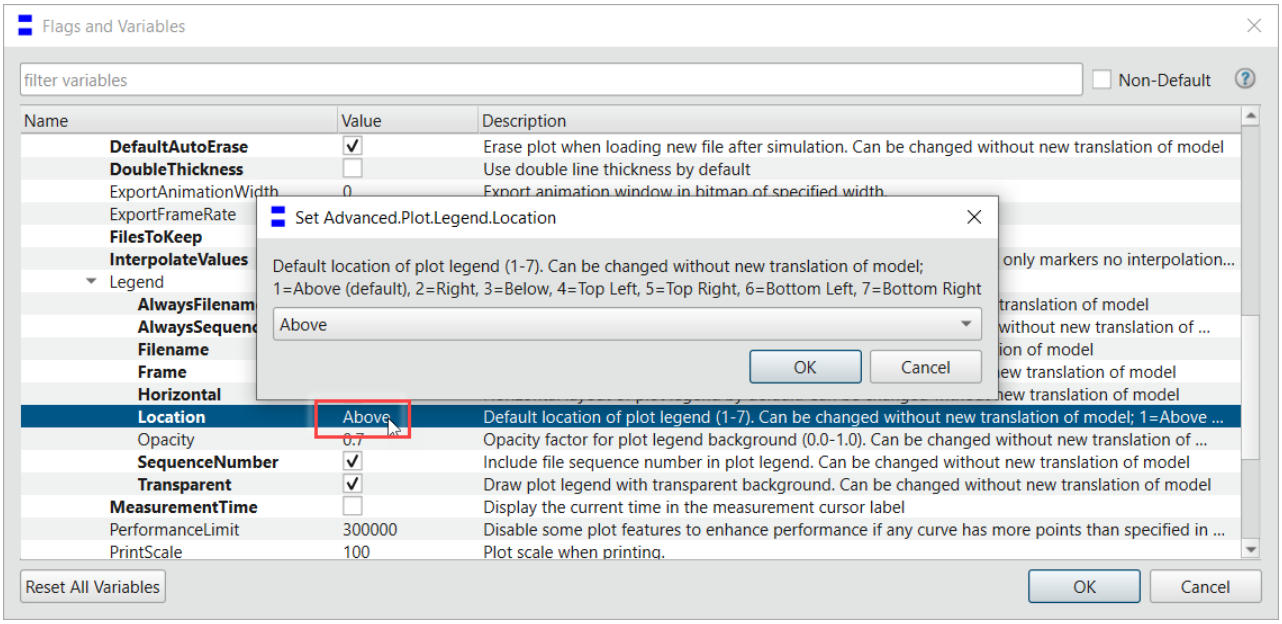
Improvements in the Flags and Variables dialog

Improved GUI when working with integer flag handling

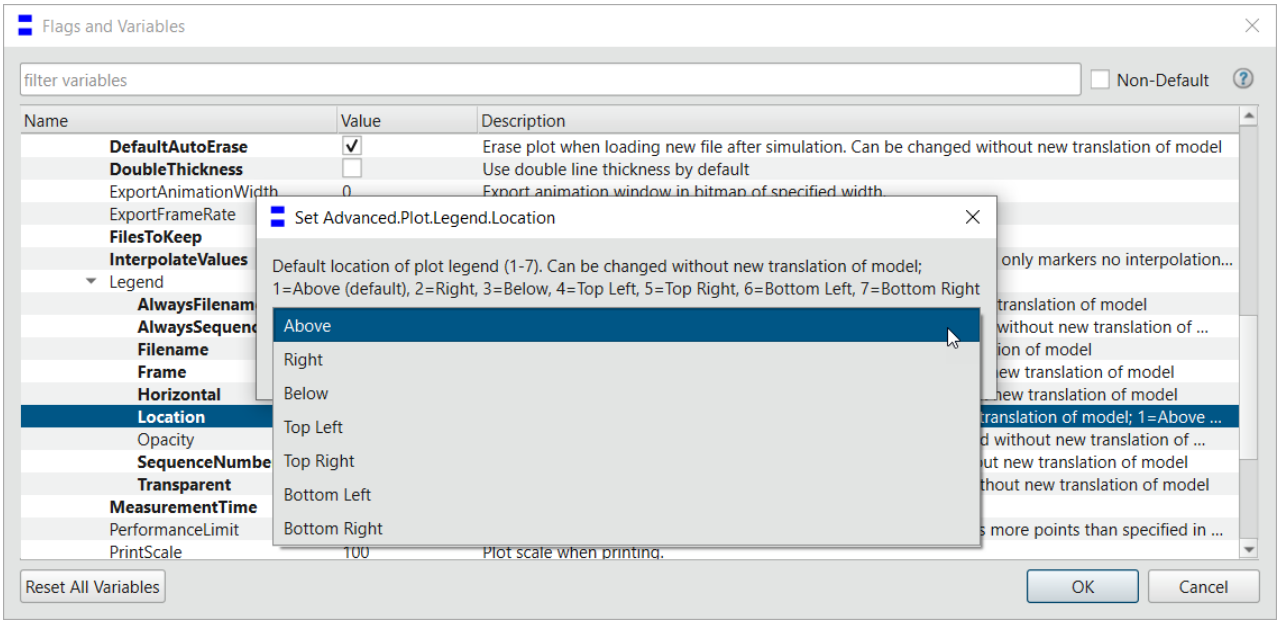
In the **Flags and Variables** dialog (reached by the command **Tools > Options > Variables…**) you can set flags and variables. For integer variables this can in practice be seen as enumerations, the display has been improved to use that fact. As an example, compare the variable `Advanced.Plot.Legend.Location` between the previous Dymola version and Dymola 2026x. In the previous version, you set the variable by entering an integer value:



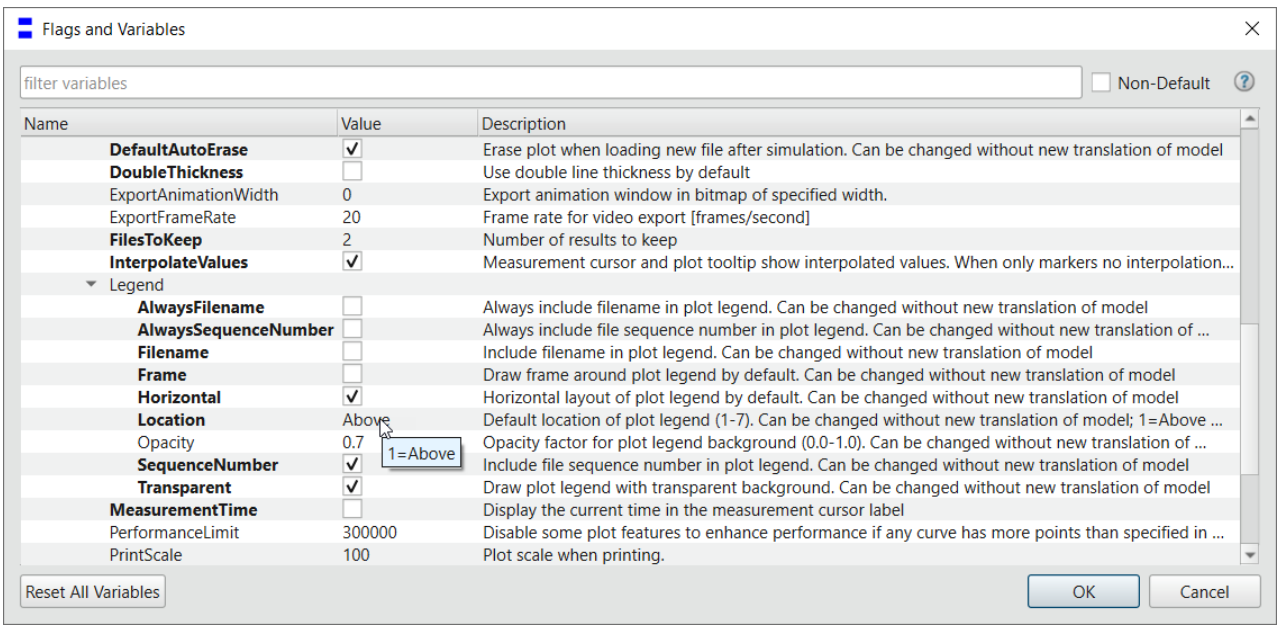
In Dymola 2026x, you see the selection corresponding to the integer value instead, and if you click that selection, you get a new dialog:



Using that dialog, you can select from the alternatives:

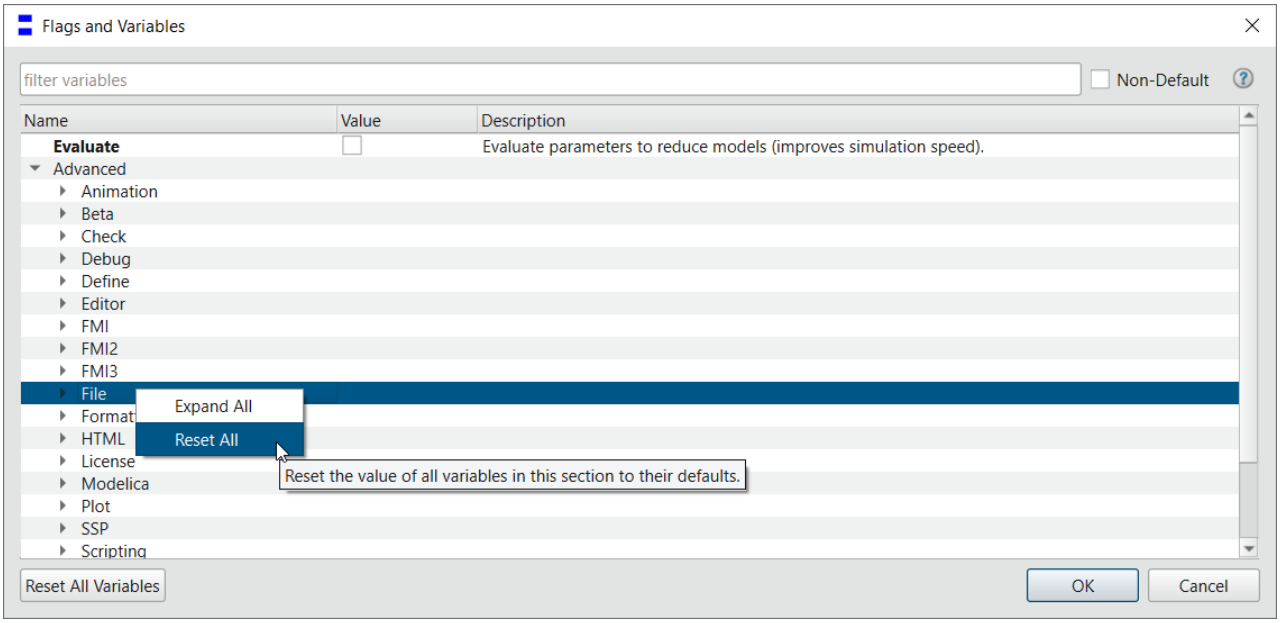


Note the new tooltip that shows both the integer and the integer value and the corresponding "enumeration value".



Context menu to expand a flag group or reset the included flags

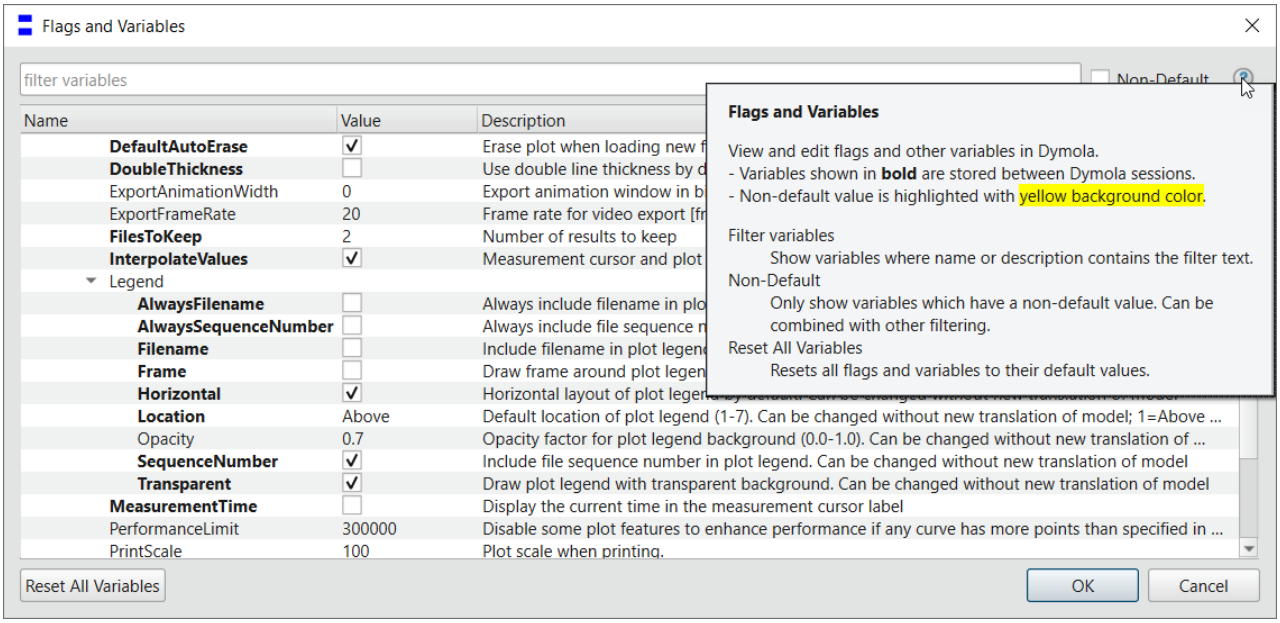
A context menu is available for flag sections/groups, an example:



- **Expand All** expands the section/group, displaying all flags in that section.
- **Reset All** reset all flags in that section/group to default values.

Help button explaining the features of the dialog

Note the minor improvement with a Help button to explain the features in the dialog, like the use of bold text and yellow highlighting of flag names:



Modelica Text editor improvements

Options for automatic indentation behavior while typing

In Dymola 2026x a new flag controls the behavior of automatic indentation while typing. The default behavior of the default automatic indentation while typing is also slightly changed.

The flag is `Advanced.Editor.InteractiveAutoIndentation`, and the possible values are:

- **0** - No automatic indentation
- **1** - Use previous line indentation for new lines
- **2** - Automatic indentation on enter and space (default value)

The flag value is saved between sessions.

For multi-line comment text, the default behavior now is to use the previous line indentation for any new line created when pressing enter/return, and otherwise perform no interactive formatting. Compared to the previous behavior, the new behavior helps to add user formatting within multi-line comment, like indented lists or commented code excerpts.

Display of selected class improved

In the Modelica Text editor, you can select an item either by putting the cursor inside it, or by selecting the complete text. You can then right-click and select **Selected Class > Open Class.** (There are alternatives here, depending where you want to the class to be opened).

In Dymola 2026x, when putting the cursor inside a text, Dymola will look for the longest text that can be searched for, while when selecting a specific text, that text is used for the search.

From this search, the relevant base class will be shown.

Note that in Dymola 2026x, the corresponding search in error messages will work the same.

Improved parsing error messages

Some error messages were previously very short if they included a list of similar tokens, these error messages now contain more information, making them clearer.

Checking bus signal sets for missing signal sources

When checking a model, there is now a check if all signals in bus signal sets do have sources, that is, they are given values by matching non-inputs. By default, warnings are generated if signals in bus sets miss sources. If you want to have errors messages instead, you can set the flag (before the checking):

```
Advanced.Modelica.PedanticBus = true
```

(The default value of the flag is `false`.) The flag is saved between sessions.

Minor code generation improvement

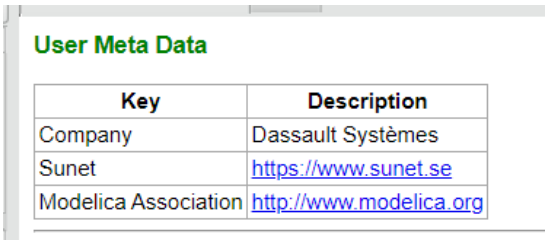
The code generation for local function changing defaults (as in the TIL library) has been improved to produce less code. The feature can be disabled by setting the flag:

```
Advanced.Translation.MergeDuplicateFunctions = false
```

(The flag is by default `true`.) The flag is saved between sessions.

Model metadata support: convert metadata to clickable links

When presenting metadata in the documentation, value strings that are considered to be URLs are now converted to clickable links, and can thus be opened in a web browser.

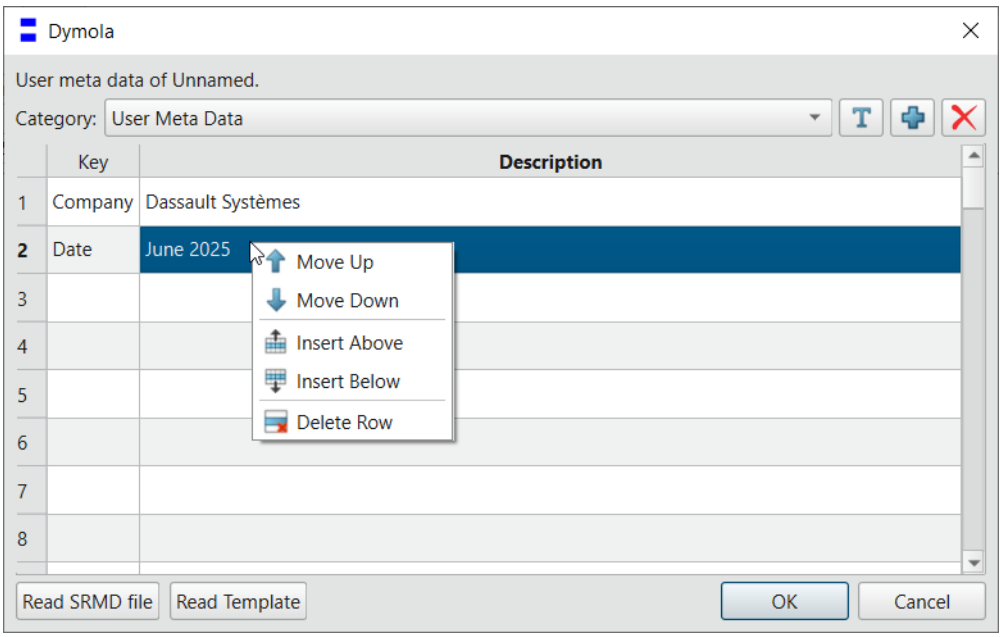


Note. You must use `http://www...` or `https://www...` when entering the text, just using `www...` is not supported.

Model metadata editor improvements

Context menu

A context menu has been added:



The context commands are:

Command	Description
Move Up	Swaps the current row with the row above.
Move Down	Swaps the current row with the row below.
Insert Above	Inserts a new row above the current one.
Insert Below	Inserts a new row below the current one.
Delete Row	Deletes the current row and moves the following rows up.

Editing by drag and drop

You can edit the table by the following drag and drop operations:

Operation	How-to
Copy cell	Select a cell and drag it to another cell.
Copy row	Select the entire row by clicking on the row number at the left, then drag the row and drop it in the **Key** field of a new row.

Copy and insert row	Same as copying a row, but drop the dragged row *between* two rows - the line separator between the rows will be highlighted.

3.3 Simulating a model

3.3.1 Plot tab

Using macros in plot signal legends

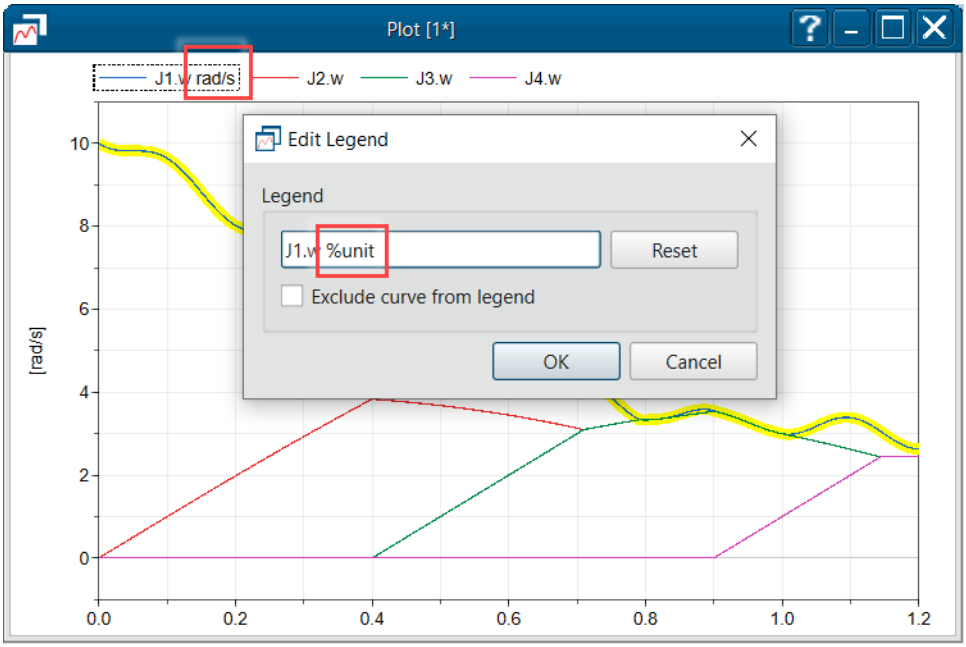
In previous Dymola versions, you could use macros when creating plot axis titles by macros. In Dymola 2026x, this is also supported for the legends of plotted signals, for some macros. The macros supported are:

- `%name` (the signal name)
- `%unit` (the current display unit)
- `%desc` (the signal description)

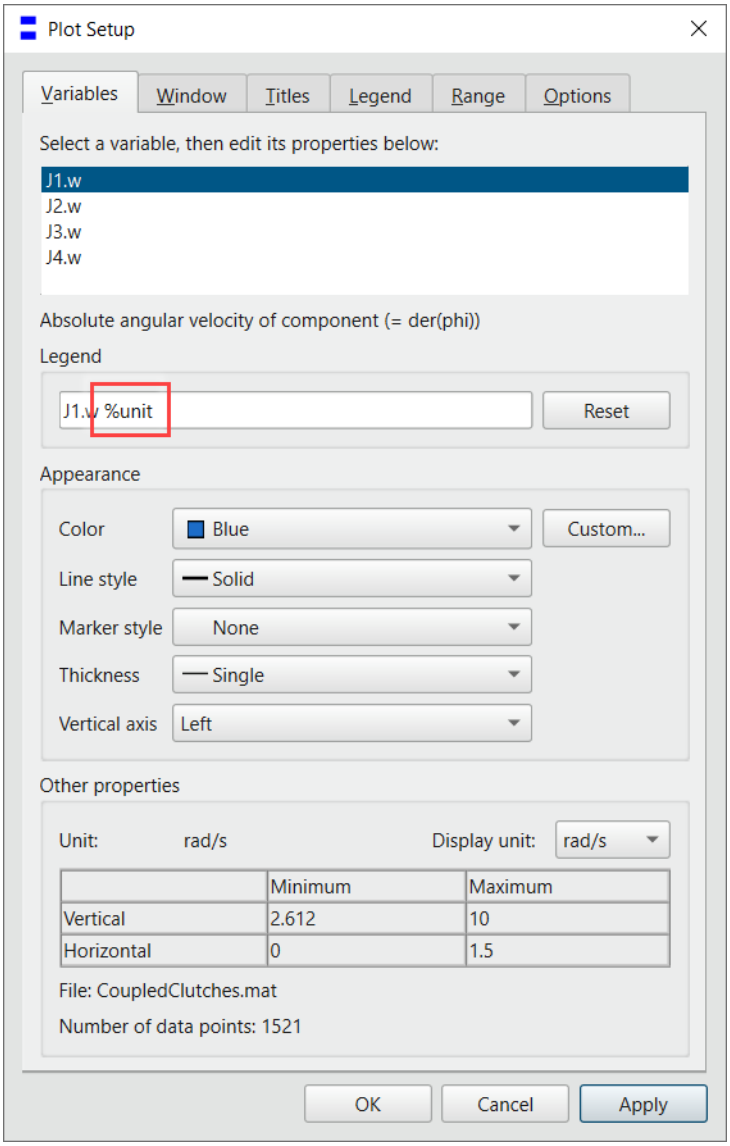
You can apply them in two ways:

Applying macros for signal legends in the existing plot GUI

By double-clicking a signal legend, or right-clicking the signal legend (or the corresponding curve) and selecting **Edit Legend…**, a dialog appear, where you can edit the legend text. An example of adding unit in the presentation:



You can also edit the legend from the plot setup, in the **Variables** tab (reached by right-clicking in the plot and selecting **Setup…**). You must also select the signal to edit:



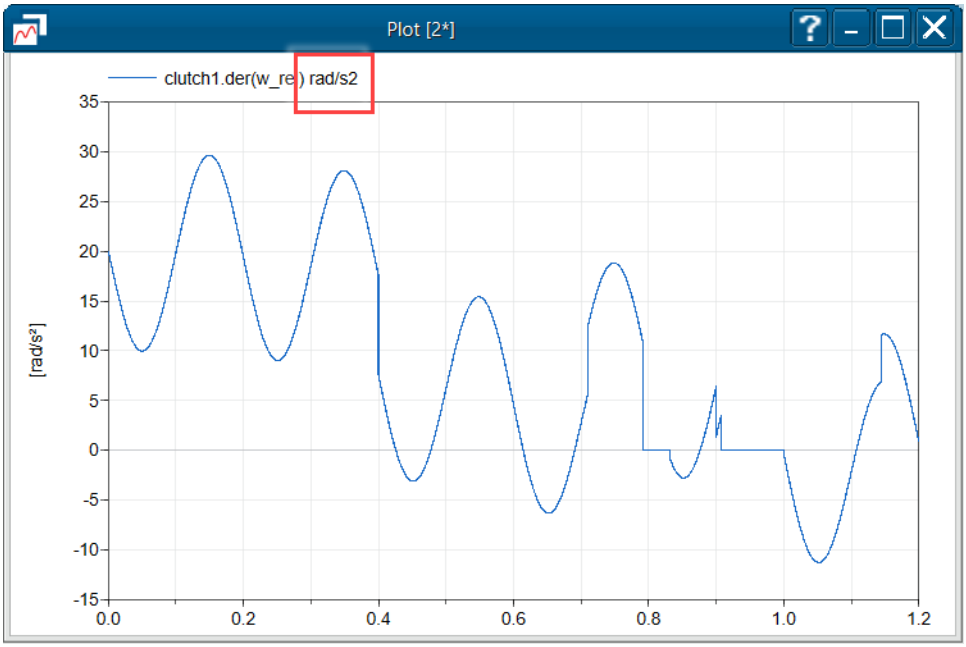
To remove any editing, you can click the **Reset** button.

Applying macros for signal legends in the built-in functions plot or createPlot

You can use the macros in the argument for signal legends in the built-in functions `plot` or `createPlot`. A simple example using the built-in function `plot` in the Coupled Clutches demo when having first created an empty plot window:

```
plot({"clutch1.der(w_rel)"}, legends={"clutch1.der(w_rel) %unit"});
```

This will give:



A possible scenario where `%unit` can be used:

- Two signals are plotted in the same diagram.
- The plot script defines custom legends for the signals.
- Because the signals have different units, and both are plotted against the left axis, we cannot use the axis description to display the unit.
- One work-around would be to plot using both the left and right axis, but that does not work well here because the range is comparable but not the same – and the point is to compare the signals.

The solution now possible is to include `%unit` in the custom signal legends strings.

3.3.2 Animation tab

Improved animation using OpenGL

In Dymola 2026x, there is full support of software emulation of OpenGL. This means that you can run animation and Plot3D in Dymola without having any GPU (Graphical Processing Unit) in the computer.

You activate the software emulation of OpenGL by setting the environment variable QT_OPENGL to software, that is:

```
set QT_OPENGL=software
```

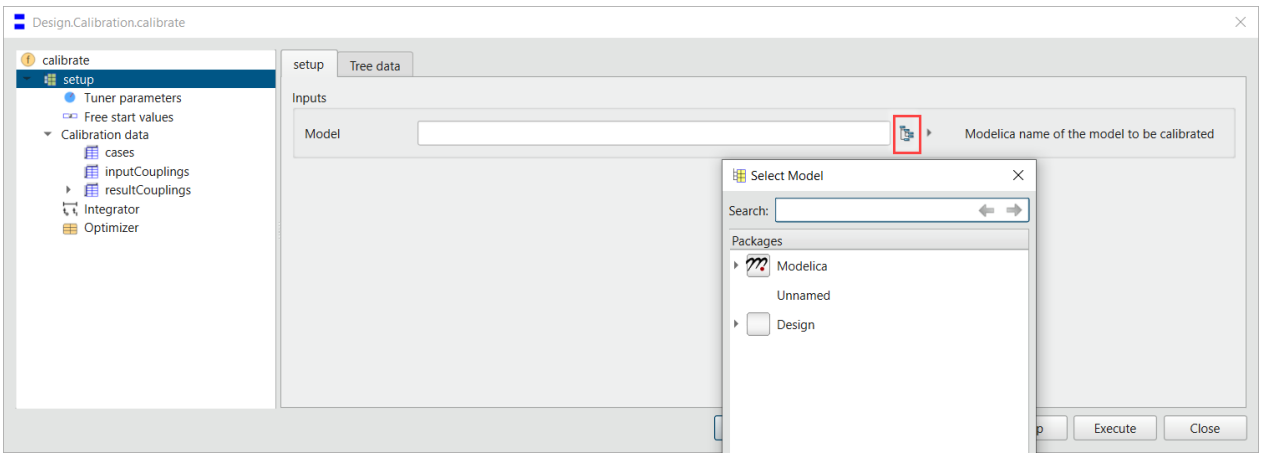
3.3.3 Scripting

New built-in function getRegistryValue

A new built-in function `getRegistryValue` has been added, to be able to read Windows registry keys. For more information, see section “Reading Windows registry keys” on page 58.

Improved annotation __Dymola_translatedModel

The annotation `__Dymola_translatedModel` is used to specify a model that is to be (or is) translated, to be used in further processing. An example from the Design package:



If the default is that no model is selected, as in the above example, executing the function will do one of the following:

- If the function call is used in a command in a model, the call will assume that you want to select the current model (and translate it if not already done).
- Otherwise, if some model is translated, it will be used as the default.
- If that is not the case, the function call will be empty, and you have to select a model using the dialog in the example above.

The first case is now used in `Design.Calibration.Examples.SimpleCar`, which means that copying and changing this model, and then running the calibration will give the correct result.

New, changed, or deleted advanced flags

New Advanced flags

New flags added in Dymola 2026x:

New flag	Default value	Description	Saved between sessions
Advanced.Beta.FMI. ExportLsStructTerminals	false	Generate terminals and icons for FMI Layered Standard for Structured Data. See section “FMI 3: Support for terminals and icons for FMI Layered Standard for Structured Data” on page 68.	Yes
Advanced.Beta.FMI. ExternalLibrary	false	Use external library for the FMI functions. See section “FMI 2 and FMI 3: Option to use a standardized package of FMI functions when importing FMUs” on page 69”.	No
Advanced.Beta.FMI3. IncludeTerminalsAndIcons	false	Includes terminals and icons in both export and import of FMUs. See section “FMI 3: Support for terminals and icons in general” on page 66.	Yes
Advanced.Beta.SSP. ExportExternalBinding	0 Inline par. binding	Export SSP parameter bindings as external files, if possible. See section “SSP: Option to export SSP parameter bindings as external files” on page 72.	Yes
Advanced.Beta.SSP. MinimumVersion2	false	Always store system description as Version 2.0. See section “SSP: Specifying the SSP version to 2.0 when exporting” on page 72.	Yes
Advanced.Beta.Translation. ArrayDeclare	false	When possible declare arrays instead of array elements. See section “More efficient handling of arrays” on page 72.	No
Advanced.Beta.Translation. TransportDelayDiscontinuity	false	For testing propagating discontinuities with spatialDistribution.	No

		See section “Minor issue with the built-in function spatialDistribution” on page 73.	
Advanced.Editor. InteractiveAutoIndentation	2 Automatic indentation on enter and space	Automatic indentation behavior of Modelica text editor while typing. See section “Options for automatic indentation behavior while typing” on page 23.	Yes
Advanced.Editor. OpenClassInNewTab	false	When opening a class using the Package browser, open it in a new tab. See section “Opening a class from the package browser in a new tab” on page 15.	Yes
Advanced.Editor. PromptDelayedConversion	false	Show a prompt when delaying conversion. See section “Using MSL 4.1.0 without updating the models” on page 13.	Yes
Advanced.FMI. DelayUnpackingOfBinaries	false	Delays unpacking of binaries in fmu to translation. See section “FMI Import: Delayed unpacking of FMU” on page 98.	No
Advanced.FMI. ShortDefaultModelIdentifier	false	Only uses last part of model name when generating modelIdentifier avoid long default modelID. See section “FMI Export: Shorter default model identifiers” on page 96.	Yes
Advanced.LibraryManagement. Source.MergeWithDefault	false	Whether to merge Advanced. LibraryManagement.Source.URL with the default. See section “Recommended libraries for on-demand download and installation” on page 42.	Yes
Advanced.LibraryManagement. Source.URL	"" (empty string)	URL for XML file containing the list of recommended Modelica	Yes

		libraries. If empty, default URL is used. See section "Recommended libraries for on-demand download and installation" on page 42.	
Advanced.Modelica. PedanticBus	false	Pedantic check for expandable connectors. See section "Checking bus signal sets for missing signal sources" on page 24."	Yes
Advanced.SSP. ExportCopyResources	true	If true, existing 'resources' and 'extra' folders are included when exporting. See section "SSP read-modify-write cycle" on page 94.	Yes
Advanced.SSP.ExportProtected	false	Include protected members in exported SSP. See section "Exporting protected items" on page 96.	Yes
Advanced.SSP.ProvidePlacement	true	Auto-generate placement annotation for components without elementGeometry. See section "Providing placement when needed" on page 95.	Yes
Advanced.SSP.PurgeResources	1 Purge ordinary files only	When importing an SSP file, remove any existing 'resources' or 'extra' directory; 0=No purge, 1=Files only (default), 2=Files and directories. See section "Purging resources when importing SSP files" on page 95.	Yes
Advanced.SSP.TempUnpack	false	When importing an SSP file, unpack in local directory. See section "Unpacking an SSP file in a temporary directory" on page 95.	Yes
Advanced.Translation. MergeDuplicateFunctions	true	Merge duplicate functions.	

Advanced.Translation. SmartConditionalJacobian	2 Before evaluating Jacobian	Evaluate parameters to avoid differentiation failure for conditional calls. See section "Handling MSL 4.1.0 rotational friction models using tables, for example, clutches" on page 12.	No

Removed and deprecated Advanced flags

Removed and deprecated flags in Dymola 2026x:

See below section, the old names are not valid anymore if they are changed. A message will appear when executing the scripts in such cases.

Removed (R) or Deprecated (D) flag	Reason for removing/deprecating	Reference
Advanced.Beta.FMI3. ExportTerminalsAndIcons (R)	Now included in the flag Advanced.Beta.FMI3. IncludeTerminalsAndIcons	See section "FMI 3: Support for terminals and icons in general" on page 66.
Advanced.Editor. DisplayPackage (D)	Replaced by the flag Advanced.UI. DoubleClickPackage.	See section "Option to prevent expanding the package browser when double-clicking a package" on page 17.
Advanced.UI. SingleClickOpensTree (R)	Not needed.	See section "Option to prevent expanding the package browser when double-clicking a package" on page 17.

Changed Advanced flags

Changed flags in Dymola 2026x:

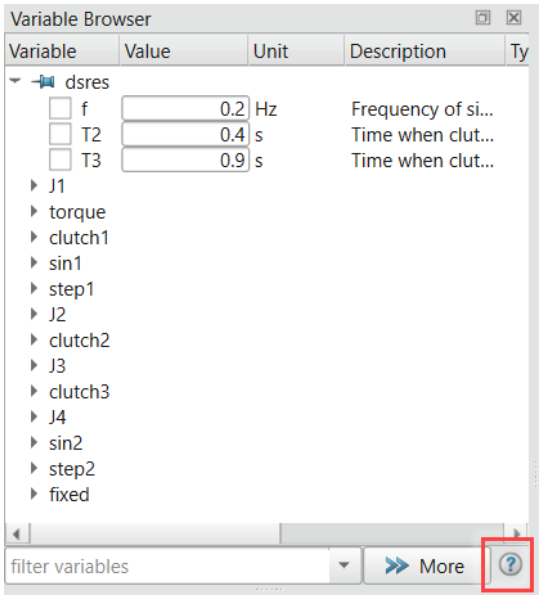
Changed flag	Change	Description
Advanced.Editor. ShowStartValues	Description (due to changed related functionality).	As default (true) show start values in Variable Browser, if false show values (at selected time). Can be

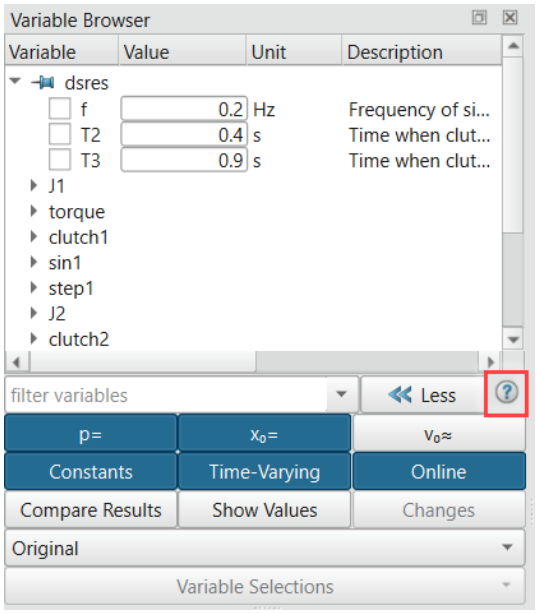
		changed without new translation of model. See section "Variable browser: Help texts, a renamed command" on page 35.
Advanced.FMI. CompileWith32	Description (for better clarity)	Attempt to generate 32-bit binary in fmu. Implicitly false for WSL compiler.
Advanced.FMI3. ExposeDynamicArrays	Default value (to false)	Expose dynamic arrays for fmu export, requires that the user sets them.
Advanced.Translation. CompileWith64	Description (due to changed DDE functionality).	Compile the model as 64-bit. Not significant for WSL or DDE (64-bit only). See section "Discontinued support for 32-bit embedded DDE server" on page 58.

3.3.4 Minor improvements

Variable browser: Help texts, a renamed command, and minor GUI improvements

You have now a **Help** button for both the basic filtering and the advanced commands in the variable browser:





The help texts are, respectively:

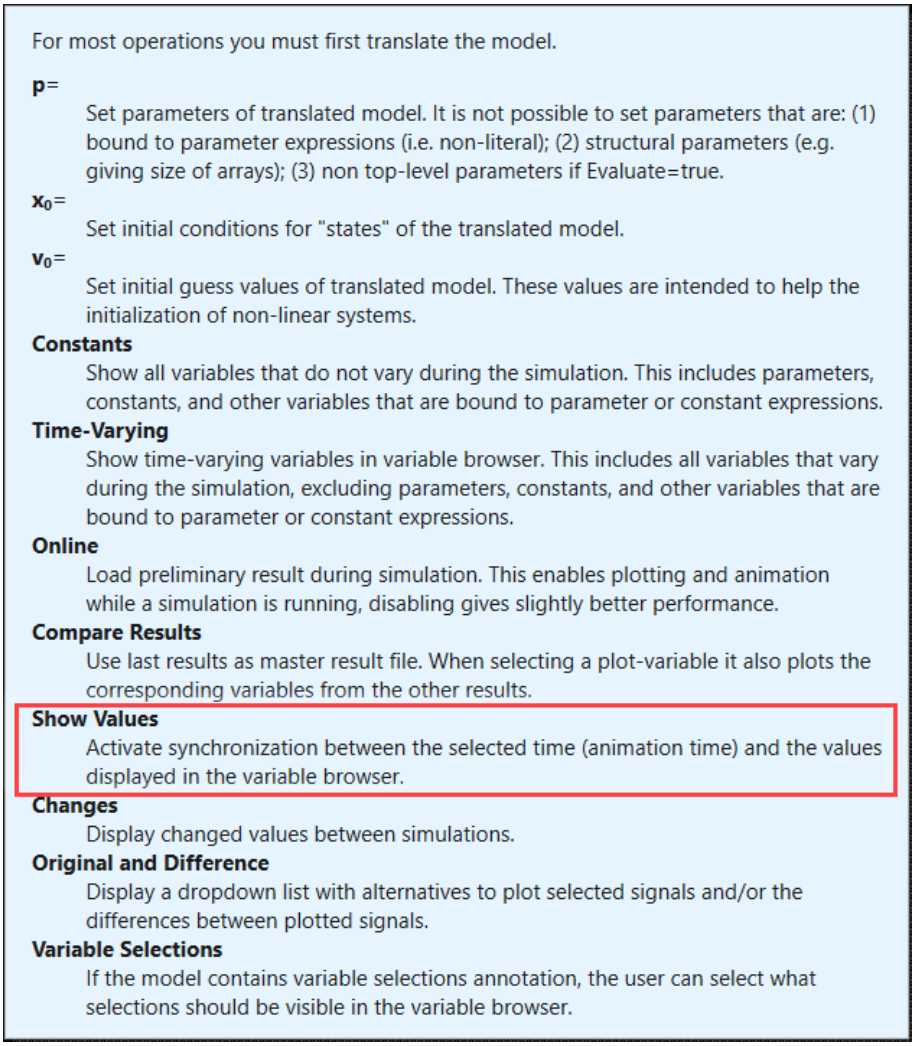
Filter variable

Only show certain components. Characters in the filter are by default not case-sensitive (changeable using the filter context menu). Special symbols are:

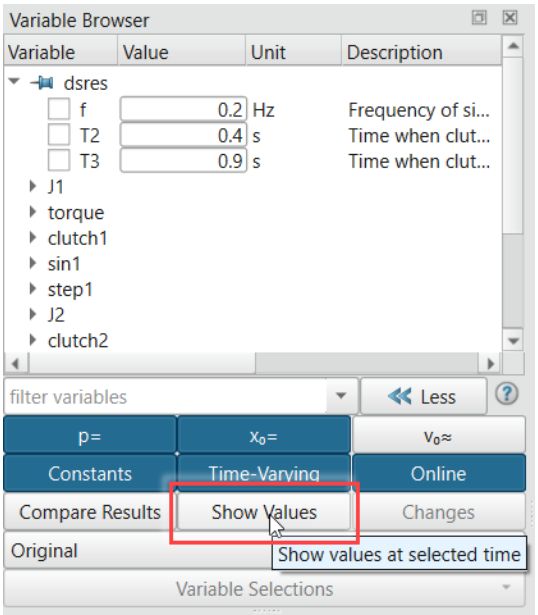
*****	everything
?	any character
{ab}	characters a or b
{a-z}	characters a through z
{^ab}	characters except a and b
E+	one or more occurrences of E
(ab|cd)	ab or cd
\d	any digit
\w	any digit or letter
^	match from start
$	match at end

More, Less

Show options to filter based on type of variable and compare different results.



The command in the advanced display, previously named **Show Start Values**, has now been renamed to **Show Values**, to indicate better the usage. The behavior of the button has been changed accordingly; you have to activate/press the button to see the values at simulation time. If the button is not activated/released, you see the start values after translation.



The help text has been clarified accordingly; see the framed description in the help text window above.

The corresponding flag `Advanced.Editor.ShowStartValues` that controls the button still has the same default value (`true`) as previously, but the description is changed to match the new behavior of the button.

The minor improvements are using blue color and white text for activated buttons in the advanced mode to indicate better which buttons are activated.

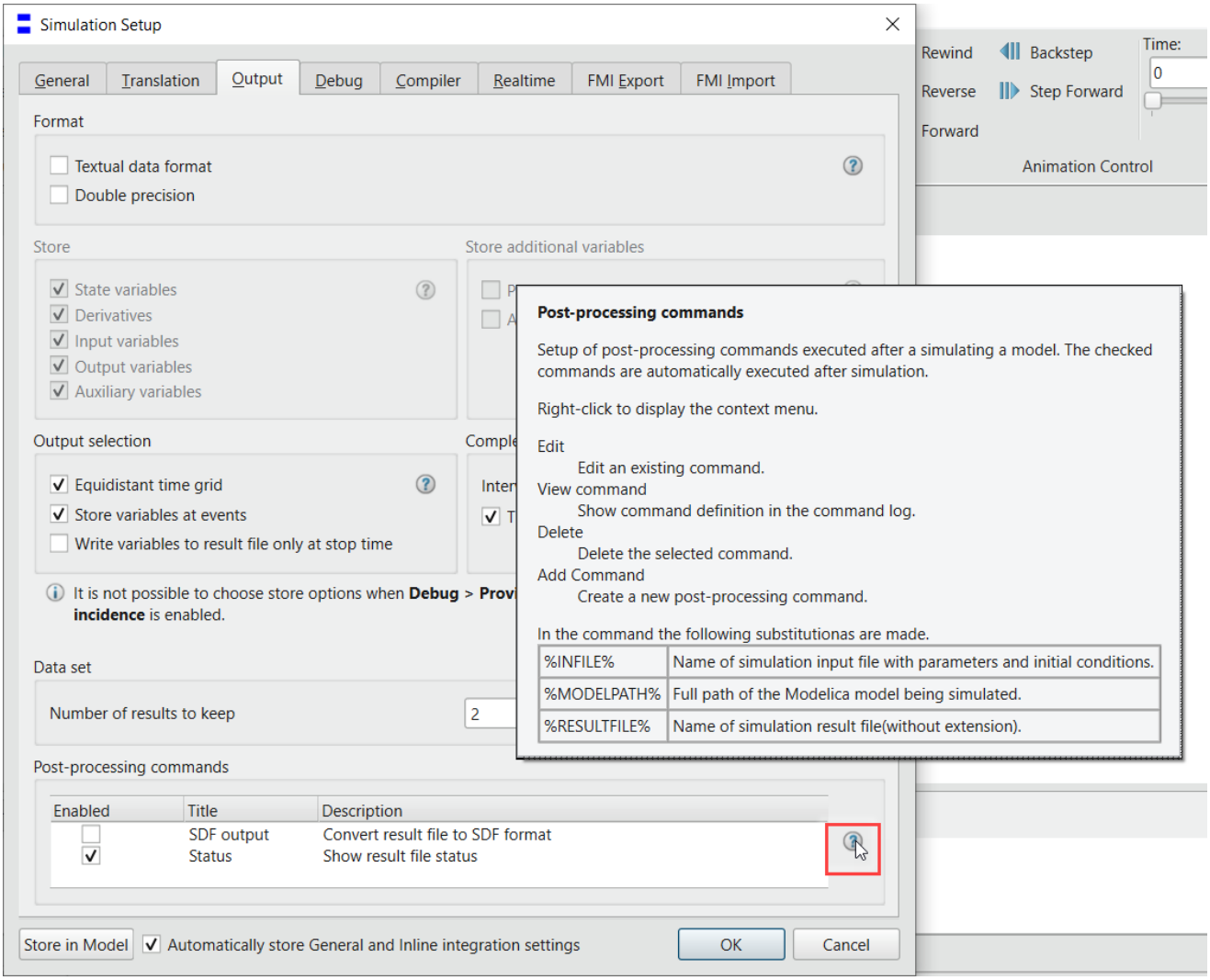
Finally, in the context menu for the variable filtering, the option not using regular expressions in the filtering has been removed; there are no cases when you need to disable this option.

Improvements in the post-processing commands handling

In the simulation setup, reached by the command **Simulation > Setup**, you can, in the **Output** tab, specify post-processing commands. In Dymola 2026x, some improvements have been done, for example:

- Support for substitution of `%MODELPATH%` in the commands.
- Adding more documentation in the **Help**-button.

The added documentation in the **Help** button now covers the feature well:


Simulation Setup
General
Translation
Output
Debug
Compiler
Realtime
FMI Export
FMI Import
Format
Textual data format
Double precision
Store
State variables
Derivatives
Input variables
Output variables
Auxiliary variables
Store additional variables
Output selection
Equidistant time grid
Store variables at events
Write variables to result file only at stop time
It is not possible to choose store options when Debug > Prov
incidence is enabled.
Data set
Number of results to keep
2
Post-processing commands
Enabled
Title
Description
SDF output
Convert result file to SDF format
Status
Show result file status
Store in Model
Automatically store General and Inline integration settings
OK
Cancel
Rewind
Backstep
Reverse
Step Forward
Forward
Time:
0
Animation Control
Post-processing commands
Setup of post-processing commands executed after a simulating a model. The checked commands are automatically executed after simulation.
Right-click to display the context menu.
Edit
Edit an existing command.
View command
Show command definition in the command log.
Delete
Delete the selected command.
Add Command
Create a new post-processing command.
In the command the following substitutionas are made.
%INFILE%
Name of simulation input file with parameters and initial conditions.
%MODELPATH%
Full path of the Modelica model being simulated.
%RESULTFILE%
Name of simulation result file(without extension).

3.4 Installation

For the current list of hardware and software requirements, please see chapter "Appendix – Installation: Hardware and Software Requirements" starting on page 104.

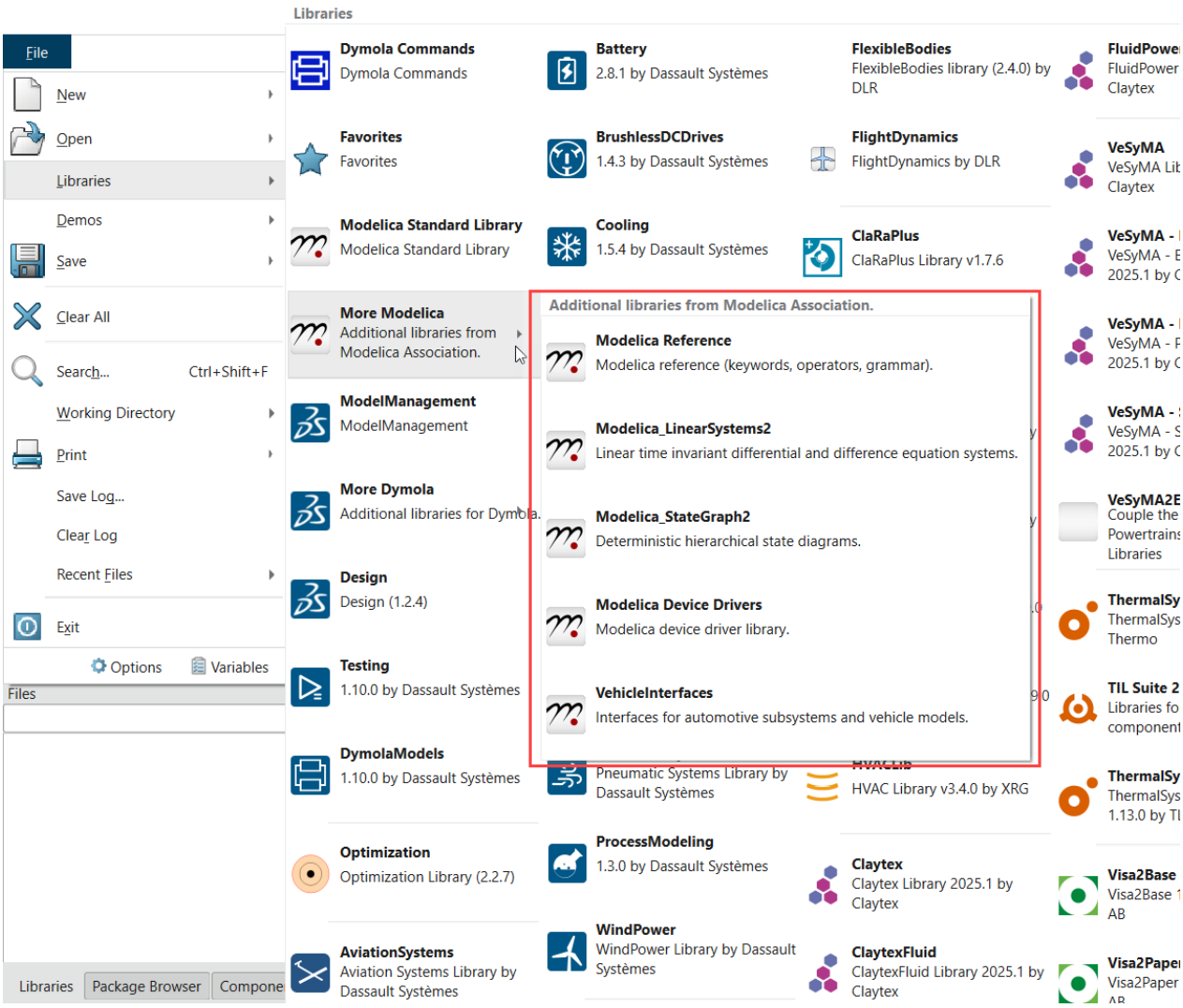
3.4.1 General improvements

Improved GUI for opening libraries

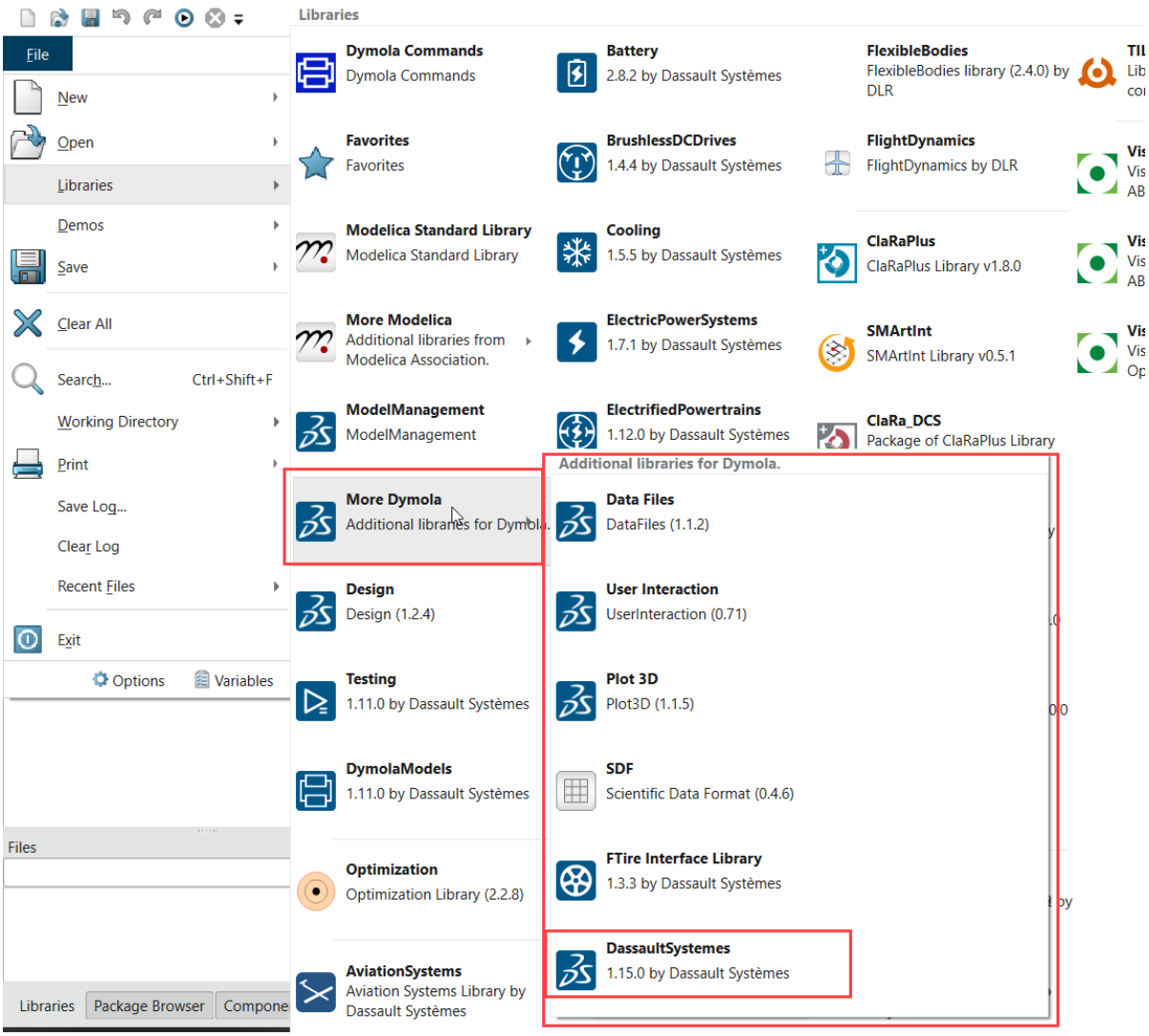
The GUI for the command **File > Libraries** has been simplified by collecting yet another number of previous separate libraries in the dialog under a group, in this case the group **More Modelica**.

The libraries that have been grouped under **More Modelica** are:

- Modelica Reference
- Modelica_LinearSystems2
- Modelica_StateGraph2
- Modelica Device Drivers
- VehicleInterfaces



Note that in the already existing group **More Dymola** the library Dassault Systèmes has now been added:



This library is licensed; if you install Dymola with no extra libraries installed, this entry is missing.

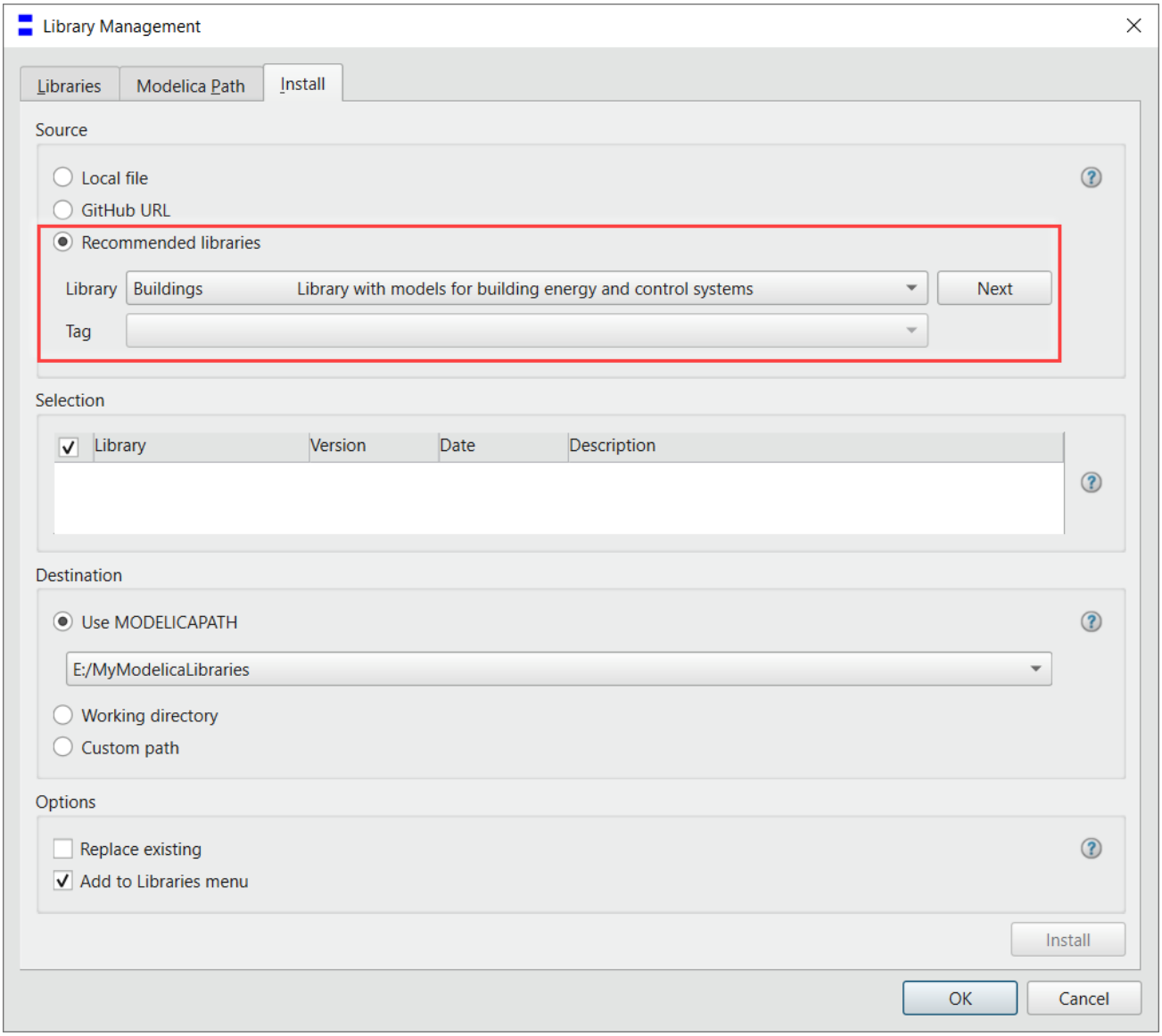
Recommended libraries for on-demand download and installation

Downloading and installing from the list of recommended libraries

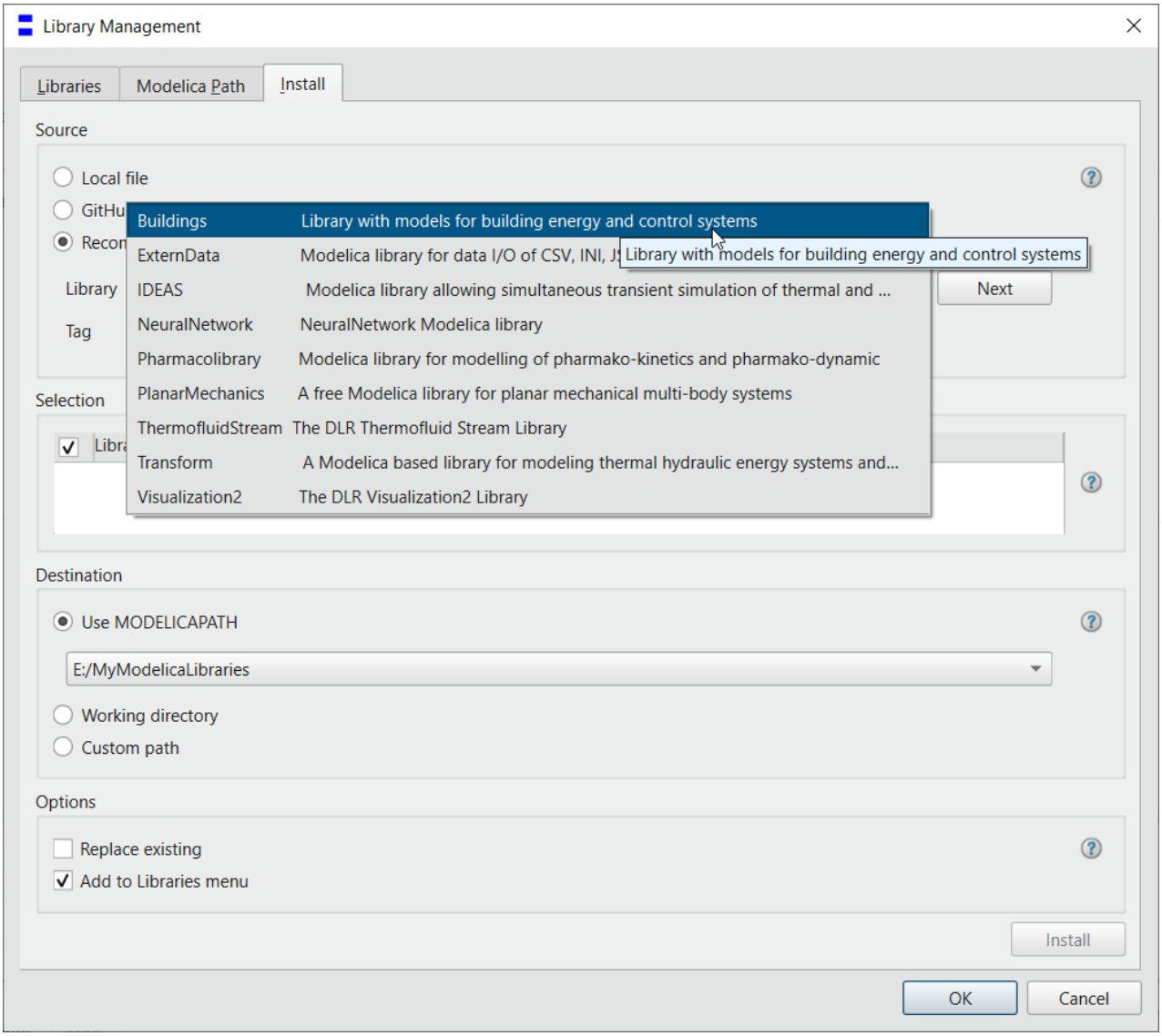
In Dymola 2026x, a list of recommended libraries for on-demand download and installation is available.

To download and install any of these libraries, do the following:

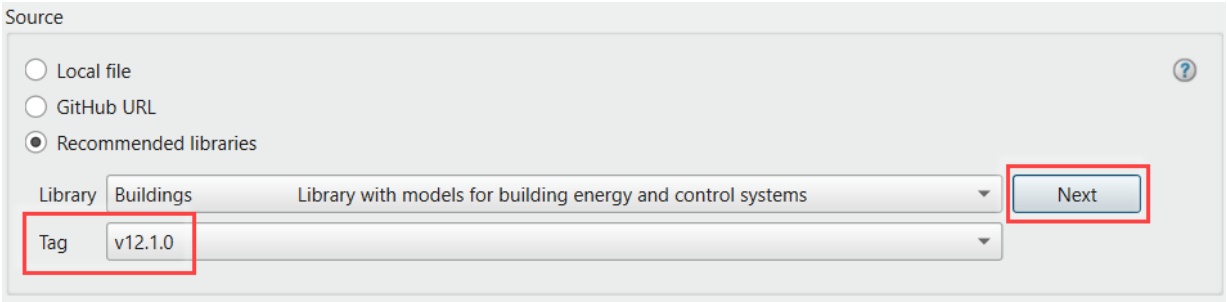
Use the command **Tools > Library Management**, and select the **Install** tab. Here you can now select **Recommended libraries**:



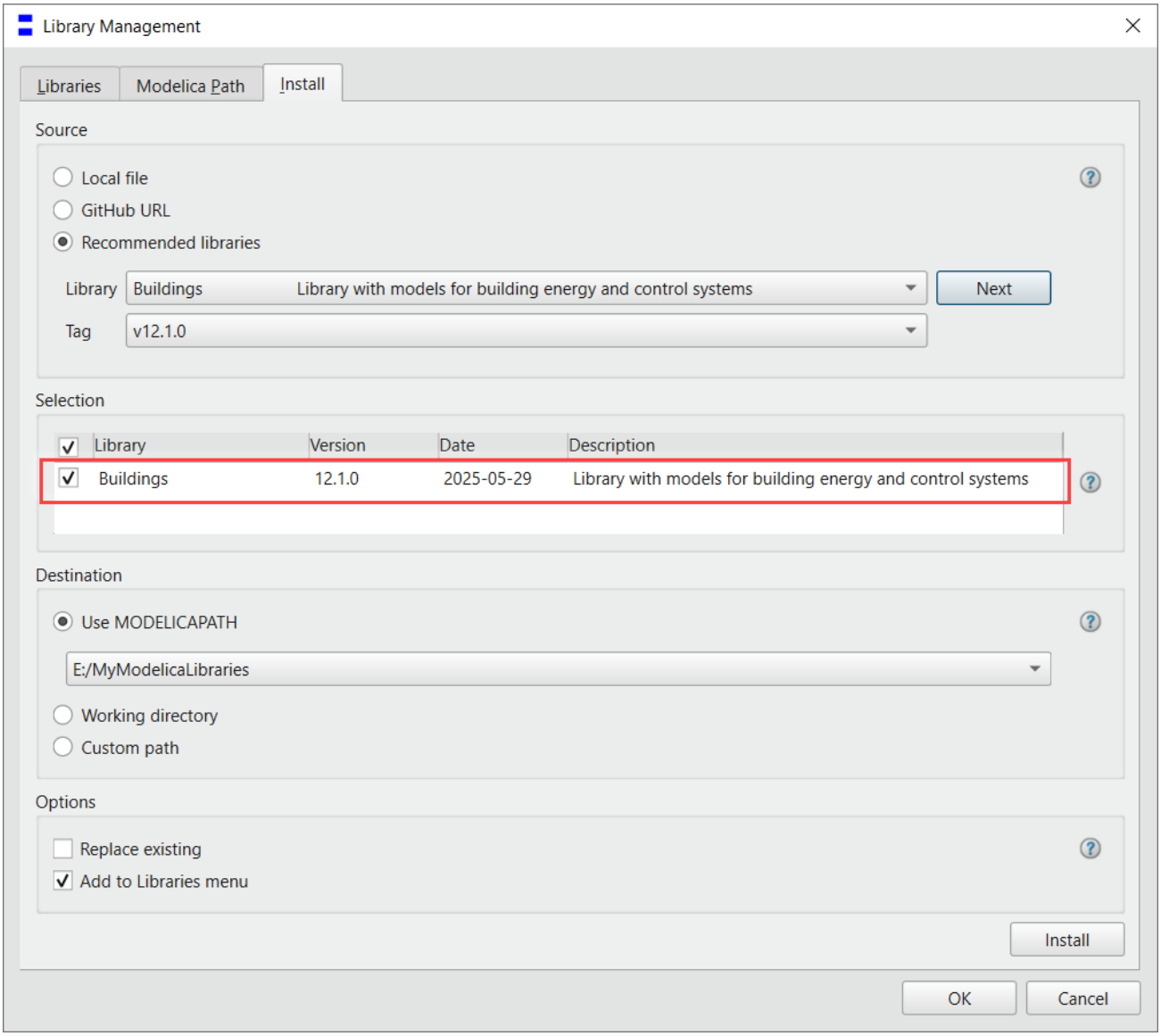
You can click the arrow in the **Library** field to get a list; you can use a tooltip to see the full description:



Let us download and install the default Buildings library. Clicking **Next** once will give the default version to download ("**Tag**"). (You can use the arrow for this field to select from possible versions to download.)

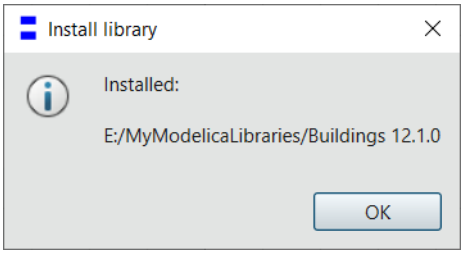


Now, click **Next** once more. The result is that the possible downloads of the selected item are presented in the **Selection** group (note that this information may take some time to fetch).



Select what you want to download and install from the **Selection** group and specify your selection for the **Destination** and **Options** group.

Now you can click **Install** to install your selection. When the selection is done, you will get a message:



Important! When you have clicked **OK** to close the message above, do not forget to click **OK** in the Library Management window to validate and close it. If you close it by clicking **Cancel** or the upper cross, the process is not finalized – you do not get the library in the **File > Libraries** list if you have selected so, as an example.

Working with the list of recommended libraries

For each Dymola version, there is a default list from Dassault Systèmes for recommended libraries. For Dymola 2026x, the file is located in the folder `Program Files\Dymola 2026x\insert`, and the file name is `modelicaLibraries.xml`.

You can create a similar file with other libraries and merge that with the default list, or replace it.

An example of a user-defined list using XML can be, for example, `MyRecommendedLibraries.xml`, with the following content:

```
<Libraries>
 <Library
  name="AixLib"
  description="A Modelica model library for building performance simulations"
  url="github.com/RWTH-EBC/AixLib"
  />
 <Library
  name="Physiolibrary"
  description="Modelica library for Physiology"
  url="file://<company-server>/VettedLibraries/Physiolibrary-3.0.0.zip"
  />
</Libraries>
```

To decide how the user-defined list is to be used, you can use two flags:

- To point to a user-defined list of libraries, you can use the flag `Advanced.LibraryManagement.Source.URL`. By default, the flag value is an empty string (""), meaning that the default list is used.
- To decide how to use the user-defined list of libraries, you can use the flag `Advanced.LibraryManagement.Source.MergeWithDefault`. The default value of the flag is `false`, meaning that the user-defined list replaces the default list. Setting the flag to `true` means that the user-defined list is merged with the default list.

So, to conclude the example, you can set

```
Advanced.LibraryManagement.Source.URL=
```

```
"E:/MyModelicaLibraries/MyRecommendedLibraries.xml"
```

Notes:

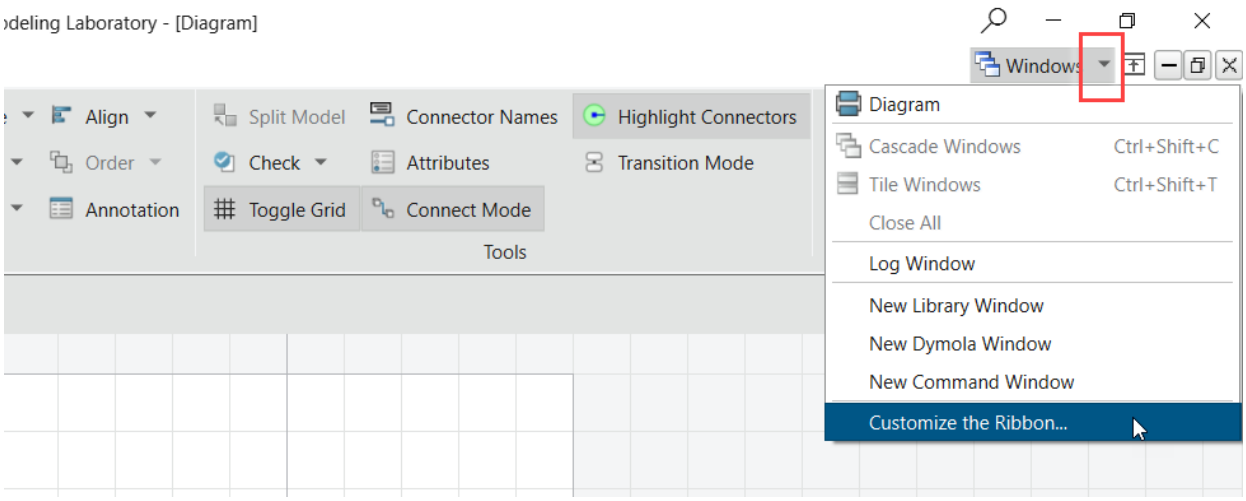
- Really, the URL should be "`file:///E:/....` but omitting `file:///` is supported.
- You must specify `<company-server>` in the file, of course, and have that library available there.

Keeping the default value of the flag `Advanced.LibraryManagement.Source.MergeWithDefault` means that your list will replace the default list.

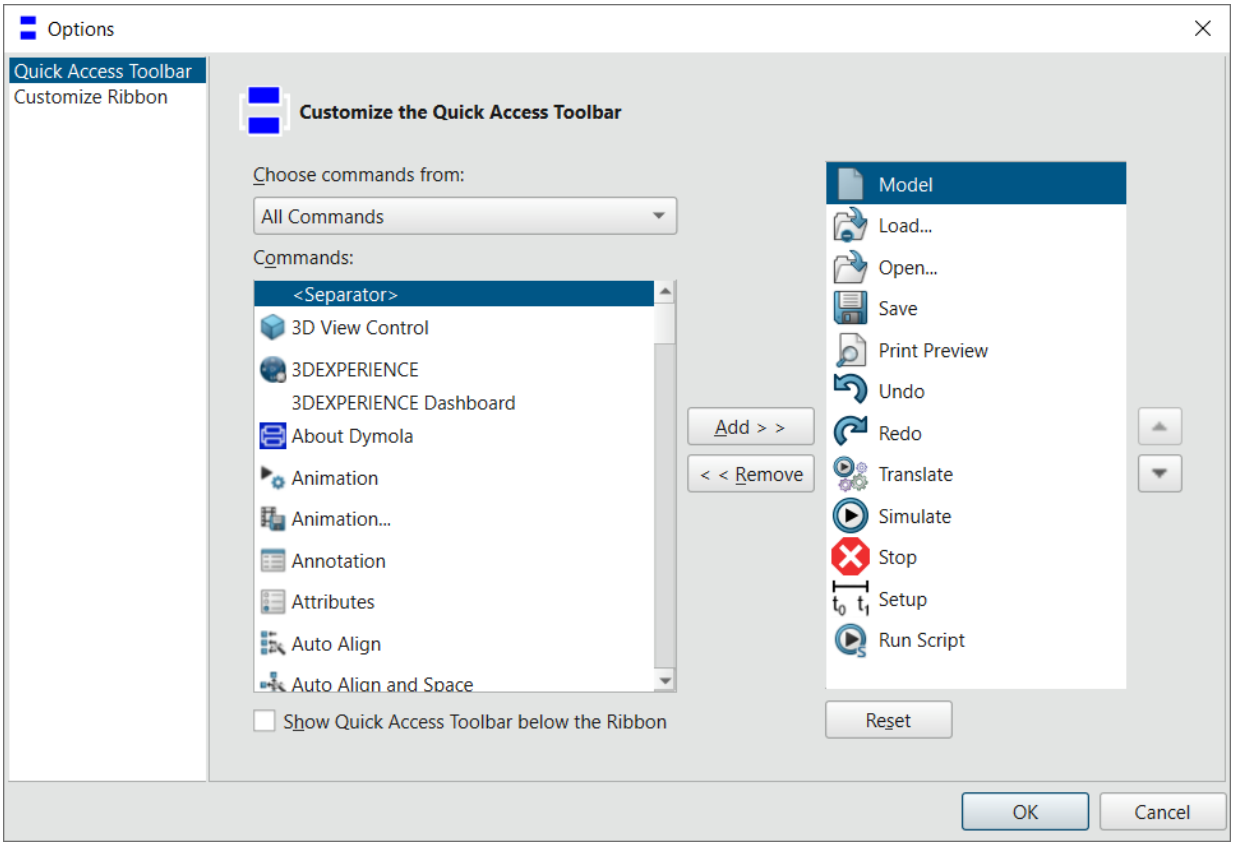
Customizing the quick access toolbar and the ribbon

The customization dialog

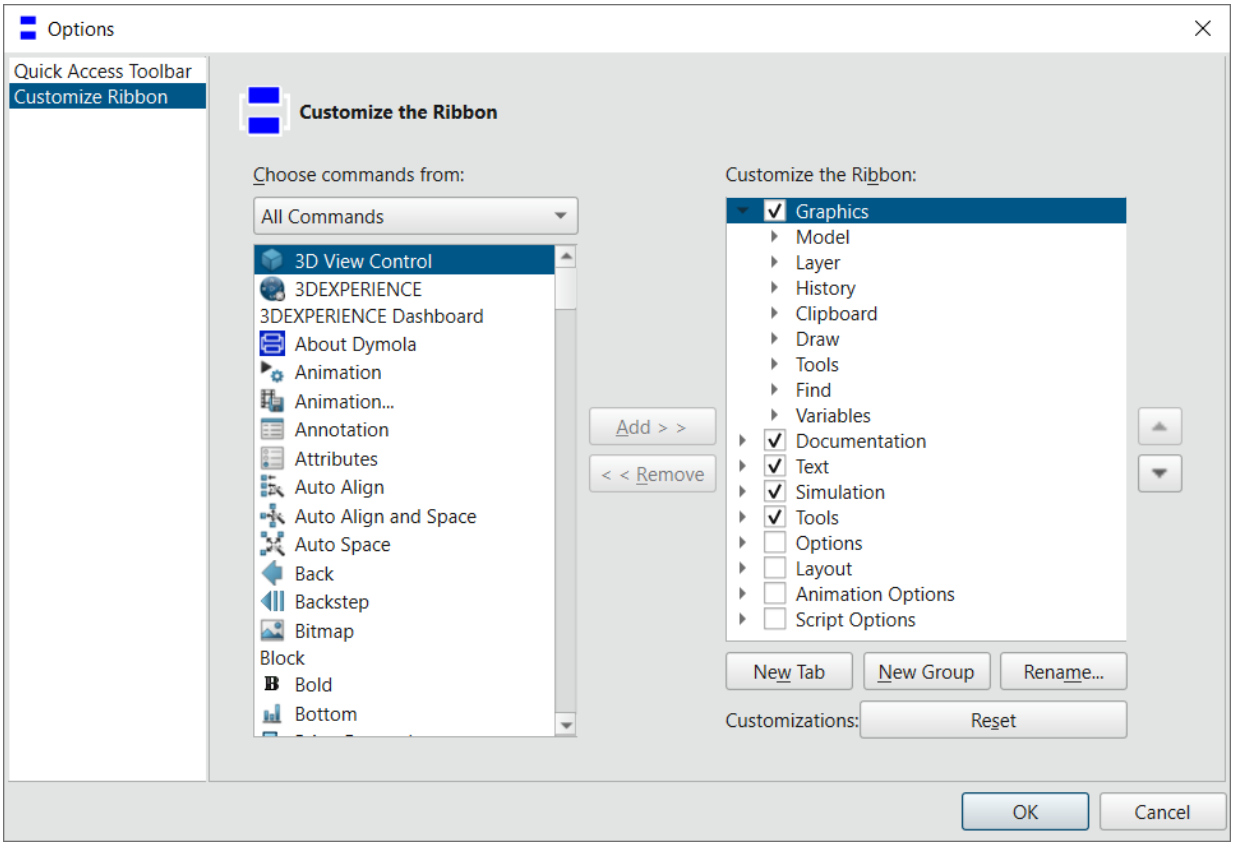
You can customize the quick access toolbar and the ribbon using the command **Customize the Ribbon...**. The command is reached by clicking the arrow after **Windows** in the upper right of the Dymola window:



Clicking the command gives the customization dialog for the quick access toolbar (default selection in the left pare):

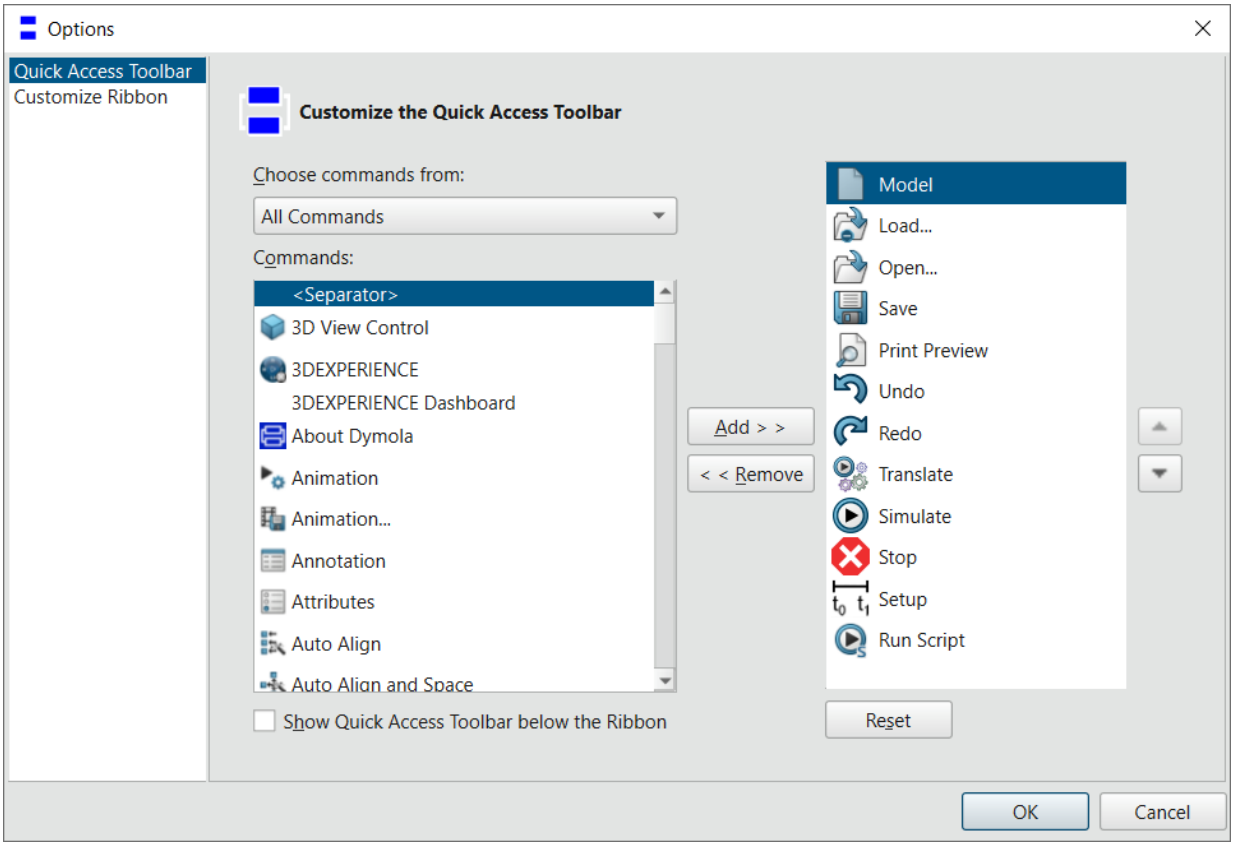


Clicking **Customize Ribbon** in the left pane gives the customization dialog for the ribbon:



Customizing the quick access toolbar

Starting with customizing the quick access toolbar:

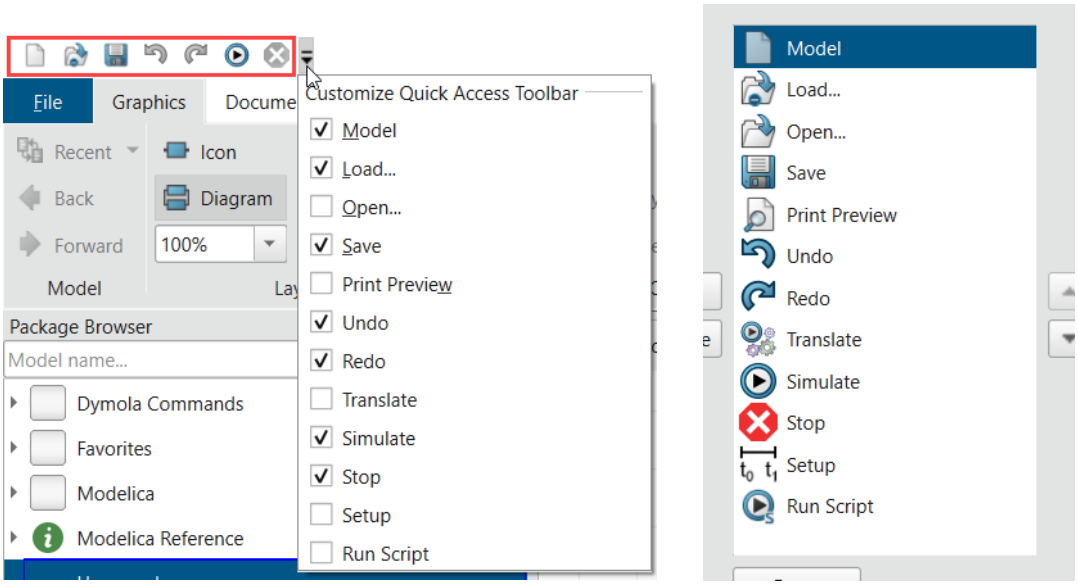


There are three parts in customizing the quick access toolbar:

- Selecting what commands that *can* be displayed, and the order
- Selecting which ones of the above commands that *are* displayed.
- Selecting the location of the quick access toolbar.

When displaying the customization dialog, it is by default displaying the customizing of the quick access toolbar – see the above figure, the left pane.

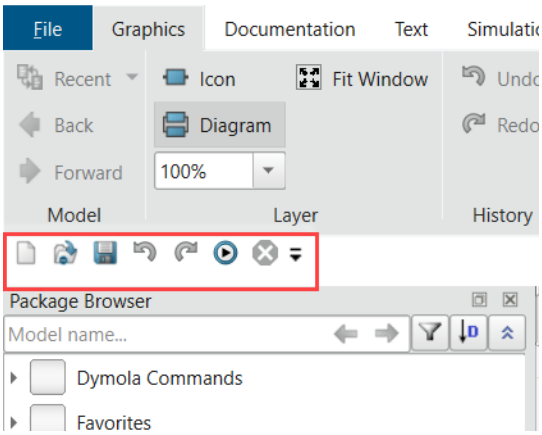
The pane to the right lists the command that *can* be displayed in the toolbar. Starting by looking at the default as in the figure above, and comparing with the actual toolbar, here you can use the arrow in the toolbar to specify which ones of the possible ones that are displayed (left figure below) – compare with the figure to the right, the right pane in the customization dialog.



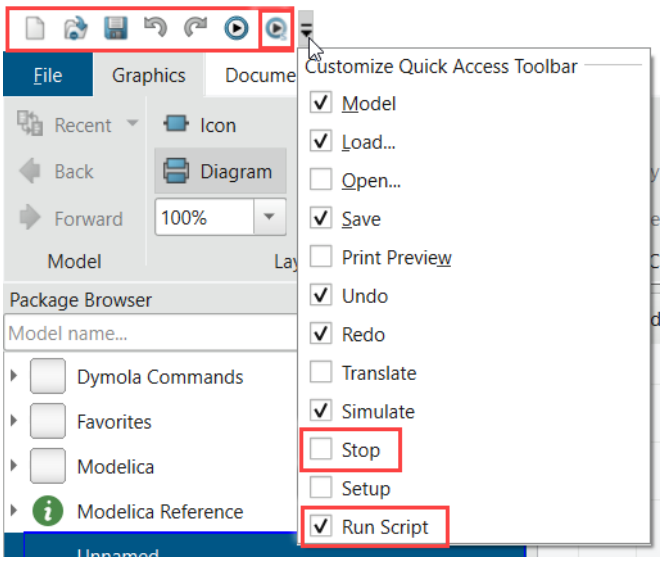
The last part of the default setup is the location of the quick access toolbar, in the upper left of the Dymola window, as displayed above.

Customizing, let us go the other way around, which is, looking at the location of the toolbar, then the selection of what commands to display, and finally the selection of possible commands.

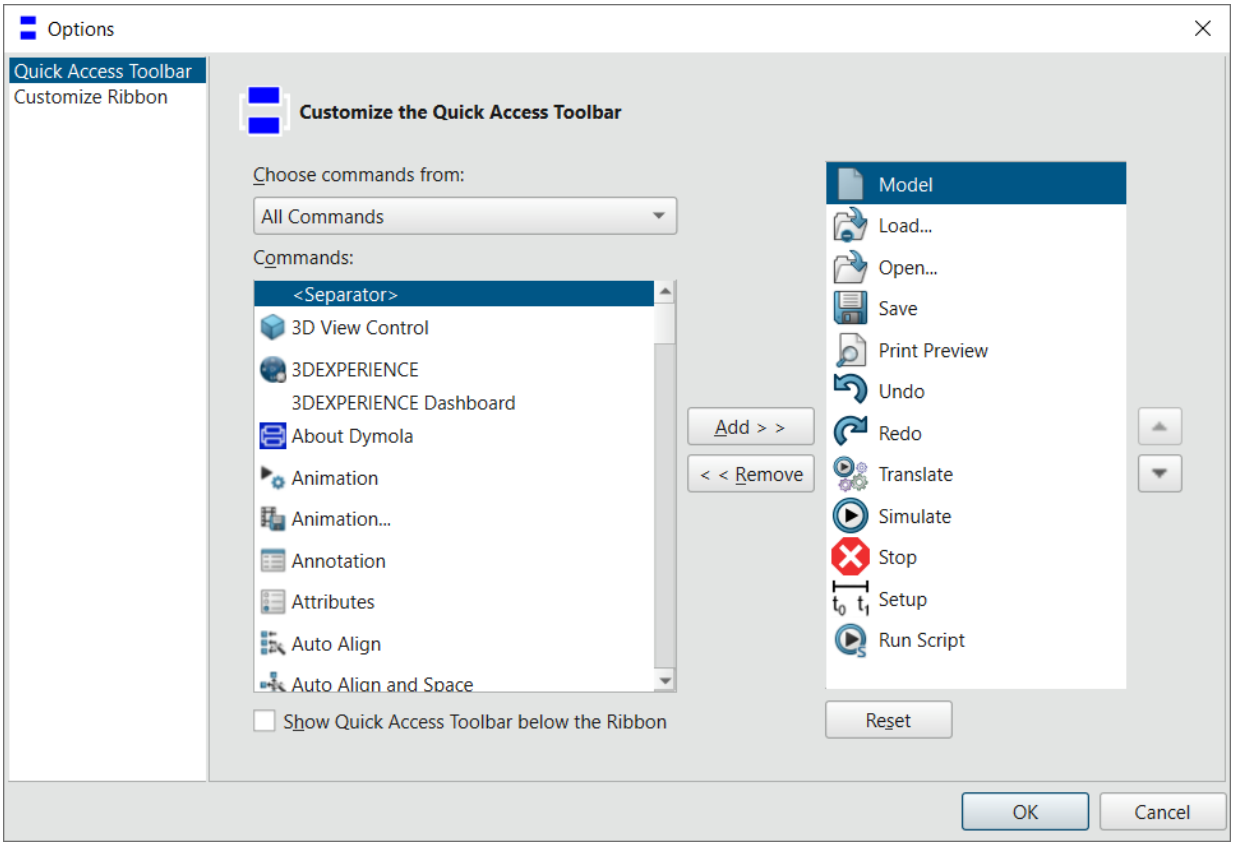
To change the location of the toolbar, activate **Show Quick Access Toolbar below the Ribbon** in the lower middle of the customization dialog, and click **OK**. The result is:



As an example of changing what commands to display, let us, in the menu for selecting what commands to display, disable **Stop** and activate **Run Script**. This will give (compare the toolbar in the figure above):



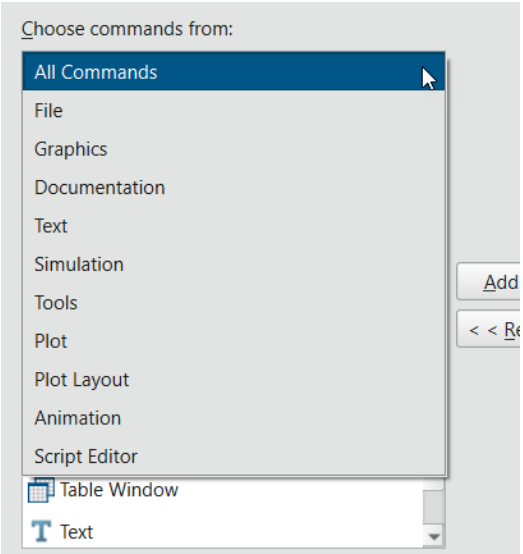
Finally, to decide the order and to select possible commands, we have to look at the customization dialog again:



You can change the order of the commands by selecting a command and use arrow up or arrow down to the right in the dialog.

To remove a command, select it and click **<<Remove**.

To add a command, you can start by selecting from where to add the command ("Choose commands from") in the middle of the dialog. The default is "All Commands", the possible selections are:



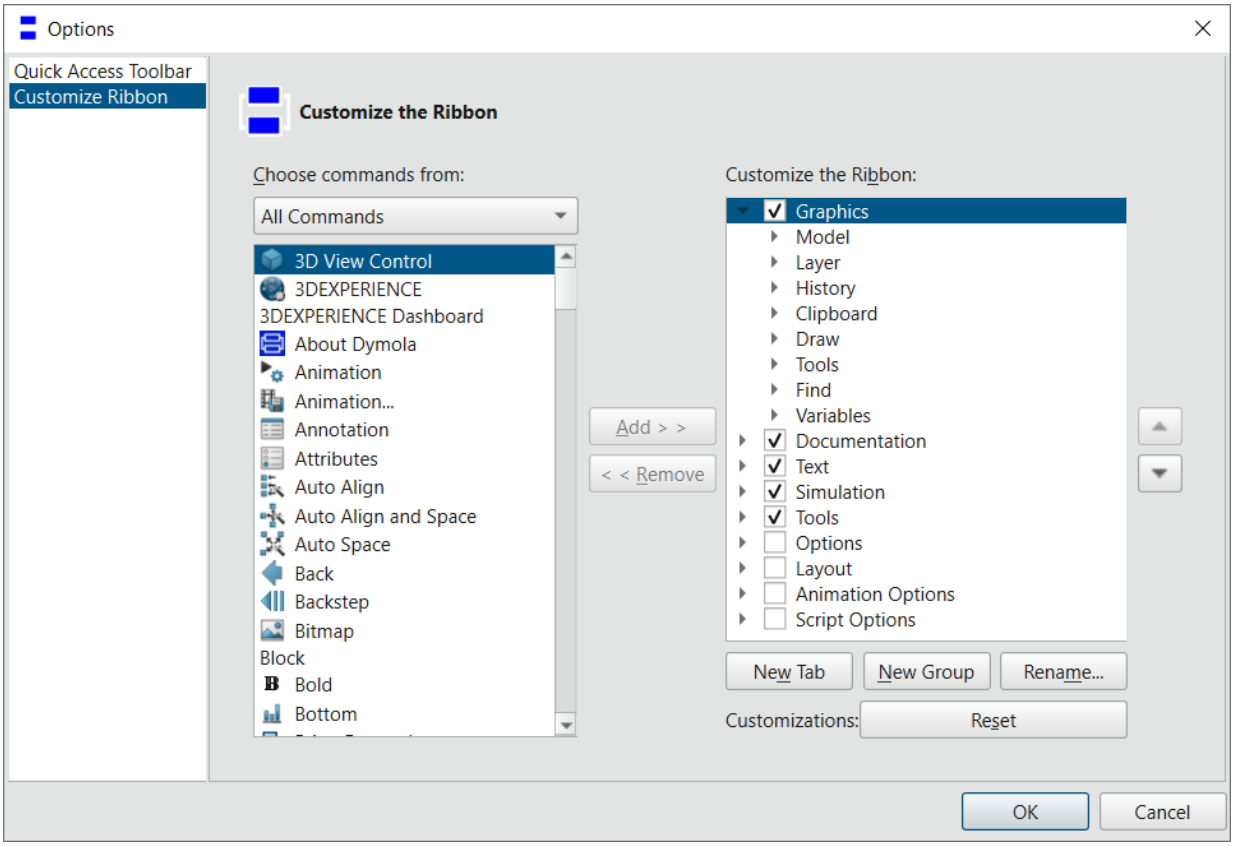
Once having found the command to add, do the following, in order:

- In the pane *to the right*, select the command that should precede the command to add (for skipping the sorting)
- In the middle pane, select the command to add, and click **Add >>**.

Note: You can reset the sorting and removal/additions of commands by clicking **Reset** under the pane to the right.

Customizing the ribbon

Compared to customizing the quick access toolbar, the customization of the ribbon is more complex, you have a number of options here:



Some ideas of basic usage:

- You can add new tabs with new groups. This gives you full flexibility to create your own command interface, more or less.
- You can add new groups in existing tabs, to have access to favorite commands in several tabs if you want to.
- You can remove existing groups from existing tabs, if you think some groups clutter rather than add flexibility.
- You can only add or remove commands from custom groups, that is, groups that you have created yourself.

Below a simple description how to do the above. The descriptions assume you have displayed the customization dialog above.

- To add a new tab:
    - Select an existing tab in the pane to the right and click **New Tab**. In the pane, you will get a new tab with the name **New Page**, after the tab you selected, with a group **New Group**.

    - In the pane, click **New Page**, and then click **Rename…**. Change the name.
    - Do the same for the New Group, to get the wanted name for the new group.
    - Click **OK** to implement your new tab in the ribbon.
- To add a new group to an existing tab:
    - Expand the tab and select a group in the pane to the right, and then click **New Group**. You will get a new group with the name **New Group**, after the group you selected.
    - Click **Rename…** and change the name of the group.
    - Click **OK** to implement your new group.
- To add a command to a custom group:
    - Select the group in the pane to the right, and if you have commands, select a command.
    - In the middle pane, select from where to fetch the command (the default is (the default is **All Commands**), and then select a command and click **Add >>**. The command is inserted after the command selected.
- To remove a group:
    - Select the group in the pane to the right, and click **<< Remove**. Note that you can select and remove any group, not only custom ones.

Notes:

- You can move tabs, groups, and commands by selecting the item and use the arrow up/down in the right of the dialog.
- You can reset all customization of the ribbon by clicking **Reset**.

Saving the customization, starting Dymola without the customization, etc.

The customization of the quick access toolbar and the ribbon is saved in the file `ribbon.dymx` that is located in the same folder as the setup file `setup.dymx`. Note however that the location of the quick access bar is stored in `setup.dymx`.

Like for `setup.dymx`, each Dymola version has its own `ribbon.dymx`.

It is possible to start Dymola without applying the customization of the quick access bar and the ribbon. You do that by using the command line option **–noribbonconfig**. Note that location of the quick access bar is not affected by this command line option since this setting is stored in `setup.dymx`.

Use the command line option **–ribbonsetup** to specify the path to the file `ribbon.dymx`, if needed.

To avoid saving changes in the customization setup, you can start Dymola wilt the command line option **–nosavesettings**. This option affects `setup.dymx`, `startup.mos`, and `ribbon.dymx`. The settings are loaded, but any changes of the settings that you do in Dymola will not be saved.

To prevent *both* reading and saving of the above settings, you can start Dymola with the command line option **–nosettings**.

If you use the command **Windows > New Dymola Window** to open a new Dymola window, the customization of the quick access bar and the ribbon is not applied. Note however that the location of the quick access bar is applied in this case as well.

3.4.2 Installation on Windows

Reading Windows registry keys

You can use the new built-in function `getRegistryValue` to read Windows registry keys. This can be used to, for example, to execute SciLab in a portable way without using external scripts.

```
getRegistryValue(key, name="", convert=true, defaultValue="")
```

The function returns the value of system registry key/name pair.

You use the input string argument `key` to specify the full path of the registry key, and the input string argument `name` to specify the name of the registry value. For `name`, the default is an empty string.

The input argument `convert` is by default `true`, meaning that Dymola performs conversion of delimiters, that is, forward slash in the key is converted to backslash, and the reverse is performed on the output.

If the key/name is not found, the string specified by `defaultValue` is returned.

Two examples of success:

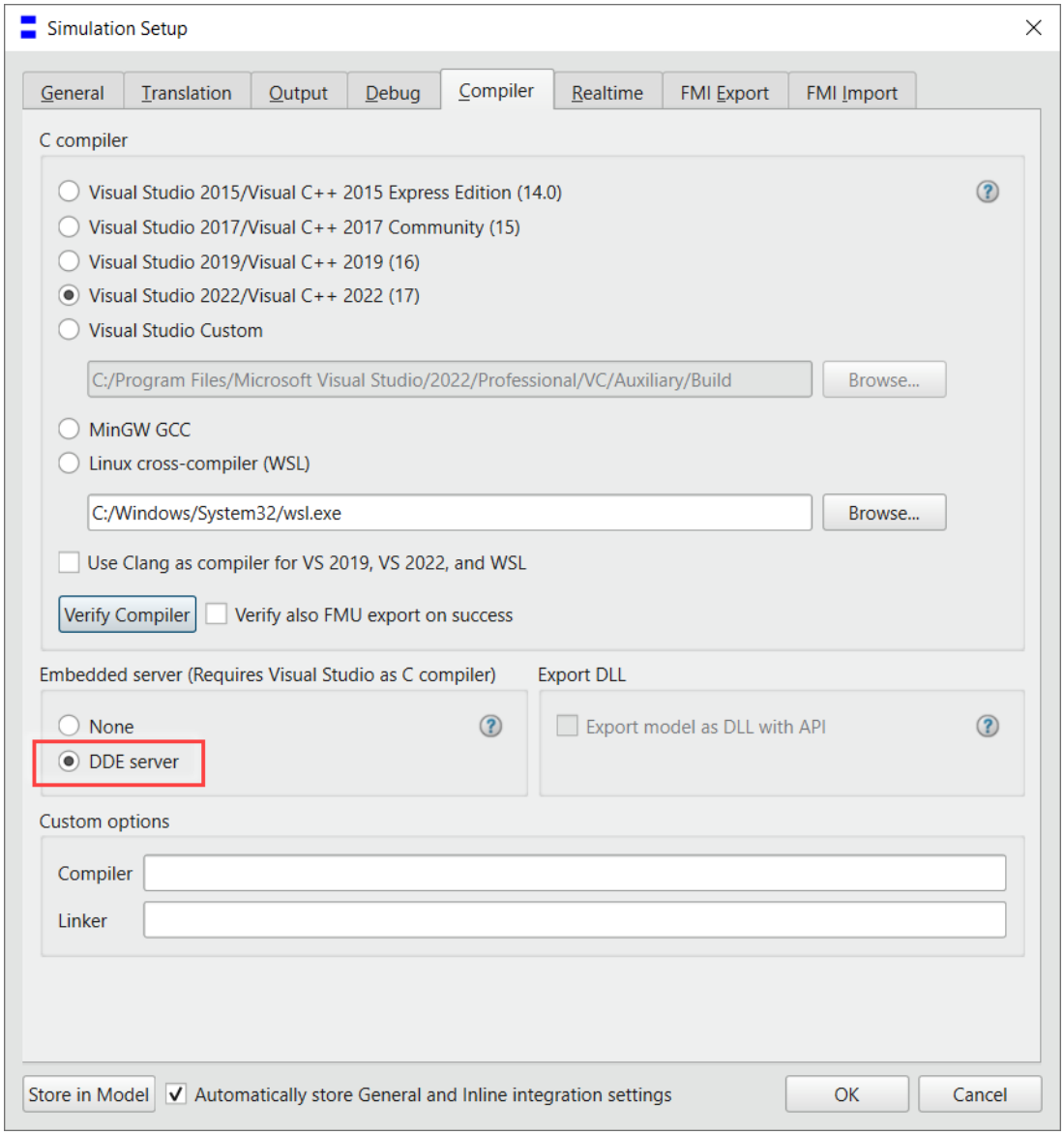
```
getRegistryValue("HKEY_LOCAL_MACHINE\\SOFTWARE\\Dassault
Systemes\\Dymola 2025x Refresh 1", "InstallDir", false)
 = "E:\\Program Files\\Dymola 2025x Refresh 1\\"

getRegistryValue("HKEY_LOCAL_MACHINE/SOFTWARE/Dassault
Systemes/Dymola 2025x Refresh 1", "InstallDir")
 = "E:/Program Files/Dymola 2025x Refresh 1/"
```

One example of failure (for a yet non-existing Dymola version – note that in this call, the user specified the argument `defaultValue`.):

```
getRegistryValue("HKEY_LOCAL_MACHINE/SOFTWARE/Dassault
Systemes/Dymola 2032x Refresh 1", "InstallDir", true, "Not
installed")
 = "Not installed"
```

Discontinued support for 32-bit embedded DDE server

If you select a Visual Studio compiler, you can also activate to have an embedded DDE server included: (You reach this dialog by the command **Simulation > Setup**, the **Compiler** tab)



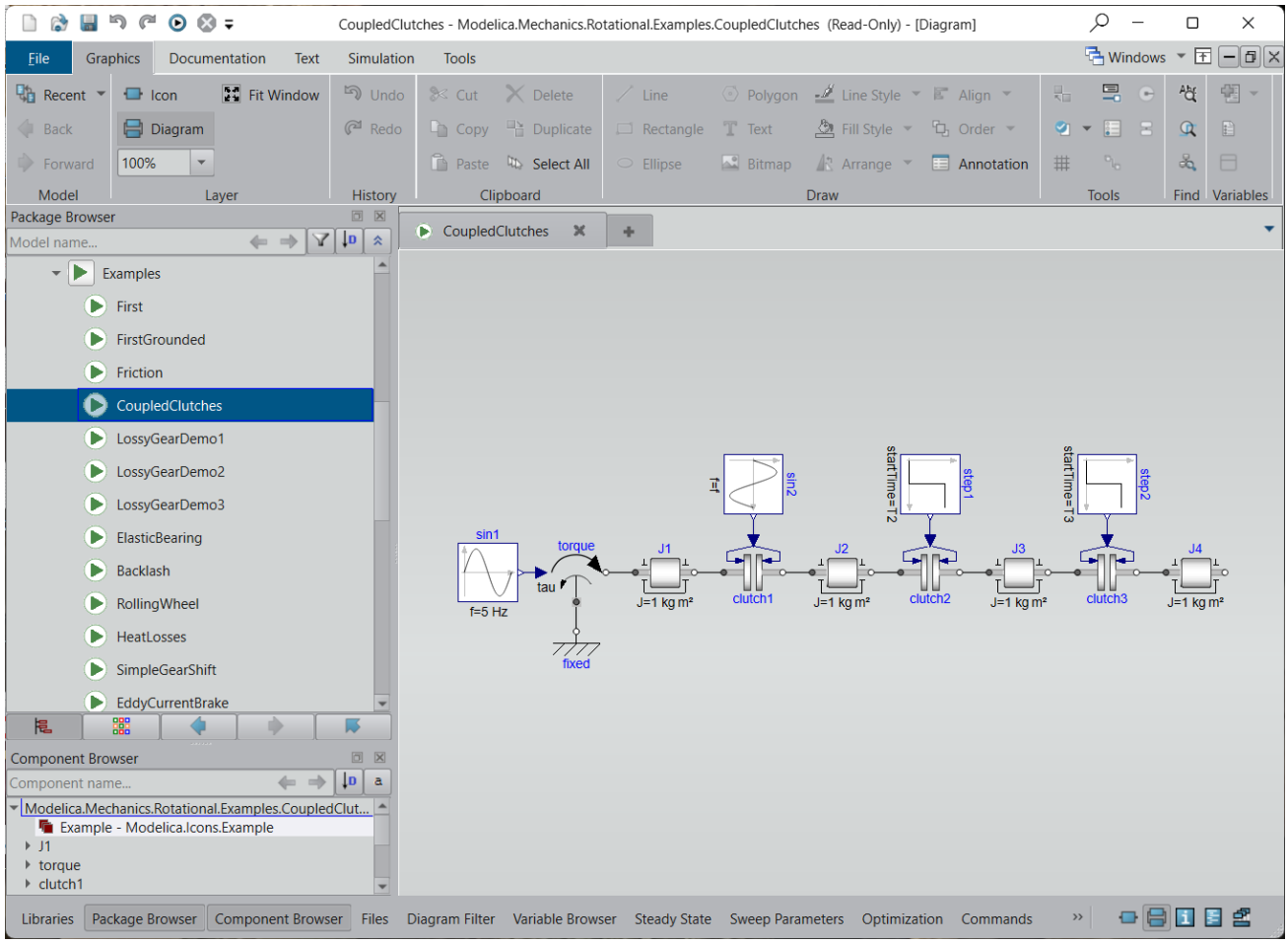
In Dymola 2026x, you only get a 64-bit embedded DDE server when you select this option.

Support for MinGW GCC 32-bit compiler deprecated, support to be discontinued from next version

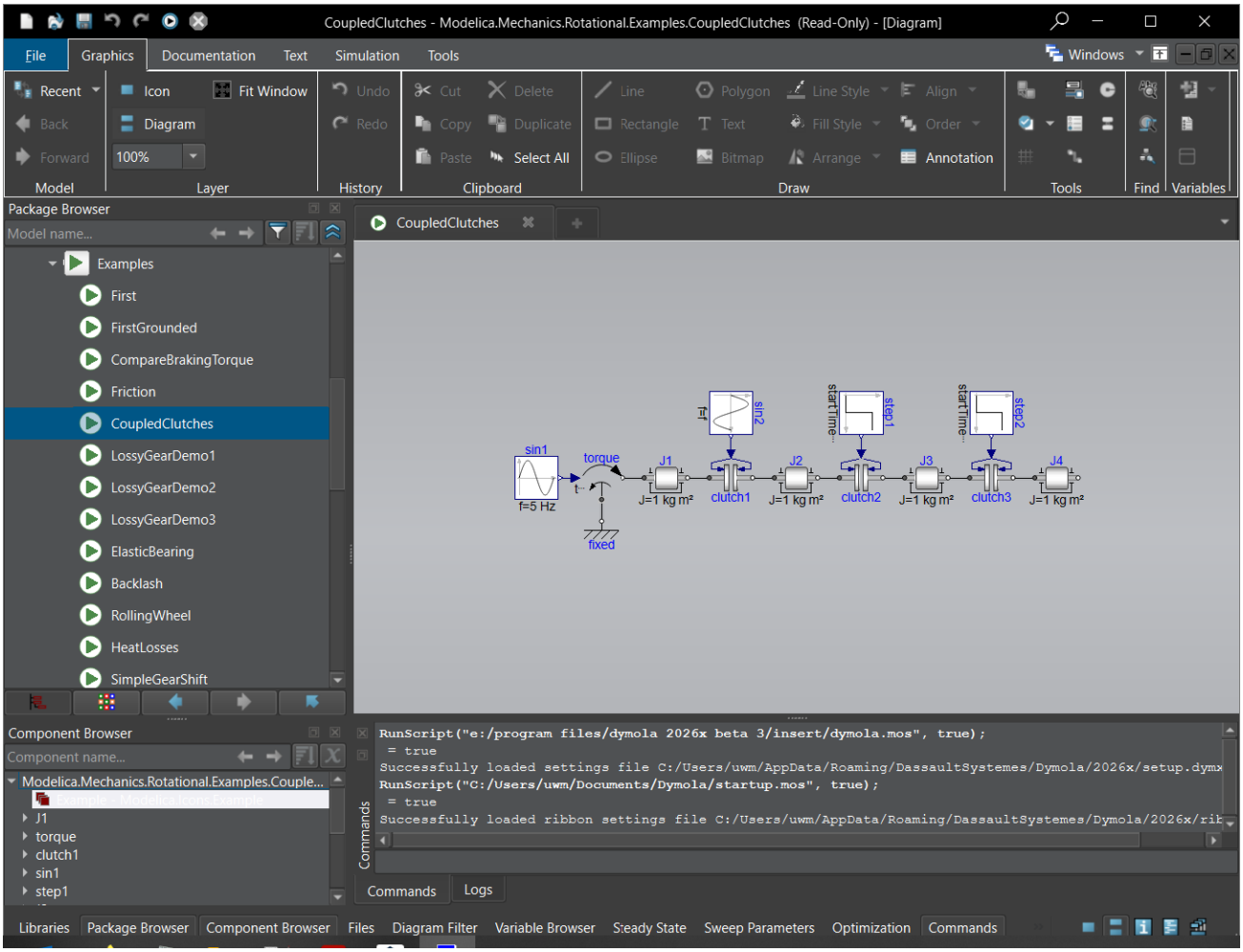
The support for the MinGW 32-bit GCC compiler is deprecated in this version, Dymola 2026x. The support will be discontinued in Dymola 2026x Refresh 1. (64-bit MinGW GCC compiler is however still supported.)

Changed color theme for dark mode

The colors used for dark mode has been changed. An example from the previous version Dymola 2025x Refresh 1:



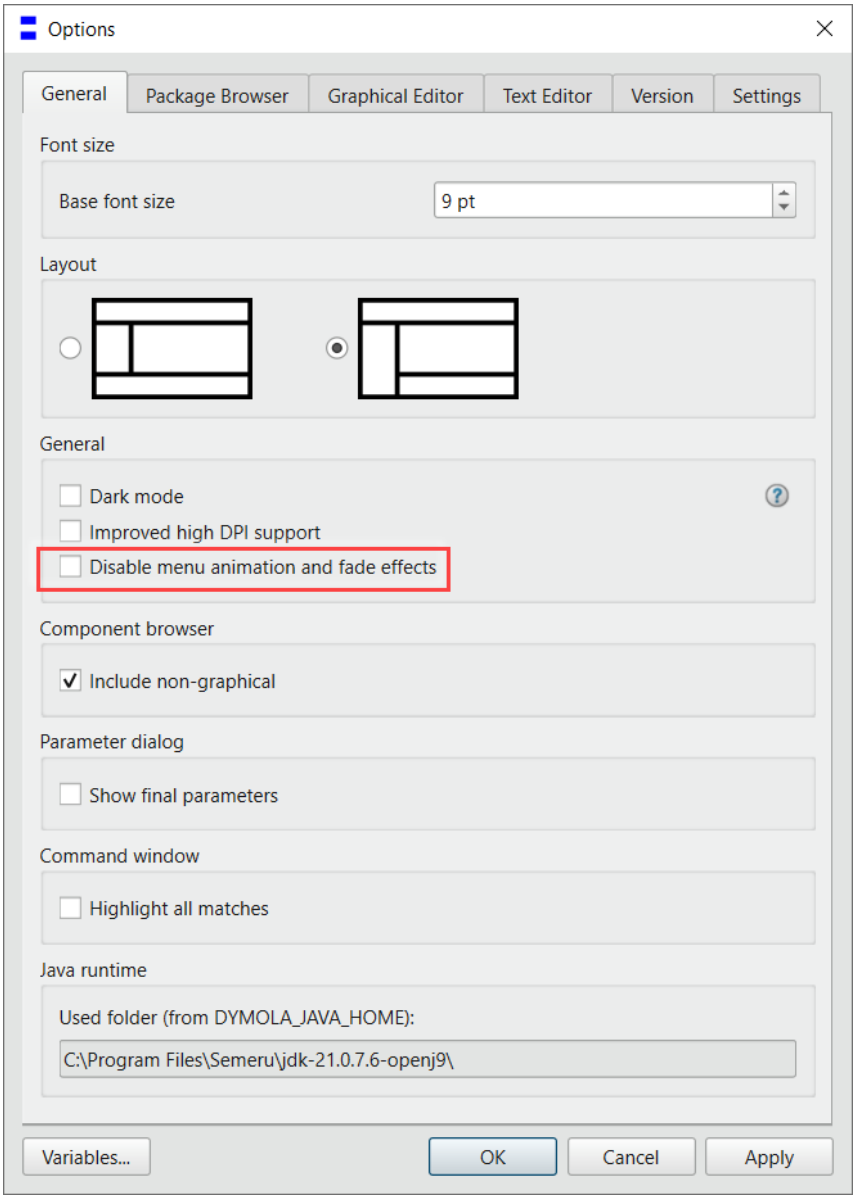
The same example in Dymola 2026x:



Handling issue with delayed context menu submenus

Some cases where the displaying of context menu submenus is delayed have been seen, possibly related to using Windows 11 24H2. If you encounter this issue, you can start Dymola with the command line option `-noeffects`. This disables menu animation and fade effects in Dymola, which seems to cause the problem.

This setting is also available using the **Tools > Options** command, the **General** tab:



3.4.3 Dymola license server on Windows and Linux

Updated DSLS (Dassault Systèmes License Server) version

Dymola 2026x supports DSLS 2026x. Earlier DSLS versions cannot be used. Note that you can use DSLS 2026x also with older versions of Dymola than 2026x.

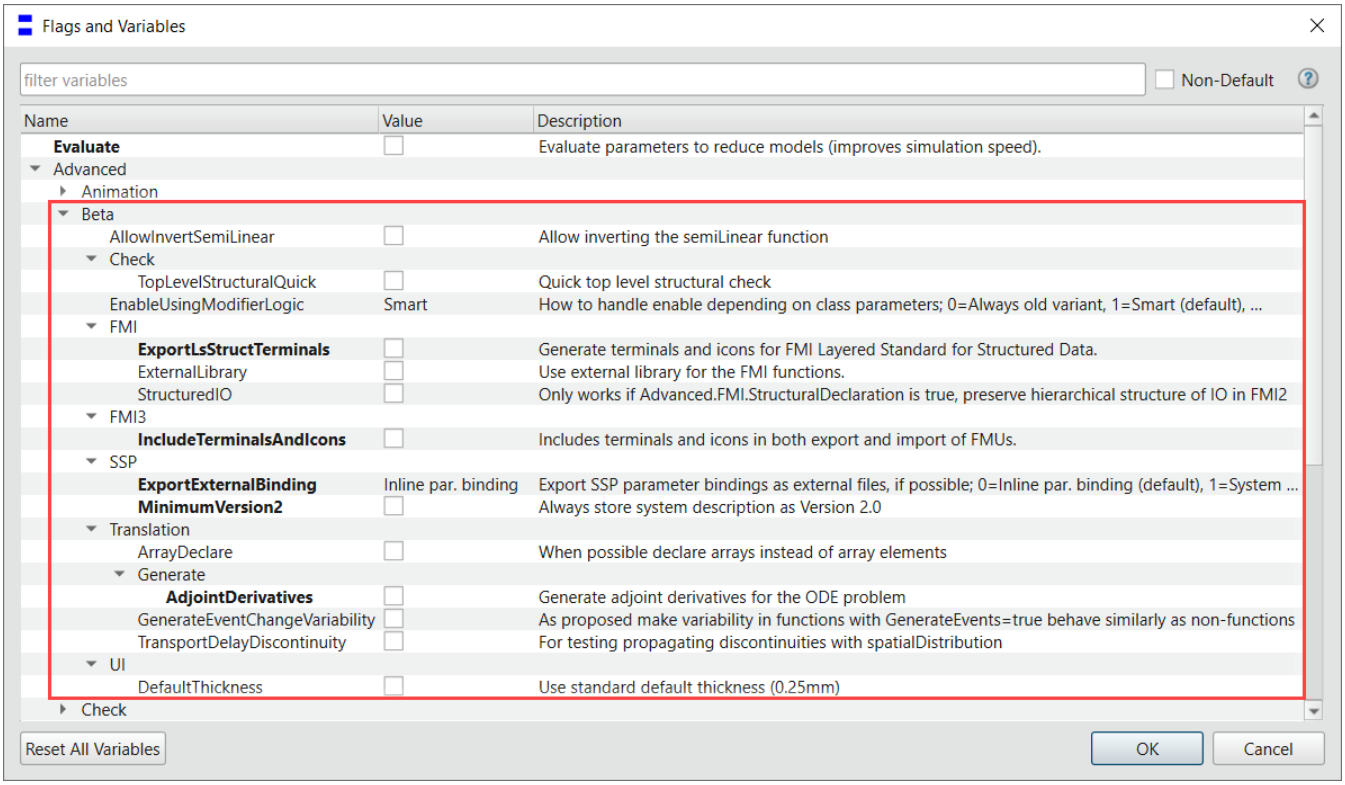
3.5 Features under Development

In this section you will find features that are “under development”, that is, they are not finalized, nor fully supported and documented, but will be when they are formally released in a later Dymola version. You may see this as a “technology preview”.

Note that they are only documented here in the Release Notes until they are finally released, then they are documented in the manuals, and also once more in the Release Notes, but then in the corresponding feature section. The text “(*This feature was a beta feature in the previous Dymola release, with the default value of false for the corresponding Beta flag.*)” is also added (the default value might be some other value).

In the end of each description, it is noted in which Dymola release the feature appeared.

The beta features that are put on flags are grouped by `Advanced.Beta` flags in **Tools > Options > Variables…**: An example from Dymola 2026x:



The features are by default not activated, to activate any of them, activate the corresponding flag.

When the features have been released, the name of the flag is changed.

In Dymola 2026x, the following “under development” features are available:

Dymola Modelica Compiler tool (not on flag)

Overview

There has been a long-standing wish to better support scripting and remote execution in Dymola. Typical use-cases include Continuous integration toolchains, optimization, deployment on computing clusters, and scripting from command scripts, or e.g. Python. In each of these cases, the normal Dymola user interface is more of a problem than a benefit, even if it can be hidden with the option `-nowindow`.

Note that also means that DMC is useful where there is no monitor available, such as headless servers or containers and virtual machines without a graphical console.

Starting in the Dymola 2025x release, we provide a minimalistic version of Dymola, the Dymola Modelica Compiler (DMC), which is a command line tool without any graphical user interface at all. In short, DMC is designed to:

- Translate and simulate Modelica models.
- Run mos-scripts.
- Accept commands on a network port, to support e.g. Python integration.

Operation

The Dymola Modelica Compiler is located in the `dymola\bin64` directory, and is started as:

```
dmc.exe
```

It supports the following command line arguments:

Option	Description
-h	Prints a summary of available operations.
-o *file-name*	Opens the named Modelica file into DMC.
-x *command*	Runs the given command line. For example: Dos: `dmc -x "simuateModel(""Foo"", stopTime=2);"` Linux: `dmc -x 'simulateModel("Foo", stopTime=2);'` (For Linux, the shell-quoting may differ between Linux versions.)
-r *file-name*	Runs a script file (.mos)
-p *port*	Starts the HTTP server on the designated port. This option is needed to support e.g. Python integration.

Constraints

Due to the nature of the command line tool, there are several constraints:

- There is no user interface to set up the license server, the C compiler, etc. Instead, DMC will read the Dymola startup script to get the initial setup.
- Commands that are graphical in nature are not implemented, for example, anything related to plotting and animation. Likewise, operations that use the graphical representation of the model, such as `exportDiagram()`.
- DMC uses a Dymola license and licenses for optional products, such as libraries. To ensure uninterrupted execution of a sequence of DMC commands, the license will remain checked out for a short period after the last DMC execution.
- DMC is in Beta-state for this Dymola 2026x release.

Using DMC with Python

When instantiating the Python interface, you can now specify if Dymola standalone should be used or the new Dymola Modelica Compiler (DMC).

There is a new argument to the constructor of DymolaInterface called `kernel`. When set to `True`, DMC is used. The Python interface will look for the executable file `dmc.exe` in the installation folder. The default value is `False`, which starts standalone Dymola.

```
dymola = DymolaInterface(kernel=True)
```

Note 1, `kernel=True` needs to be set even when providing the path to the DMC executable.

```
dymola = DymolaInterface("C:/Program Files/Dymola
2026x/bin64/dmc.exe", kernel=True)
```

Note 2. On Linux, you need to call the start script for DMC to get the correct environment variable setting. There are alternatives:

- For the latest installed Dymola:

  ```
  dymola = DymolaInterface('/usr/local/bin/dmc', kernel=True)
  ```

- For Dymola 2026x:

  ```
  dymola = DymolaInterface(
  'opt/dymola-2026x-x86_64/bin64/dmc.sh', kernel=True)
  ```

(This feature appeared in Dymola 2025x.)

Allowing inverting semiLinear calls

Inverting semiLinear calls might improve index reduction. To test it, set the flag `Advanced.Beta.AllowInvertSemiLinear = true`. (The flag is by default `false`.)

(*This feature appeared in Dymola 2024x Refresh 1.*)

Smarter top-level structural check

A new beta-feature is that the top-level structural check can be restricted to only check the top-level model, ignoring sub-components, by setting the flag

`Advanced.Beta.Check.TopLevelStructuralQuick=true` (in addition to `Advanced.Check.TopLevelStructural=true`). This is currently faster than regular check, and still detects relevant issues at the top. (Both flags are by default `false`.)

The check will fail with an inconclusive result if the model relies on variables from sub-components.

There are two limitations that will be improved for the future:

- It is not possible to proceed from this check to a normal check including sub-components. Translate or disabling `Advanced.Check.TopLevelStructural` works as normal).
- Even if faster than a regular check, it could be even faster.

(Note that outside this beta feature, the top-level structural check has been improved in general in Dymola 2025x.)

(This feature appeared in Dymola 2025x.)

Better handling of enabling the editing of a parameter in the parameter dialog by another parameter

Previously, some cases of enabling the edition of a parameter in the parameter dialog by another class parameter did not work. The new flag `Advanced.Beta.EnableUsingModifierLogic` can be used in those cases. The value of the flag can be any of:

- 0 – No change of behavior compared to older Dymola versions.
- 1 – Smart handling, enable new logic when parameters depend on parameters. This is the default value of the flag.
- 2 – Always use the new handling of parameters.

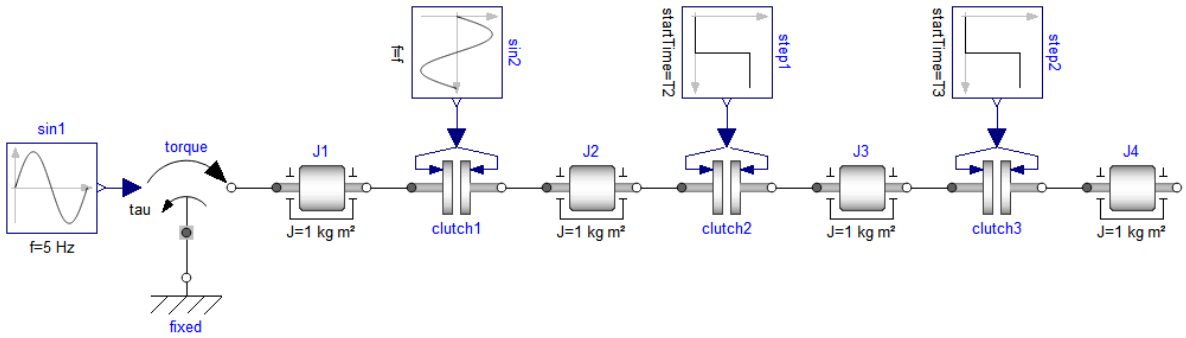
(*This feature appeared in Dymola 2024x Refresh 1.*)

FMI 3: Support for terminals and icons in general

FMI 3.0 introduces terminals and icons. This feature is now being implemented in Dymola to support exporting and importing Modelica components with connectors for FMI Model Exchange.

The idea for the export is to expose connectors that are unconnected in the Modelica model, i.e. the variables inside it, their causality, as well as the graphical data of the connectors. For a Modelica tool, this lets the component be imported with declarations of the connectors along with their graphical aspects.

For an example of an FMU to export and import, take the **Inertia** component in `Modelica.Mechanics.Rotational.Components`. An example model containing this component is **CoupledClutches**;

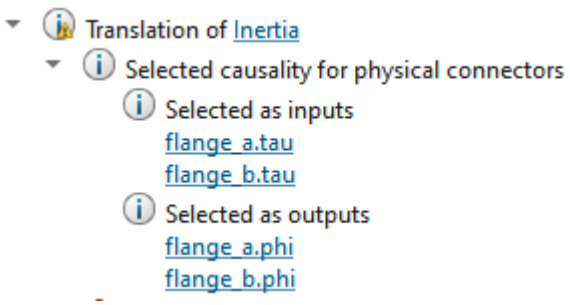


Open an **Inertia** component (such as **J1**), and set the flags;

```
Advanced.Beta.FMI3.IncludeTerminalsAndIcons = true
Advanced.FMI.AllowPhysicalConnectors = true;
Advanced.ExtendedStructuralCheck = false;
Advanced.ExtendedStructuralDiagnosis = false;
```

and export the **Inertia** as a Model Exchange FMU. With terminals and icons support, the FMU will contain information on the **Inertia**'s ports as well as graphical representation. A directory `terminalsAndIcons`, with an additional XML-file `terminalsAndIcons.xml` along with the icons from the model is added to the FMU.

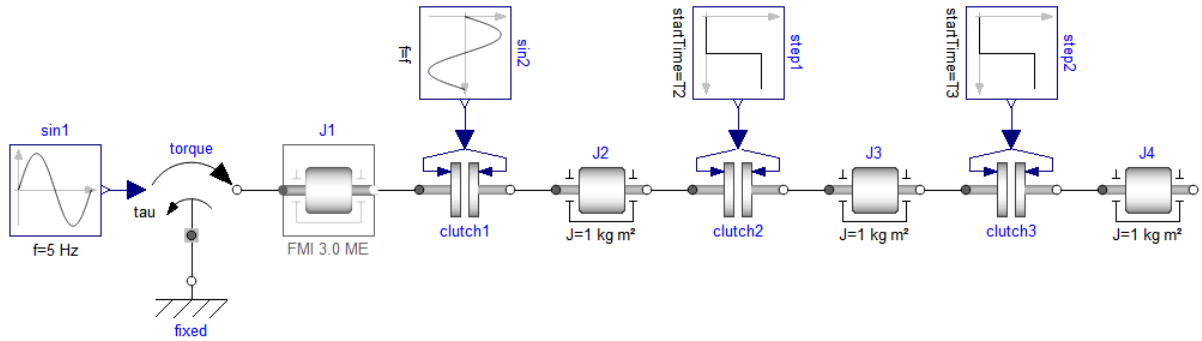
Note that, with the flag `Advanced.FMI.AllowPhysicalConnectors = true`, Dymola tries to determine the causality for the physical connections on its own:



But the "correct" causality depending on the context might need to be explicitly set by the user. For example by modifying the variables in the **Flange**:

```
connector Flange "One-dimensional rotational flange"

  output SI.Angle phi "Absolute rotation angle of flange";
  flow input SI.Torque tau "Cut torque in the flange";
  a
end Flange;
```

Now, you can import the **Inertia** FMU back into Dymola, exchange **J1** in **CoupledClutches** with the FMU, and finally connect all ports again.



Current limitations:

- This implementation focuses on ME-FMUs. Co-sim FMUs will in this current state not work in general.
- Black box is not compatible with this implementation of terminals and icons
- Import of the following is not yet supported:
    - Buses (expandable connectors) in the `terminalsAndIcons.xml`
    - Connectors/terminals containing stream variables

(*This feature appeared in Dymola 2023x Refresh 1, and was continued in Dymola 2026x.*)

FMI 3: Support for terminals and icons for FMI Layered Standard for Structured Data

A short description of the new standard "FMI Layered Standard for Structured Data" is: "Based on FMI 3.0, this layered standard defines how variables (especially parameters) of an FMU can be structured and grouped in a more flexible way than with the "structured naming convention" of the FMI Standard. The first version of this layered standard is focused on the definition of sampled maps." For more about the standard, see https://github.com/modelica/fmi-ls-struct/blob/main/docs/index.adoc.

Dymola 2026x supports export of FMUs with terminals and icons for this new standard. To activate the FMU export of terminals and icons for this standard, you can set the flag:

```
Advanced.Beta.FMI.ExportLsStructTerminals = true
```

(The flag is by default `false`. The flag is saved between sessions.)

Note that the standard is not yet finalized. For this reason, the flag is a beta flag.

(*This feature appeared in Dymola 2026x.*)

Option to use default thickness according to specification

There is a conclusion in the Modelica Association that the default line thickness should always be 0.25. In Dymola, the default line thickness depends on the context. To adapt to the standard, you can set the flag `Advanced.Beta.UI.DefaultThickness = true`. (The flag is by default `false`.)

(This feature appeared in Dymola 2025x.)

FMI 2: Option to preserve the hierarchical structure of IO in FMI 2

When importing an FMU, you can preserve the hierarchical structure of the IO by setting the flag:

```
Advanced.Beta.FMI.StructuredIO = true
```

(The flag is by default `false`.)

Notes:

- The flag `Advanced.FMI.StructuralDeclaration` must be `true` for the above to work.
- The flag is only working for FMUs of FMI version 2.

As example of the effect, consider a hierarchical connector

```
A.b
A.c
```

If the flag `Advanced.Beta.FMI.StructuredIO` is `false`, you will have two inputs when you import the FMU:

```
Modelica.Blocks.Interfaces.RealInput A_b
Modelica.Blocks.Interfaces.RealInput A_c
```

If you set the flag to `true`, and reimport the FMU, you instead get:

```
connector A_con
  input b
  input c
end Acon;
A_con a;
```

(This feature appeared in Dymola 2025x Refresh 1.)

FMI 2 and FMI 3: Option to use a standardized package of FMI functions when importing FMUs

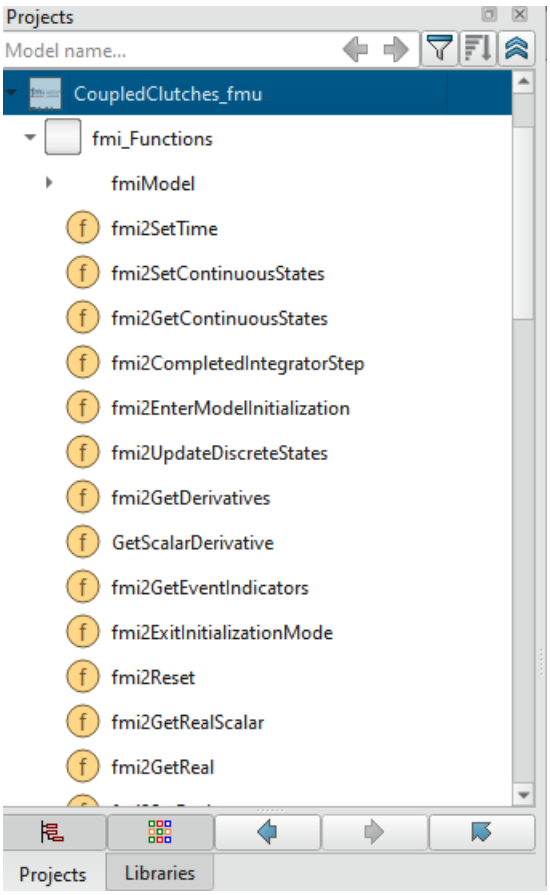
By default, each imported FMU produces fresh C-code. If you set the flag

```
Advanced.Beta.FMI.ExternalLibrary = true
```

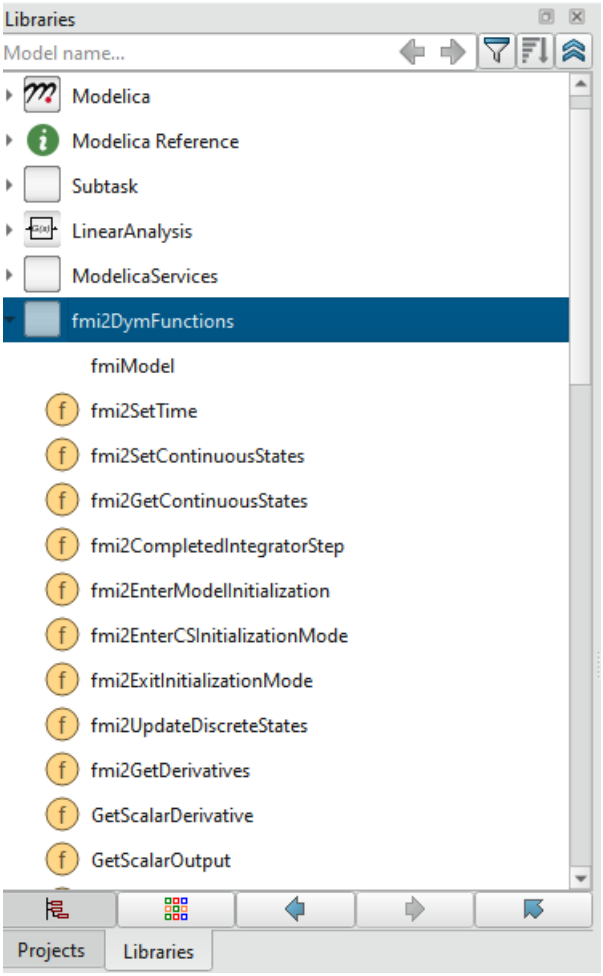
a standardized import package with all FMI2 and FMI3 functions that are common between FMUs is loaded and unitized in the Modelica wrapper for the FMU for both co-simulation and model exchange.

(The flag is by default false.) The flag is not saved between sessions.

For an example, start by keeping the default value `false` for the above flag and export and import the Coupled Clutches demo as a model exchange FMU using FMI 3. In this case, all FMI functions are present inside the Modelica wrapper for the FMU. To see this, you must first activate the setting **Show protected classes** using the command **Tools > Options**, in the **Package Browser tab**. Now you can look at the imported FMU in the package browser:



Now set the flag `Advanced.Beta.FMI.ExternalLibrary = true` and re-import the FMU. The FMI functions are now included in the external library under the **Libraries** tab of the FMU.



All imported FMI 2 FMUs will share the C-functions from the **fmi2DymFunctions** library. For FMI 3, the external library is called **fmi3DymFunctions**.

Important! FMU source code import is currently not supported using such external libraries.

(This feature appeared in Dymola 2026x.)

Generating adjoint derivatives for the ODE problem

As a part of handling parameter sensitivity in, for example, FMUs, you can generate analytical adjoint derivatives for the ODE problem by setting the flag:

```
Advanced.Beta.Translation.Generate.AdjointDerivatives = true
```

Setting the flag to `true` makes the function fmi3GetAdjointDerivative more efficient.

(The flag is by default `false`.)

(This feature appeared in Dymola 2025x Refresh 1.)

Improved variability check for GenerateEvents-functions

To improve the variability check for GenerateEvents-functions, you can now set the flag:

```
Advanced.Beta.Translation.GenerateEventChangeVariability = true
```

(The flag is by default `false`.) The basic idea is that when events are generated for a function returning a Boolean based on a Real there should be no error that the Boolean is a continuous-time Boolean (as it isn´t), but instead an error if the function uses noEvent internally.

In most cases this was already handled correctly since the check is usually done after inlining – but this flag generalize it, and also checks the functions themselves.

(This feature appeared in Dymola 2025x Refresh 1.)

More efficient handling of arrays

To minimize the number of declare-calls for array variables you can set the flag:

```
Advanced.Beta.Translation.ArrayDeclare = true
```

(The flag is by default `false`.) The flag is not saved between sessions.

The idea is to have only one declare-call per array variable, if possible, instead of one declare-call per array element.

(This feature appeared in Dymola 2026x.)

SSP: Specifying the SSP version to 2.0 when exporting

To always store the system description as version 2.0 when exporting an SSP, you can set the flag:

```
Advanced.Beta.SSP.MinimumVersion2 = true
```

(The flag is by default false. The flag is saved between sessions.)

(This feature appeared in Dymola 2026x.)

SSP: Option to export SSP parameter bindings as external files

To export the SSP parameter bindings as external files (SSV files), if possible, you can use the flag `Advanced.Beta.SSP.ExportExternalBinding`. The possible values of the flag are:

- **0 Inline par. binding** This is the default value.
- **1 System par. binding external**
- **2 All external**

(The flag value is saved between sessions.)

The file names are derived from the component or system name.

If the external files cannot be written for some reason, Dymola will instead store the parameter bindings inline.

(This feature appeared in Dymola 2026x.)

Minor issue with the built-in function spatialDistribution

If using the built-in function `spatialDistribution` for a variable that changes significantly at events, those discontinuities should, in some cases, be propagated to get the correct functionality. This can be done by setting the flag:

```
Advanced.Beta.Translation.TransportDelayDiscontinuity = true
```

(The flag is by default `false`.) The flag is not saved between sessions.

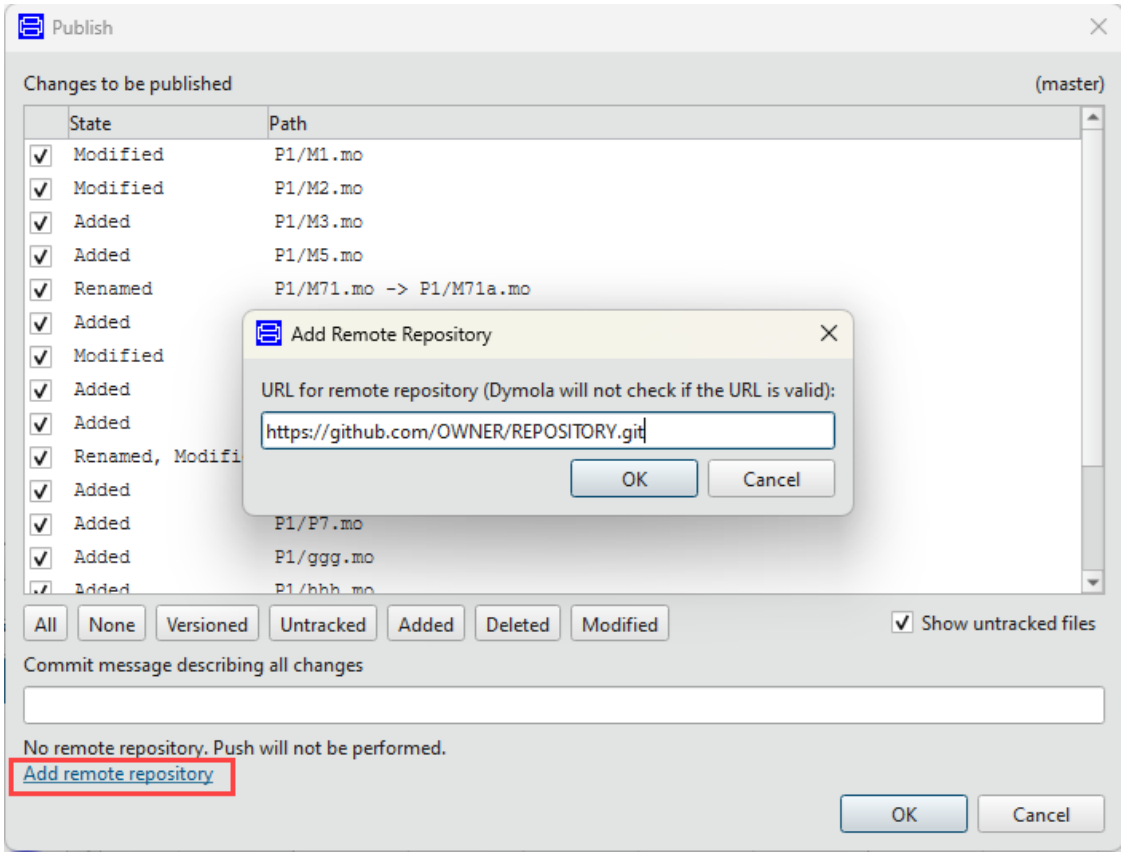
(This feature appeared in Dymola 2026x.)

3.6 Model Management

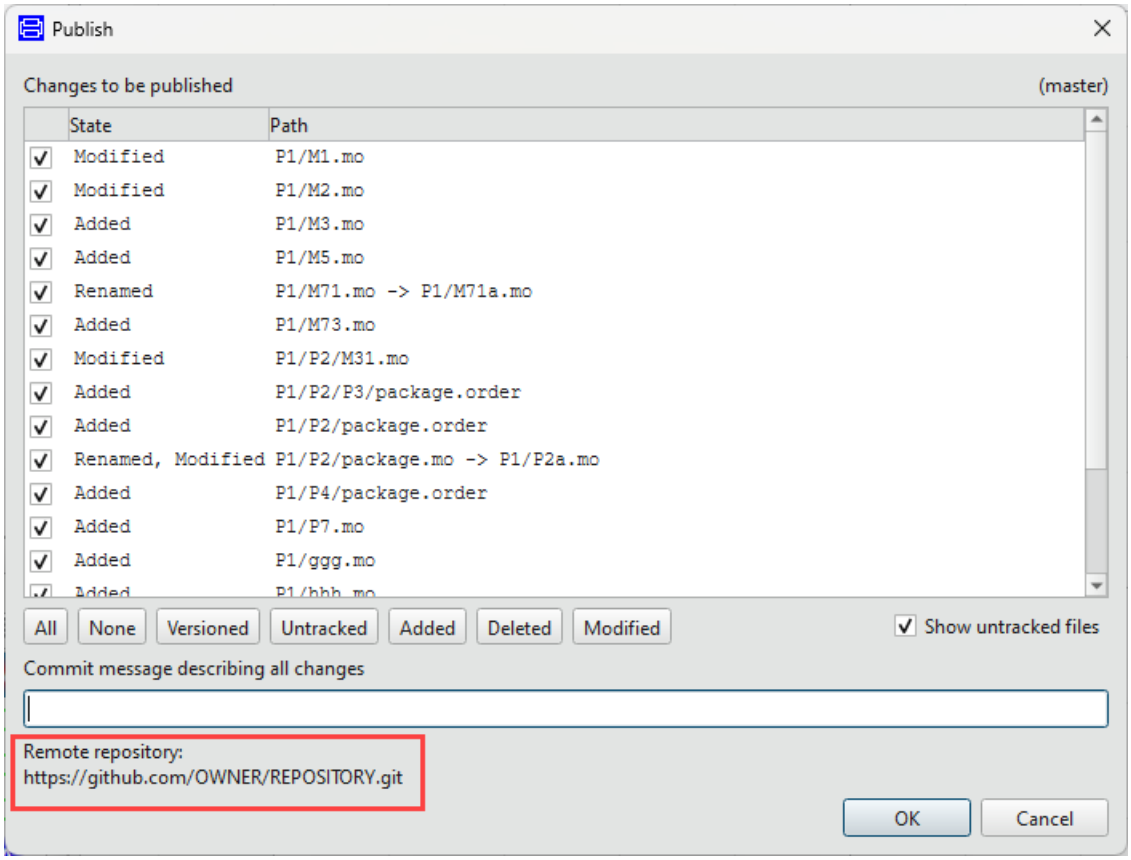
3.6.1 Extended Git support

Support of remote directories

Remote Git directories are supported in Dymola 2026x. If the Git model does not have a remote directory, there is a link in the Publish dialog where you can set one:



As in previous image, enter a URL and press OK. The remote directory is set and the Publish dialog is updated:



To create the remote directory, Dymola will run the command `git remote add origin` followed by the URL you entered. Note that Git, and therefore Dymola, does not check the remote repository URL; you can type anything.

If you set a remote repository using the command line, you get the same behavior.

```
E:\gitmodels\repo2>git remote add origin https://github.com/OWNER/REPOSITORY.git
E:\gitmodels\repo2>
```

3.6.2 Improvements in the 3DEXPERIENCE app "Design with Dymola"

Using 3DEXPERIENCE Platform files in Design with Dymola

General

Apart from using the 3DEXPERIENCE Platform as a versioning tool for Dymola models, you can directly use files located in the 3DEXPERIENCE Platform in the following cases (for details, see the sections below):

- When a model has a parameter that is about selecting a file, you can select, in most cases, a file from the 3DEXPERIENCE Platform.
- If you want to use an FMU, SPP, or SVV file, you can import it from the 3DEXPERIENCE Platform

Important! Since we are linking to objects the 3DEXPERIENCE Platform, there are two major advantages:

- You can use the 3DEXPERIENCE Platform tools for versioning etc.
- You can change the objects in the 3DEXPERIENCE Platform and then reuse them in Design with Dymola.

About files in the 3DEXPERIENCE Platform

Files as we know them, for example, CSV files or FMU files, are never stored as "pure files" in the 3DEXPERIENCE Platform; they are stored as/in "documents", sometimes referred to as ENOVIA Documents. Such a document can be seen as a container for one or several files.

The location of a file in the 3DEXPERIENCE Platform can be specified an URL.

To work with documents in the 3DEXPERIENCE Platform, you can use the ENOVIA app "Document Management". Using this app, you can create documents, upload files, and perform other actions on documents/files.

This app is documented in the 3DEXPERIENCE documentation. To reach the top level of the documentation, use the link https://www.3ds.com/support/documentation. Note that you need a 3DEXPERIENCE ID to access the documentation. (Note also that how to reach this documentation sometimes changes, but below is the status now.)

From this top level, under **User Guides**, select **Explore Now**.

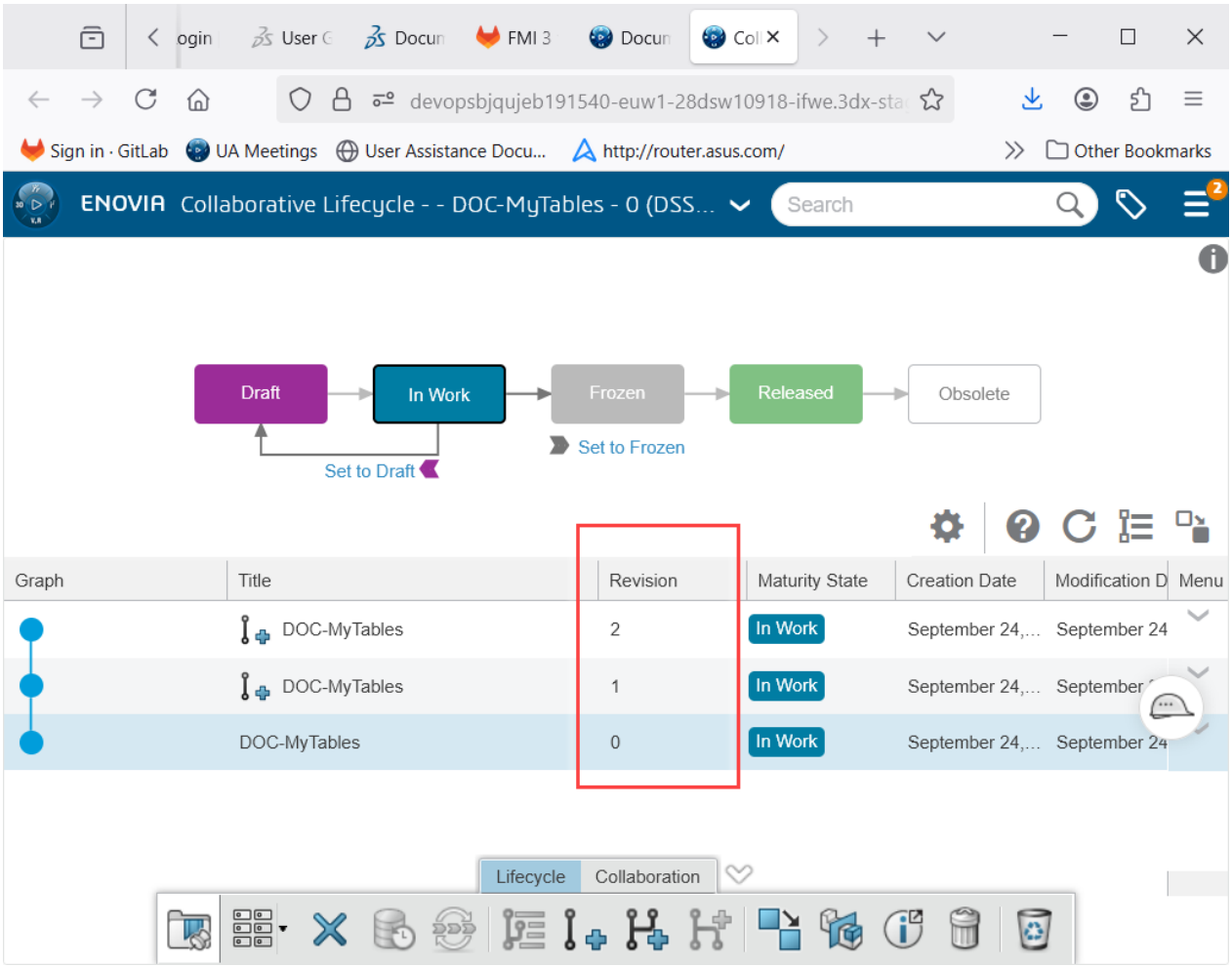
Under **3DEXPERIENCE platform**, under **3DEEXPERIENCE on the Cloud**, select the preferred language. You are now at the top level of the documentation pages.

In the left pane, expand **Social and Collaborative > Structured Collaboration & Collaborative Content > Document Management Wed App**. If you expand this entry, a good starting point may be "Document Management Home Page".

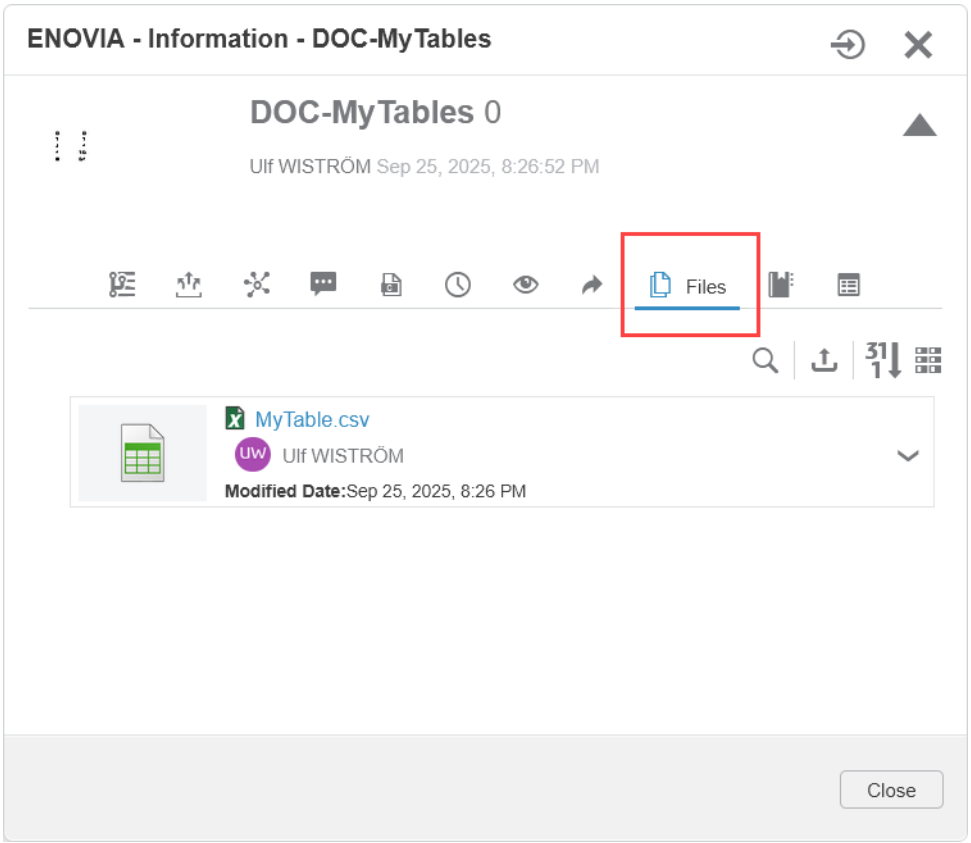
About iterations and revisions

If you update a *file* inside a document, you have created a file *iteration*. For revisions, you create a new *revision* of a *document* using the ENOVIA Collaborative Lifecycle app.

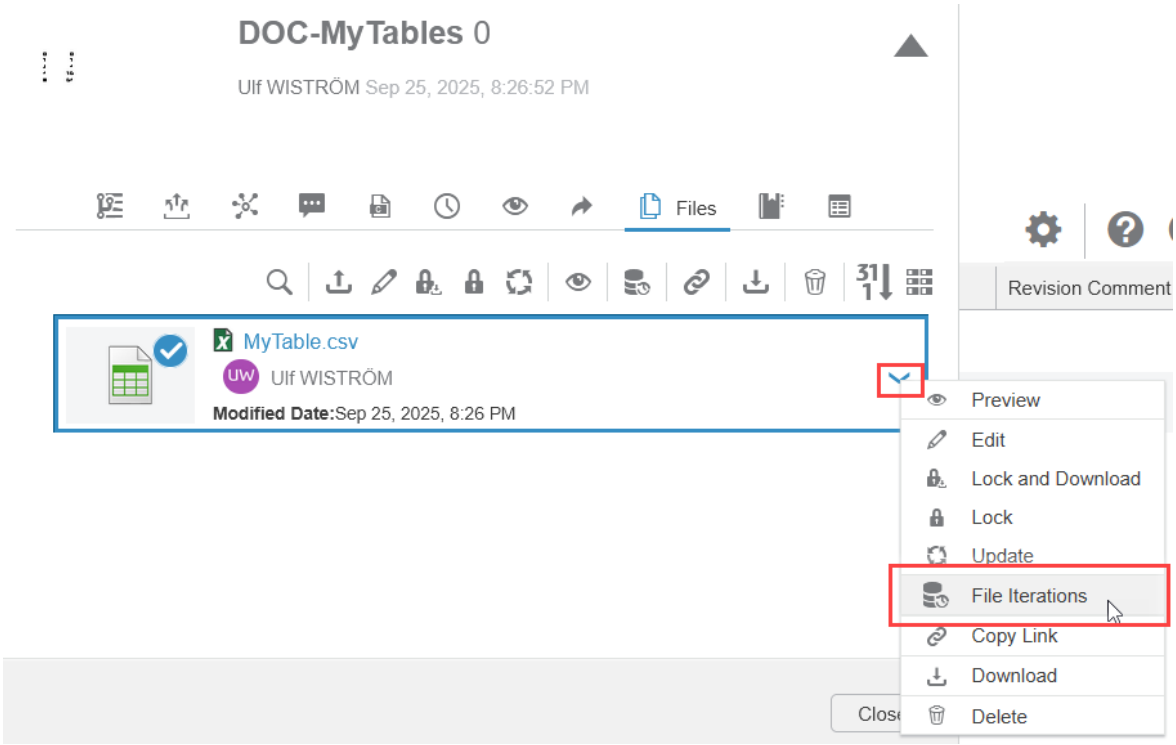
Let us look at a document with several revisions:



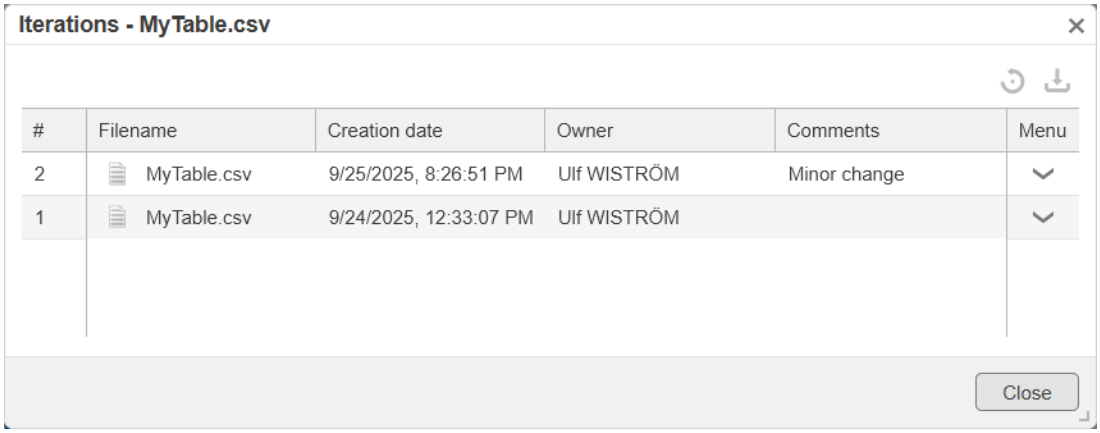
Now, let us look at the file iterations in the document revision 0. Right-click the lowest line (revision 0) and select **Information**. In the window you will get, select the **Files** tab. You will get:



We see that this revision contains only one file. To see the iterations of this file, click the arrow and select, like:



You will get:



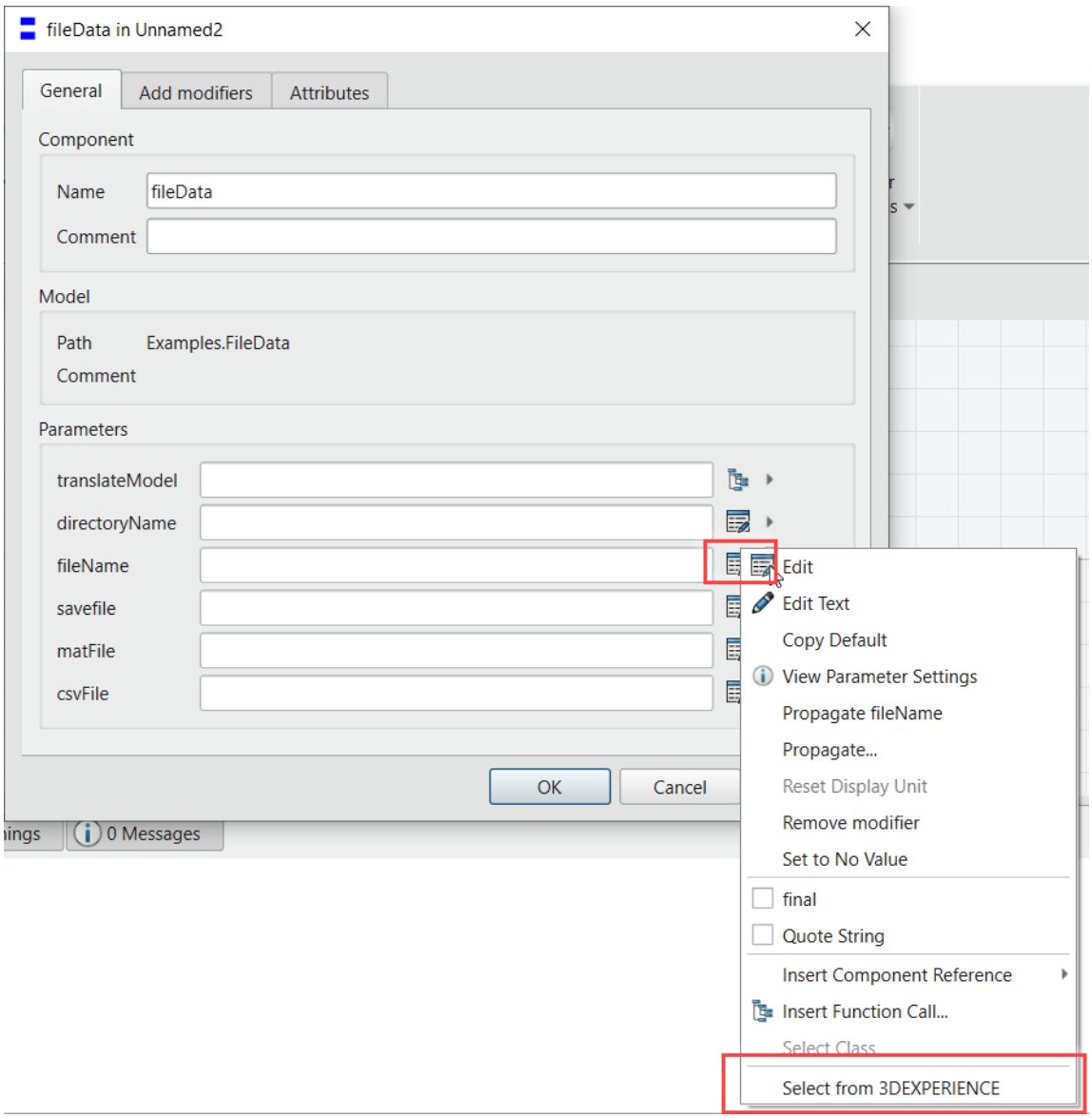
So we see that we have two iterations of this file.

We don´t go further into this here, but it is important to understand the principal difference between iterations and revisions.

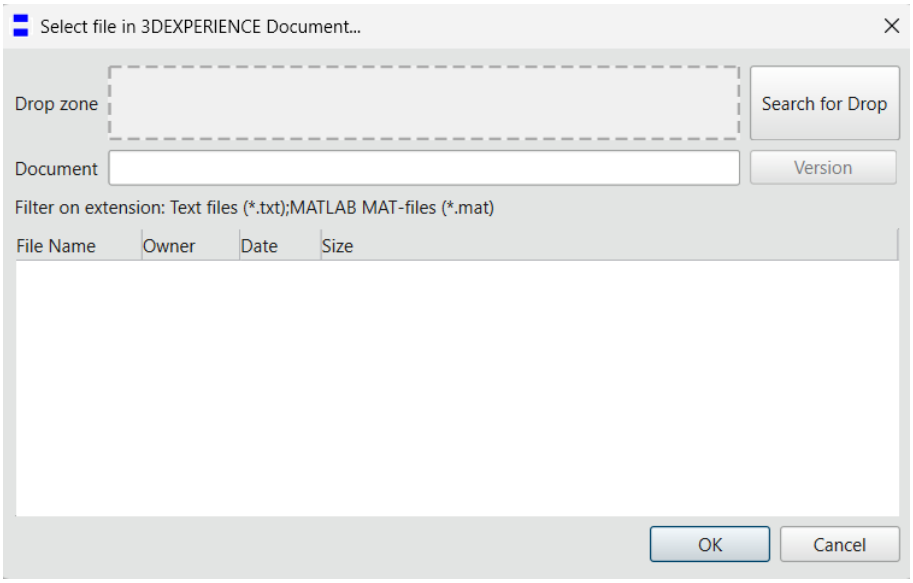
For a model parameter, specifying a file from the 3DEXPERIENCE Platform

Specifying the file.

To create a model parameter for selecting a file, you use the annotation `loadSelector`. When a model with such a parameter is instantiated as a component, you can use the context menu of that parameter to select a file from the 3DEXPERIENCE Platform. A simple example of such a component with different file handling parameters (click the arrow after the **Edit** icon of the file selection parameter to get the context menu):



If you now click **Select from 3DEXPERIENCE**, you will get the dialog:

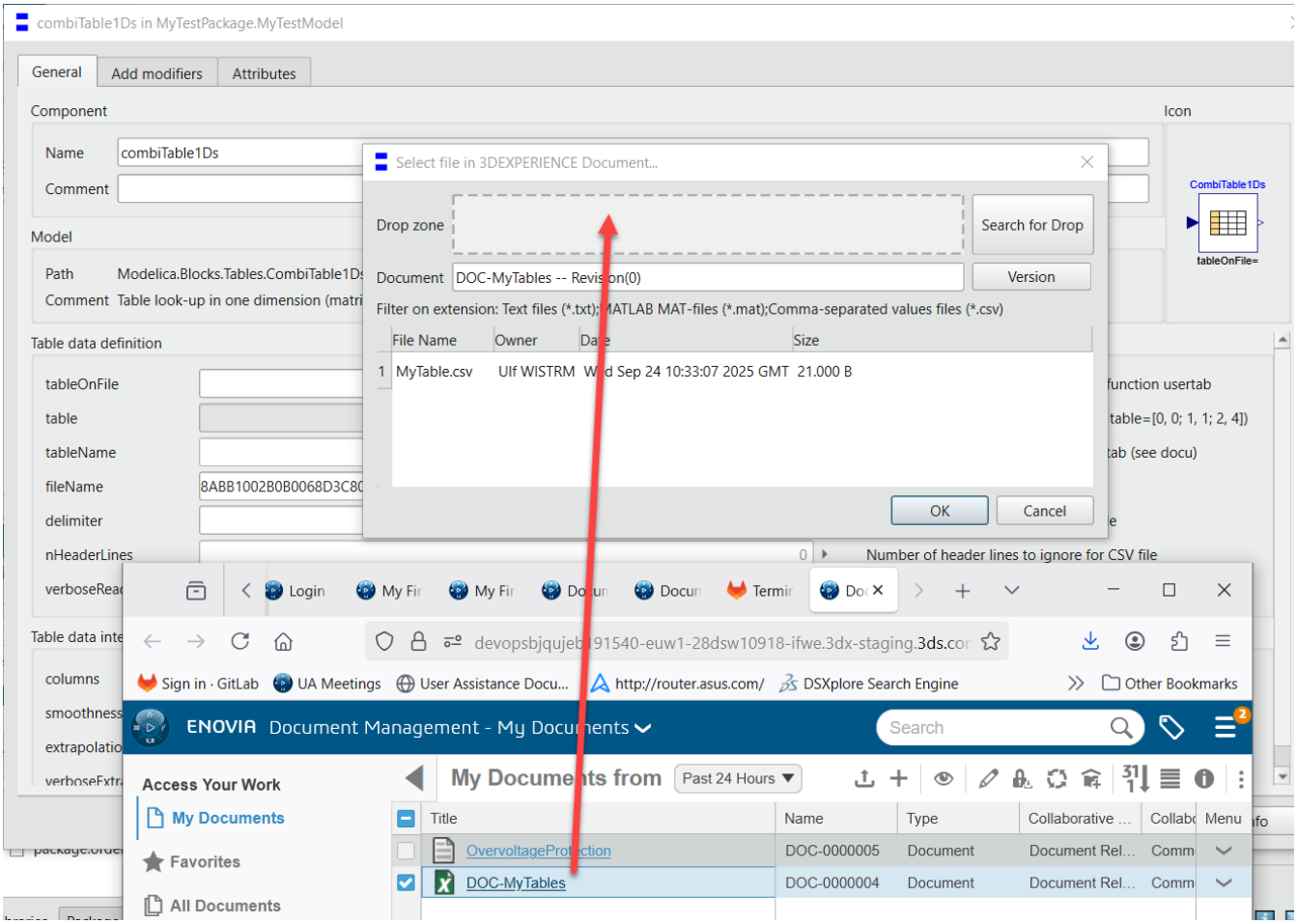


The idea is that you should start by clicking **Search for Drop**. This will open the Document Management app, and display the document you can select from. You can then drag any of these documents and drop it in the **Drop zone**.

Important! You *must* drop in the **Drop zone**, that is the only way to select a file.

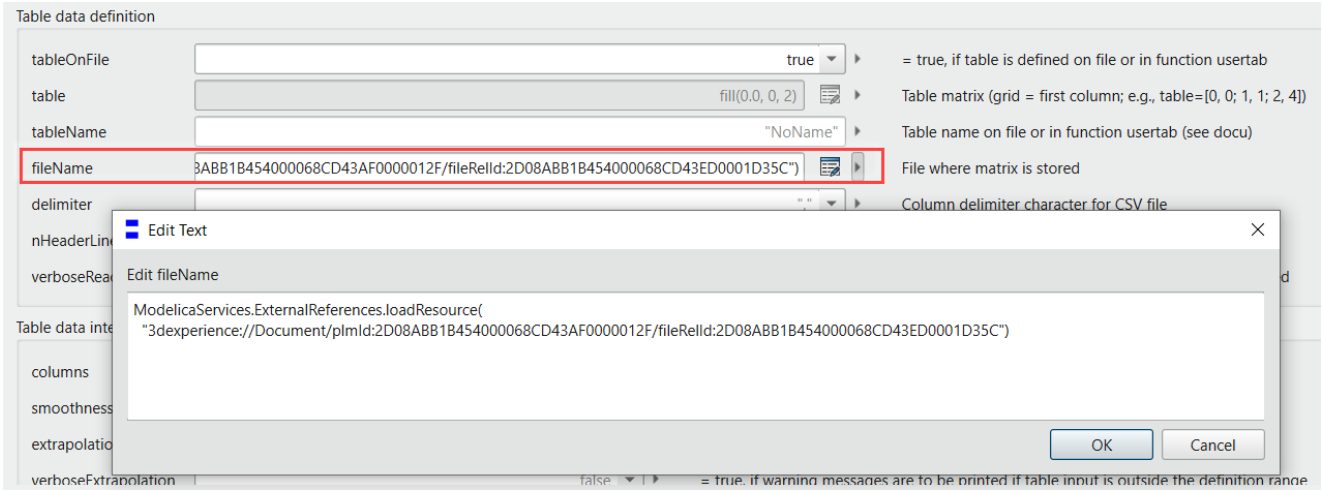
Actually, you first target a document, if there are more files of the wanted type in the document; you have to perform a second selection in the lower part of the dialog to specify the file. (For an example of dialog when you have to select, see the next section.)

In a bit more realistic example, the drop can look like:

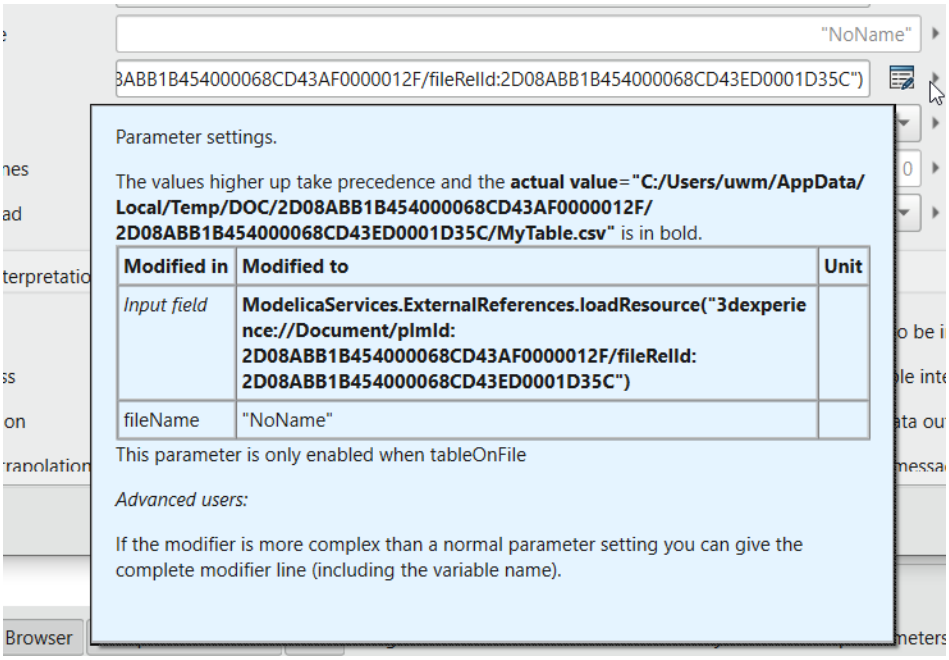


Actually, you see the result after the drop; the **Document** field shows the dropped file.

Going back to the parameter dialog, we can see that the file is represented by the URL to it. We can use the context menu of the parameter; if you select you can use the context command **Edit Text** to see the full URL:



The context menu command **View Parameter Settings** also gives good information:



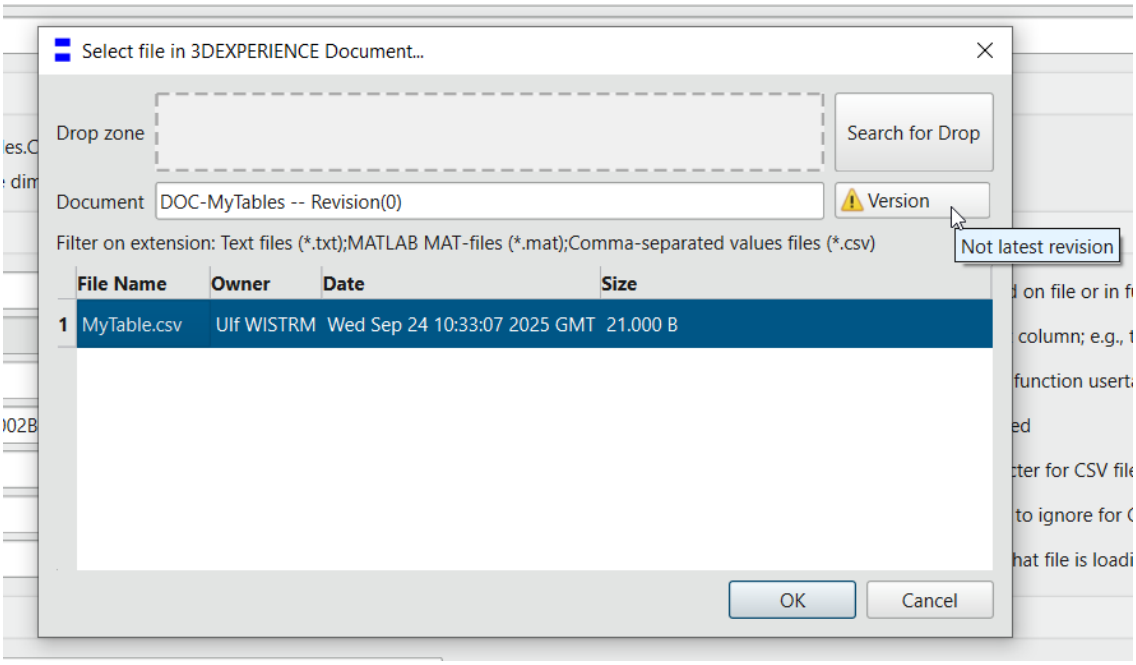
Working with files in the 3DEXPERIENCE Platform means that you can select revision etc. For more about this, see the 3DEXPERIENCE documentation.

Updating the file.

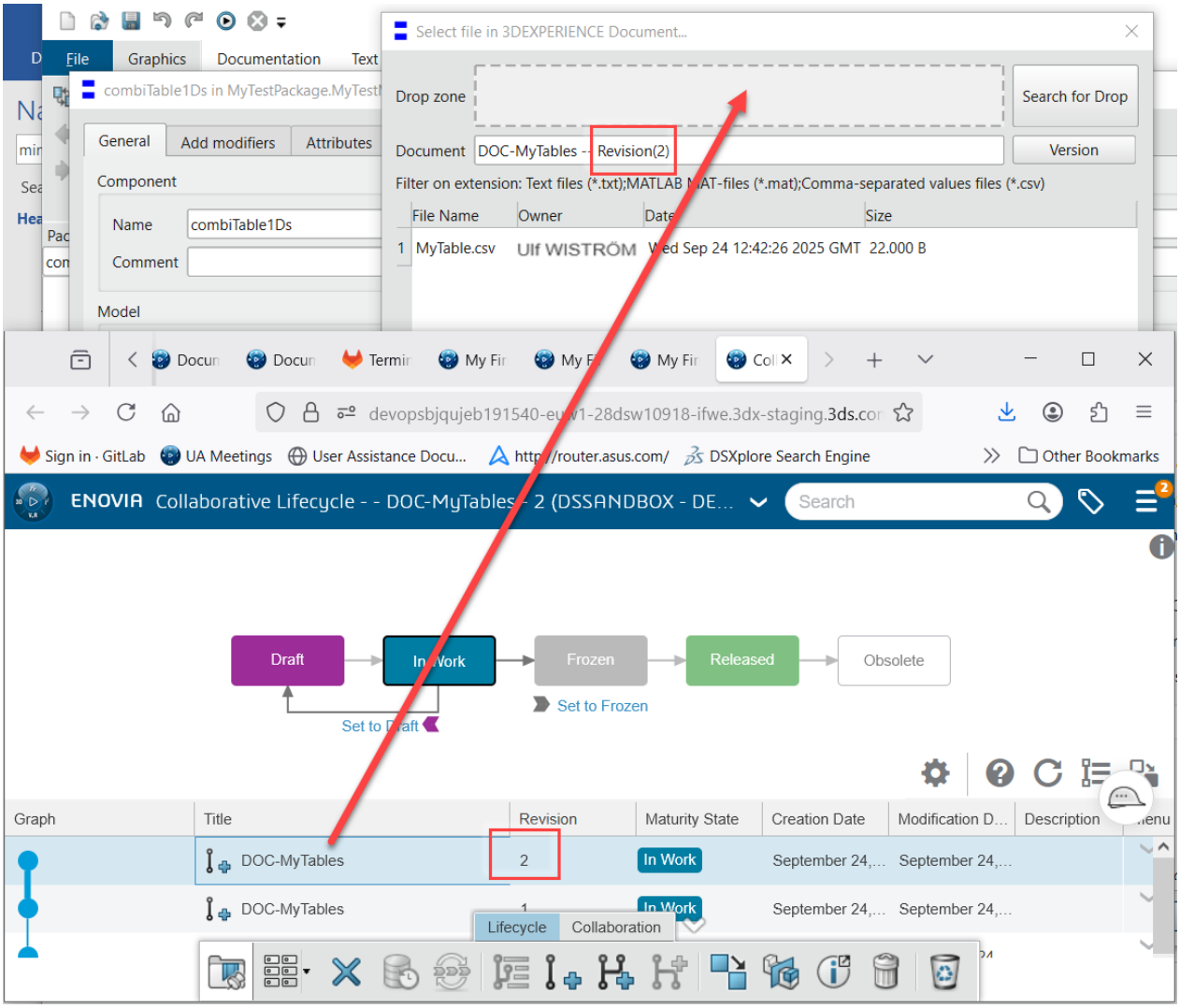
To update the file, there are two cases (see “About iterations and revisions” above for those terms):

Only a new *iteration* of the used file is available in the platform (we work with only one document revision, or have created the new iteration in the document revision we work with). In this case, you will get the new iteration by performing a new translation of the model. Note that you will get no warning if a new iteration is available.

A new *revision* of the document where the file is located is available in the platform. In this case you see a warning *if* you open the context menu of the parameter and select **Select from 3DEXPERIENCE**. This will open the **Select file in 3DEXPERIENCE Document…** window (as above), with the selected file prefilled. Now you also see a warning icon before the **Version** command:



In this case, to use the latest revision, you must click **Version**, and, from the Collaborative Lifecycle app that opens, drag the wanted revision to the **Drop zone**. An example:



You actually see the result of such an update, the warning has disappeared, and the revision is changed.

To use the new revision, you must translate the model.

Importing an FMU, an SSP, or an SSV from the 3DEXPERIENCE Platform

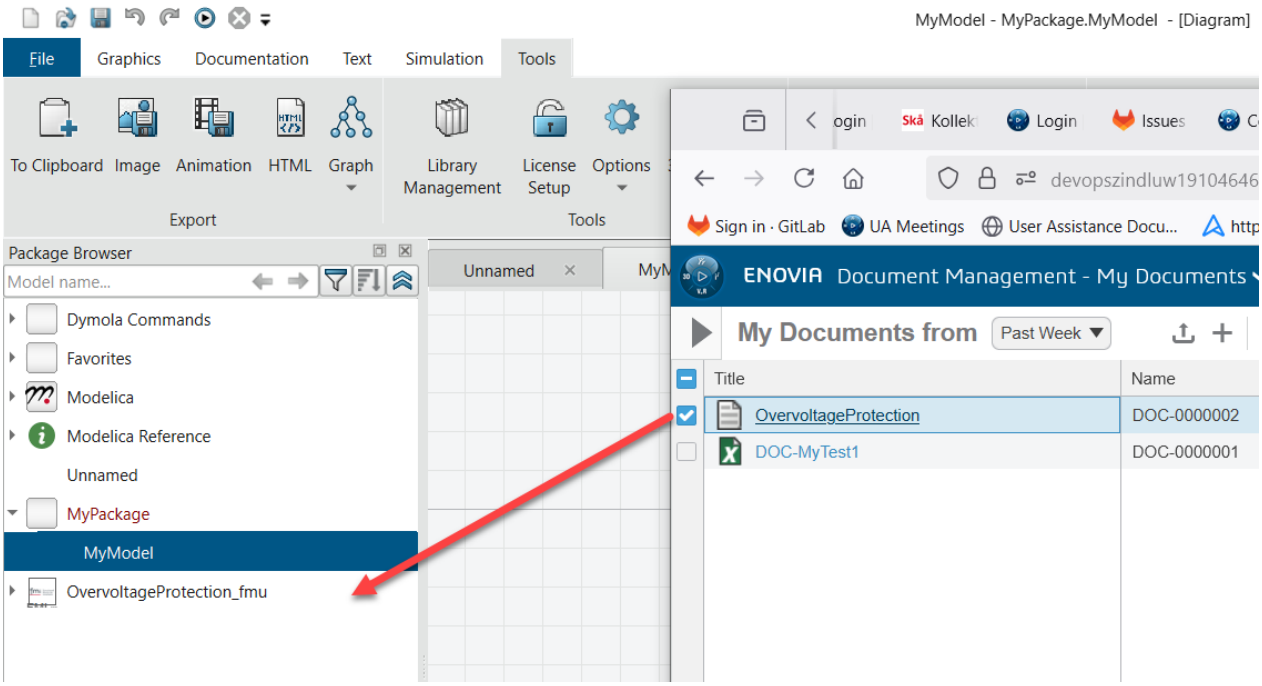
Specifying the FMU

You can import an FMU, an SSP or and SSV from the 3DEXPERIENCE Platform, simply by dragging the file from the 3DEXPERIENCE Platform and dropping it in the package browser of Design with Dymola.

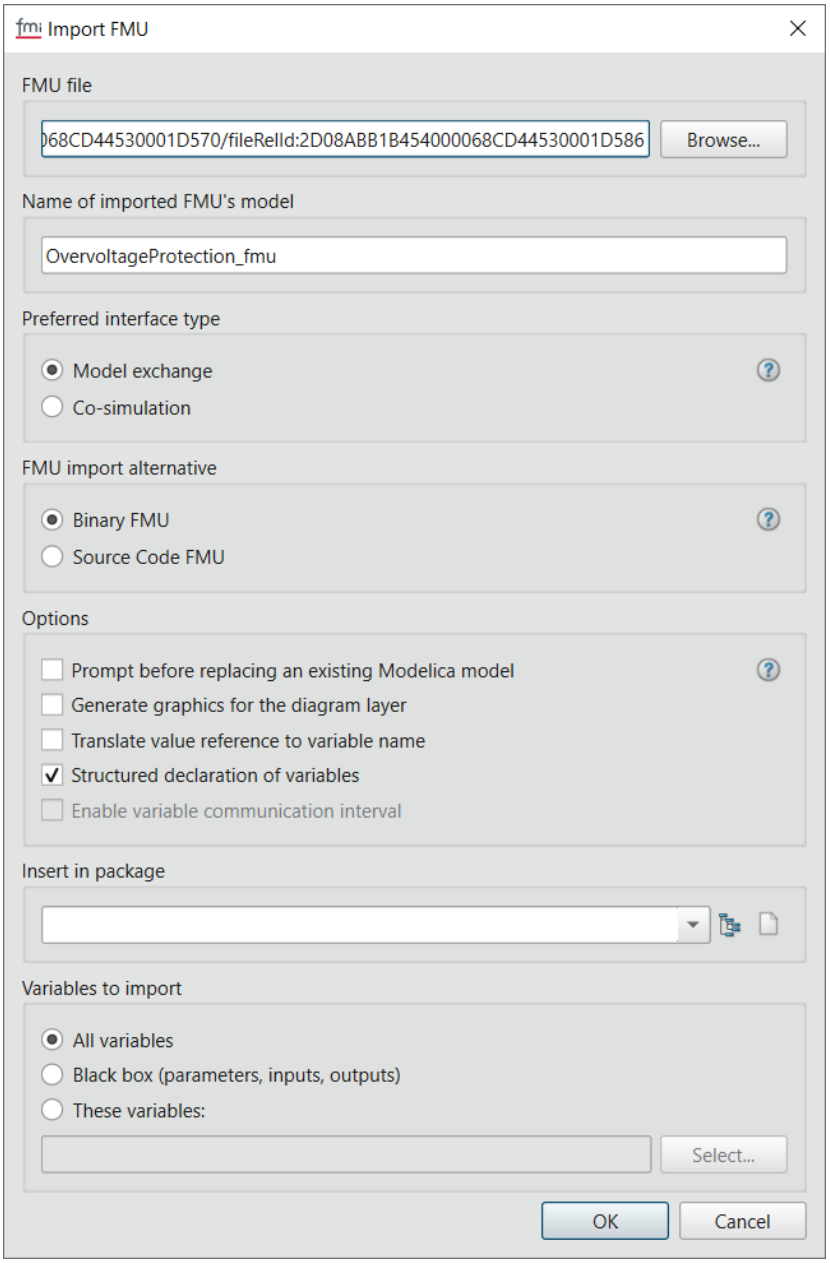
The below exemplifies working with an FMU, it works the same if working with an SSP or SSV file.

To display the FMU to drop in the 3DEXPERIENCE Platform, you can use the Document Management app. In that app, you can have a number of documents, containing one or several FMUs (and, of course, other files). **Note** that you can now only import one FMU per document, if there are more FMUs in the document; you have to perform a second selection to specify the file.

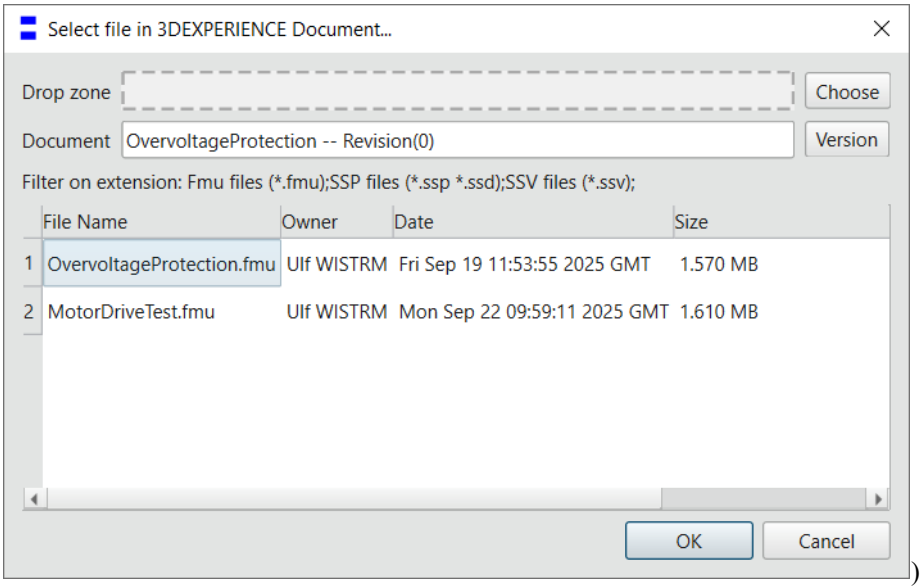
An example of dropping:



When dropping, you, in this case get the dialog for importing the FMU:



(If you have more than one FMU in the document, you must however first select which FMU to import, by selecting file and clicking **OK** in a dialog like this:

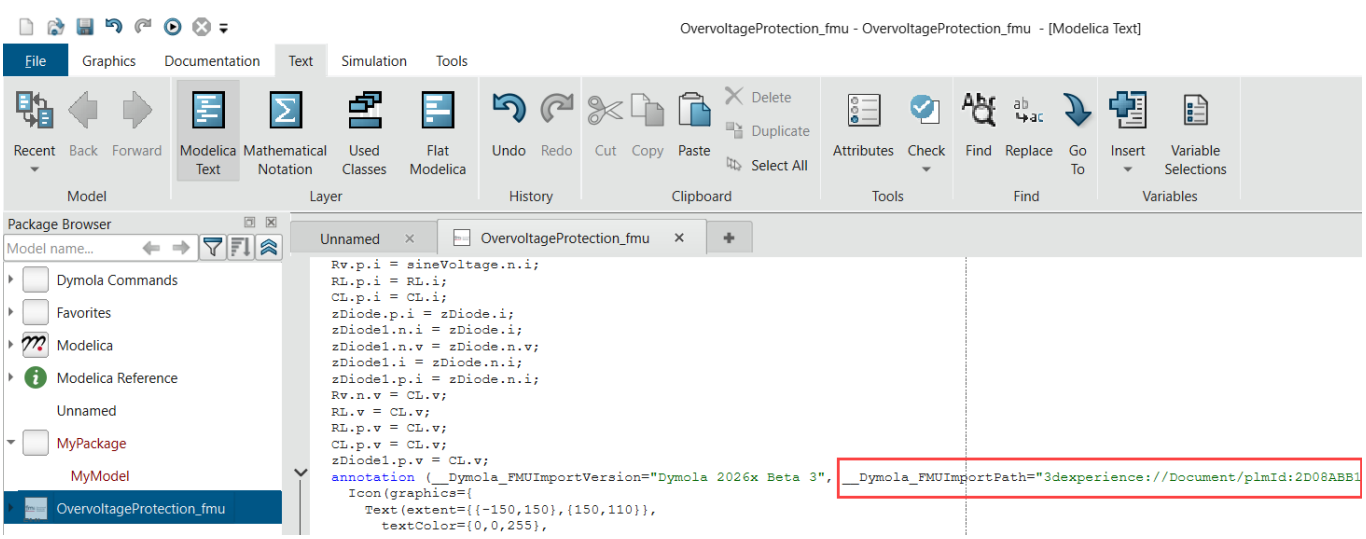


)

Note that above the default behavior is displayed, in most cases you want the FMU to be in a package, you have then to create the package before importing the FMU, and use **Insert in package** in the above dialog to select what package to import the FMU to.

To see the path for the imported you can do the following:

- Double-click the imported FMU in the package browser.
- In the Dymola ribbon, click the tab **Text**.
- Scroll down to the last **annotation** item. Click the arrow before it to expand it – you will now get:



(The URL is quite long, so the full URL is not seen in the image above.)

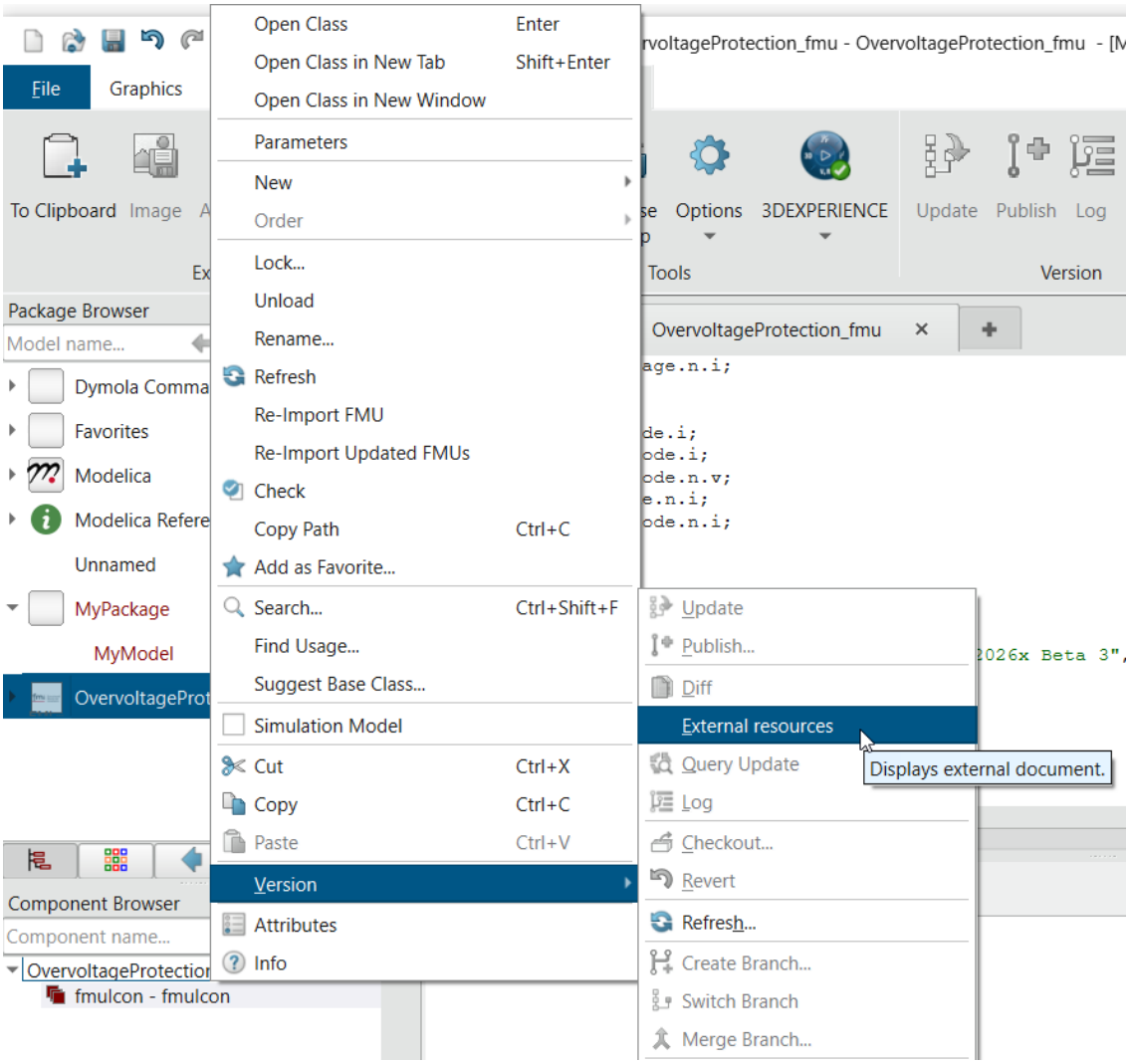
Updating the FMU.

To update the FMU, there are two cases (see "About iterations and revisions" above for those terms):

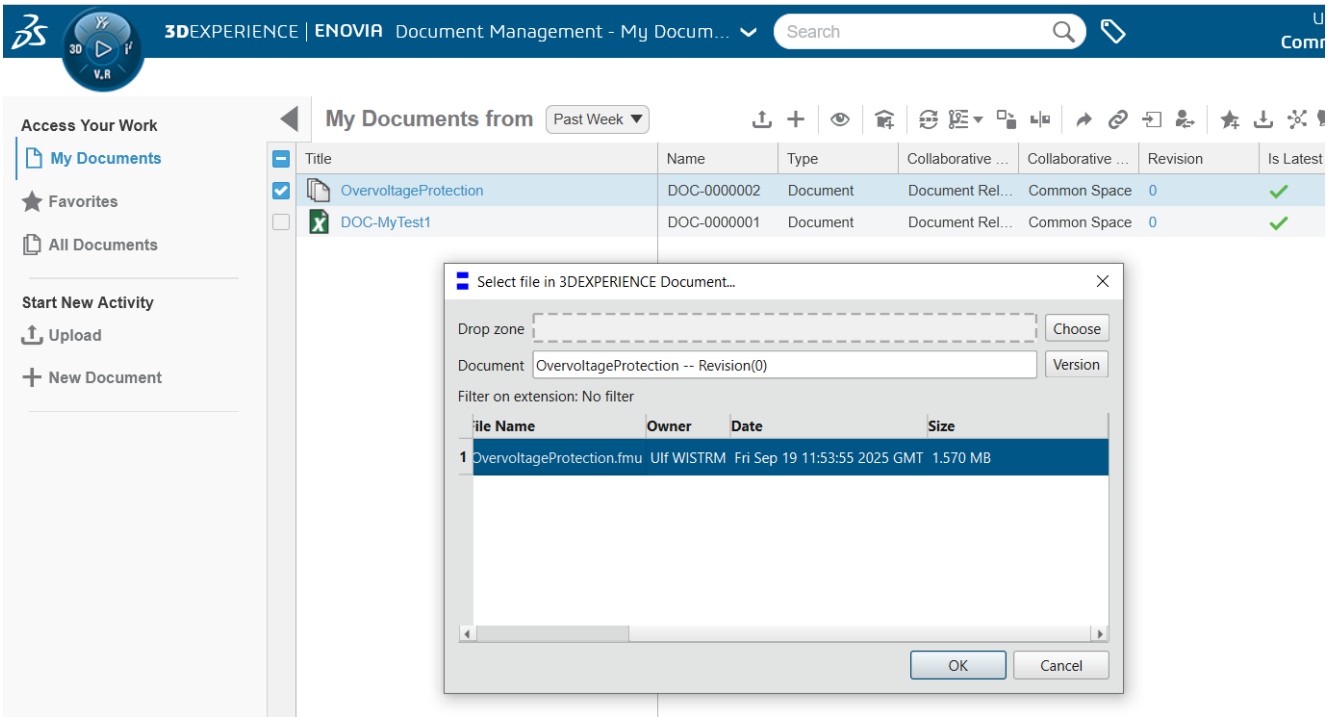
Only a new *iteration* of the used FMU is available in the platform (we work with only one document revision, or have created the new iteration in the document revision we work with). In this case, you will get the new iteration by right-clicking the FMU in the package browser and selecting **Re-Import FMU**. You have to translate again to use the new iteration. Note that you will get no warning if a new iteration is available.

A new *revision* of the document where the FMU is located is available in the platform. To see the status (if a new revision is available) and to update, you must open the **Select file in 3DEXPERIENCE Document…** window.

To do this, right-click the FMU in the package browser, select **Version**, and then **External Resources**:

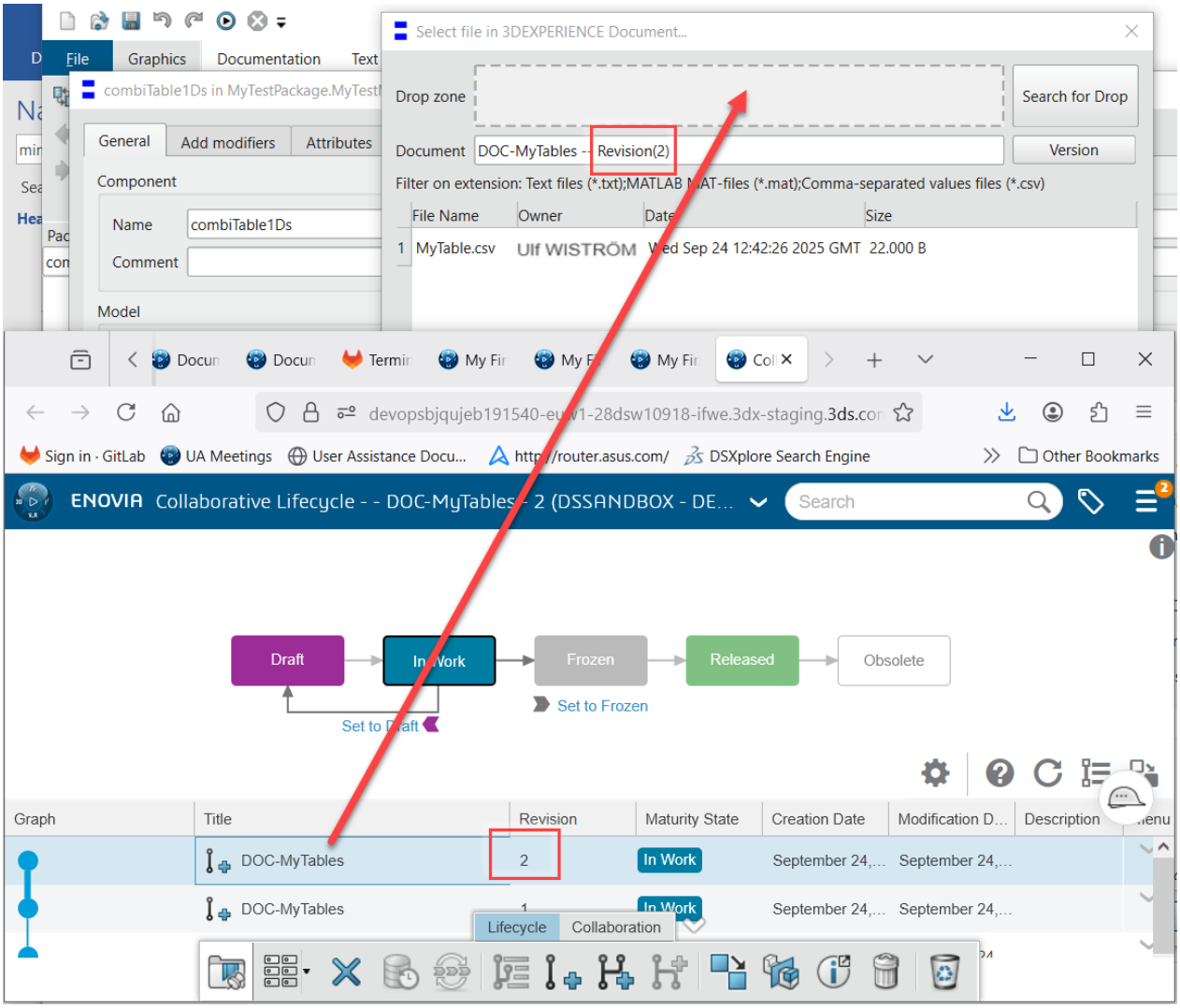


This command will open My Documents page in the Document Management app, and in particular, the file selector for the document where the FMU is stored. Here you, for example, can select a more recent revision of the FMU; you can even select to use another FMU (if any other is available in the document):



Here the latest available document revision is used; there is no warning sign in front of the **Version** command. If that was not the case, it will look like:

In this case (if you get this warning), to use the latest revision, you must click **Version**, and, from the Collaborative Lifecycle app that opens, drag the wanted revision to the **Drop zone**. An example (the image is from previous section, but the principle is the same for FMUs):



You actually see the result of such an update, the warning has disappeared, and the revision is changed.

To use the new revision, you must translate the model.

3.7 Other Simulation Environments

3.7.1 Dymola – Matlab interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2020a (ver. 9.8) up to R2025a (ver. 25.1). On Windows, only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. On Linux, the gcc compiler is supported. The LCC compiler is not supported, neither on Windows nor on Linux.

3.7.2 Real-time simulation

Compatibility – dSPACE

Dymola 2026x officially supports the DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2026x generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases:

- dSPACE Release 2020-A with Matlab R2020a
- dSPACE Release 2020-B with Matlab R2020b
- dSPACE Release 2021-A with Matlab R2021a
- dSPACE Release 2021-B with Matlab R2021b
- dSPACE Release 2022-A with Matlab R2022a
- dSPACE Release 2022-B with Matlab R2022b
- dSPACE Release 2023-A with Matlab R2023a
- dSPACE Release 2023-B with Matlab R2023b
- dSPACE Release 2024-A with Matlab R2024a
- dSPACE Release 2024-B with Matlab R2024a and R2024b
- dSPACE Release 2025-A with Matlab R2024a, R2024b, and R2025a

The selection of supported dSPACE releases focuses on releases that introduce support for a new Matlab release and dSPACE releases that introduce a new version of a cross-compiler tool. In addition, Dymola always support the three latest dSPACE releases with the three latest Matlab releases. Although not officially supported, it is likely that other combinations should work as well.

New utility functions – dym_rti_build2 and dym_rtimp_build2

Dymola 2021 introduced a new function, `dym_rti_build2`, which replaces `dym_rti_build` for building dSPACE applications from models containing DymolaBlocks. The new function uses the new dSPACE RTI function `rti_build2` instead of the old function `rti_build`.

A corresponding new multi-processor build function, `dym_rtimp_build2`, is also introduced.

These functions are supported with dSPACE Release 2019-B and later.

Note on dym_rti_build and dSPACE Release 2017-A and later

The function `rti_usrtrcmerge` is no longer available in dSPACE Release 2017-A and later. Therefore, it is required to run the standard `rti_build` function (with the 'CM' command) after `dym_rti_build` to get your _usr.trc content added to the main .trc file. For example:

```
>> dym_rti_build('myModel', 'CM')
>> rti_build('myModel', 'Command', 'CM')
```

Note that this note applies the new functions `dym_rti_build2` and `rti_build2` as well.

Compatibility – Simulink Real-Time

Compatibility with Simulink Real-Time has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R2020a (Simulink Real-Time ver. 6.12) to R2025a (Simulink Real-Time ver. 25.1). Only Microsoft Visual C compilers have been tested.

3.7.3 Java, Python, and JavaScript Interface for Dymola

Supported Java versions for the Java Interface updated

From Dymola 2026x, supported Java versions are Java 17 and higher. Note that for 3DEXPERIENCE applications, the officially supported Java versions are Java 21 and higher.

This update is valid for the "new" Java interface, note that the support for the "old" one is now discontinued, see section "Discontinued support for the old Java interface, including external Java" on page 100.

Supported Python versions for the Python interface updated

Already from Dymola 2025x Refresh 1, supported Python versions are Python 3.10 and higher.

3.7.4 SSP Support in Dymola

Important; additional features!

Note that there are some FMI Beta features as well. Please see the section "Features under Development" starting on page 63.

SSP read-modify-write cycle

The SSP read-modify-write cycle, including both import and export, has been improved:

- You can export an SSP archive by selecting the top-level package in Dymola, and export the entire package and all its models. Exporting a model inside the package will only export that model.
- By default, the SSP archive will also contain the current contents of the `resources` and `extra` folders. If you don´t want that, you can set the flag

`Advanced.SSP.ExportCopyResources = false`. (The value of the flag is by default `true`. The value of the flag is saved between sessions.)

- The import of SSP by default unpacks the entire SSP file in the current directory, including whatever files exist in the SSP file, even if Dymola cannot understand the content. If you want to have the previous behavior, see section "Unpacking an SSP file in a temporary directory" below.
- To make a clean import, Dymola by default purges the resources when importing SSP files. For more information, see section "Purging resources when importing SSP files" below.

SSP import

Purging resources when importing SSP files

You can purge the 'resouces' directory, the 'extra' directory, and any .ssd files when importing an SSP file. You can do that using the flag `Advanced.SSP.PurgeResources`. To purge gives a clear view of the SSP contents in these directories. The possible values of the flag are:

- **No purge** (0)
- **Files only** (1) (default value)
- **Flies and directories** (2)

Notes

- The **Files only** alternative is useful when files and directories are managed by a version control system..
- The default alternative **Files only** is a change from the earlier behavior in Dymola, which was corresponding to **No purge**.

Unpacking an SSP file in a temporary directory

By default, when importing an SSP file, Dymola unpacks all the contents of the SSP file in the current directory, including documentation files and files not used by Dymola.

To unpack instead in a temporary directory, and unpacking only the FMU and SSD files that are used, you can set the flag:

```
Advanced.SSP.TempUnpack = true
```

(The default value of the flag is `false`.) The flag value is stored between sessions.

Note: The default value `false` is a change from the earlier behavior in Dymola, which was corresponding to `true`.

Providing placement when needed

Some tools exporting SSP files do not provide any graphical information. To make components visible in the graphical editor, Dymola can provide initial placement when needed. This feature is by default activated. You can deactivate the feature by setting the flag:

```
Advanced.SSP.ProvidePlacement = false
```

(The default value of the flag is `true`,) The value of the flag is kept between sessions.

Improved handling of inlined components

SSP import creates Modelica models for any SSP systems that used inline as components in a system description. This was also the case in previous Dymola versions, but then such models were created locally inside the model where they were used. In Dymola 2026x, those models are created at the same level as the model where they are used.

SSP export

Exporting protected items

You can select to include protected members in the SSP export. You do that by setting the flag:

```
Advanced.SSP.ExportProtected = true
```

(The default value of the flag is `false`.) The value of the flag is kept between sessions.

Option to save files without creating an SSP directory when exporting

The built-in function `exportSSP` for exporting an SSP has a String argument `fileName` that specifies what file name to use for the created SSP file that in turn contains the created files (SSD files etc.). If you set the argument to "`unpacked`", no SSP file is created, the files that were to be packed into that file will instead be saved separately. This can be used to support the save-in-repository model.

Updated SSP export of FMU components

The SSP export of FMU components has been updated. Dymola will no longer export parameters for input connector start values and internal FMI co-simulation parameters. This means that the exported parameter list corresponds to the declared FMU parameters.

Exporting as SSV instead of SSP if the model just contains an extends-clause

Dymola creates a model with modifiers or a base class when importing SSV files. This idea is now also implemented for export, if you perform an SSP export of a model that just contains an extends-clause, the model is exported as SSV.

3.7.5 FMI Support in Dymola

Unless otherwise stated, features are available for FMI version 1.0, 2.0, and 3.0.

Important; additional features!

Note that there are some FMI Beta features as well. Please see the section “Features under Development” starting on page 63.

FMI Export: Shorter default model identifiers

By default, the model path is included in the default FMU model identifier.

As an example, when you generate an FMU from the Coupled Clutches demo, the default model identifier is:

```
Modelica_Mechanics_Rotational_Examples_CoupledClutches
```

From Dymola 2026x, you can select to omit the model path in the default model identifier, that is, for the above you just get:

```
CoupledClutches
```

To omit the model path in the default model identifier, you can set the flag `Advanced.FMI.ShortDefaultModelIdentifier = true`.

(The flag is by default `false`. The flag is saved between sessions.)

FMI Export: All solvers supported for analyzing numeric integration

In general, you can use the option **Which states that dominate error** to investigate which variables are the critical ones if the simulation is slow due to tolerance requirements for some state variables. You can activate this option by using the command **Simulation > Setup**, the option is in the **Debug** tab.

When you simulate an FMU that has been exported with this option activated, you can look in the simulation log to see the critical variables. Notes:

- For *all* FMUs, you must also activate the option **Debug settings included in translated or exported models** before exporting the FMU for the above option to work. (This option in located in the **Debug** tab as well)
- For FMUs using any of the solvers Cvode or Ida, you must *also* activate **Store result in mat file** in the FMU export dialog when you export the FMU.

The above is the main use for analyzing FMUs, but the option also allows you to analyze the numeric integration using the command **Simulation > [Simulation] Performance**. Note that when you use this alternative, only one FMU can have **Which states that dominate error** activated when simulating. Also, the above notes applies as well.

FMI 3: Continued implementation of Beta support for terminals and icons

Please see section “FMI 3: Support for terminals and icons in general” starting on page 66.

Minor improvement FMI 3 export: Update of variable-size parameter arrays

The default value of the flag `Advanced.FMI3.ExposeDynamicArrays` has been changed to `false`. Previously if was set to `true` by default, requiring users to manually set the unknown arrays. It can however happen that if unknown arrays that are not set, the FMU cannot simulate. This implies that is better the user must deliberately activate this feature if wanted.

FMI Import: Delayed unpacking of FMU

To improve collaboration when working with FMUs, you can delay the unpacking of imported FMUs until translation. This improves collaboration, as there is no need to share directories of unpacked FMUs.

To activate this feature, set the flag:

```
Advanced.FMI.DelayUnpackingOfBinaries = true
```

(The flag is by default `false`.) The flag is not saved between sessions. Note that this feature is always activated when working with Design with Dymola.

To use this feature in a good way, any FMU should preferably be stored inside an open Modelica package, and in that case, the references to the FMU will use Modelica-URIs instead of a file path.

When importing the FMU with the above flag set, only the necessary information for constructing the import-interface is unpacked, and references to resources and binaries will refer to parts of the FMU – not any unpacked information on disk. Images for the FMU icon and connectors are included in the FMU wrapper and do not refer to external files.

When translating a model with such FMUs all of the information will be unpacked and used for running the simulation. A diagnostics will be given if the FMU has changed.

Note that the flag only matters when importing the FMU – when translating the model unpacked FMUs and non-unpacked FMUs can freely be combined.

3.7.6 eFMI Support in Dymola

A number of improvements have been implemented:

Support for embedded code generation for neural networks

You can now generate embedded code for neural networks modeled via `eFMI.NeuralNeworks`, including preservation of multi-dimensional tensor-flow arithmetic's in the generated eFMI GALEC code.

For an example of neural networks integrated into Modelica model/physics equations, please consult `eFMI_TestCases.M11_NeuralQVM`.

New GALEC built-in functions implementation of Software Production Engineering

A new GALEC built-in functions implementation of Software Production Engineering is implemented. The new implementation satisfies MISRA C:2023 and is compliant with the upcoming eFMI Standard 1.0.0 Beta 2, which finalizes the specification of GALEC built-in functions.

Added Arduino support

You can now build Arduino sketches from eFMU Production Code containers. This enables deployment of production code on Arduino compatible hardware

(`.DymolaEmbedded.EmbeddedConfiguration.ProductionCode.build_Arduino_sketch()`). For details, please consult the requirements documentation.

Support for tunable parameters in MATLAB/Simulink exporter

The MATLAB/Simulink scripts to import eFMI Production Code containers generated by Software Production Engineering now also support tunable parameters exposed via the `__Dymola_eFMI_ExposeTunableParameters = true` annotation.

Checking eFMUs for eFMI Standard compliance with eFMPy

You can now check eFMUs for eFMI Standard compliance with eFMPy. The function `.DymolaEmbedded.EmbeddedConfiguration.check_eFMU()` is extended with an eFMPy argument besides the existing ones to configure eFMI Compliance Checker and eFMI Container Manager checks. For details, please consult the requirements documentation.

Preserving multi-dimensional equations of source models

The GALEC code generator now supports avoiding scalarizing multi-dimensional equations of source models and instead preserve them. The involved multi-dimensional variables are also treated correctly in the generated GALEC code. To apply this, you can use the new `code_configuration.multidimensional_aritmetics` option for the eFMU generation.

Note. This feature is in an experimental prototype stage and requires explicit annotation of the multi-dimensional equations to preserve a multi-dimensional expression. You do that by using the annotation `__Dymola_Expand = false`. This annotation is supported for equations and for-loops of equation sections.

Some minor improvements

- Further Modelica implementations of the built-in functions of the eFMI GALEC language added (`solveUnivariatePolynomial`).
- Updated check of code scripts to work with the latest Cppcheck version (open source 2.18.0, premium 25.8.1)

3.7.7 Model structure: Improved model editing API

Introduction

The functions to create and edit Modelica models using function calls, located in the package `ModelManagement.Structure.AST` have been slightly improved. Below the changed functions are listed, in alphabetical order in each subpackage.

Changed AST functions

For editing classes (in the **Classes** subpackage)

Name	Description of changes
CreateClass	A new Boolean input argument `overwrite` has been added, by default `true` (to preserve backward compatibility). If you set it to `false`, it prevents overwriting. **Note** that you can use `AST.Misc.ClassExists` to test if the class exists.
ExtendsInClass	A new Boolean input argument `returnPath` has been added, by default `false`. If you set it to `true`, the full path of each extended class is returned, not only the local name.
GetClassAttributes	Two new output arguments have been added: - A Boolean argument `isEncrypted` that returns `true` if the class is encrypted (and not concealed). - A String argument `fileName` that returns the name of the file where the class is defined.

3.8 Advanced Modelica Support

Discontinued support for the old Java interface, including external Java

Previously, an old Java implementation was available, intended for calling Java functions from Modelica, or Modelica functions from Java functions. Starting from Dymola 2026x, this implementation is not available anymore. A newer Java interface for accessing Dymola remotely has been available since long, and FMI provides integration of models in other languages.

3.9 New libraries

Below is a description of new libraries. For a full description, please refer to the libraries documentation.

3.9.1 Sustainable Supply Systems Library

With the Sustainable Supply Systems library, you can efficiently model power systems with multiple different sources of energy.

The library aims to accelerate the evaluation and optimization of complex, interconnected supply systems, energy, and resource management. By offering a scalable system simulation framework for creating virtual twins, it helps organizations to predict the most efficient path to achieve sustainability goals.

To enable the above, the library provides

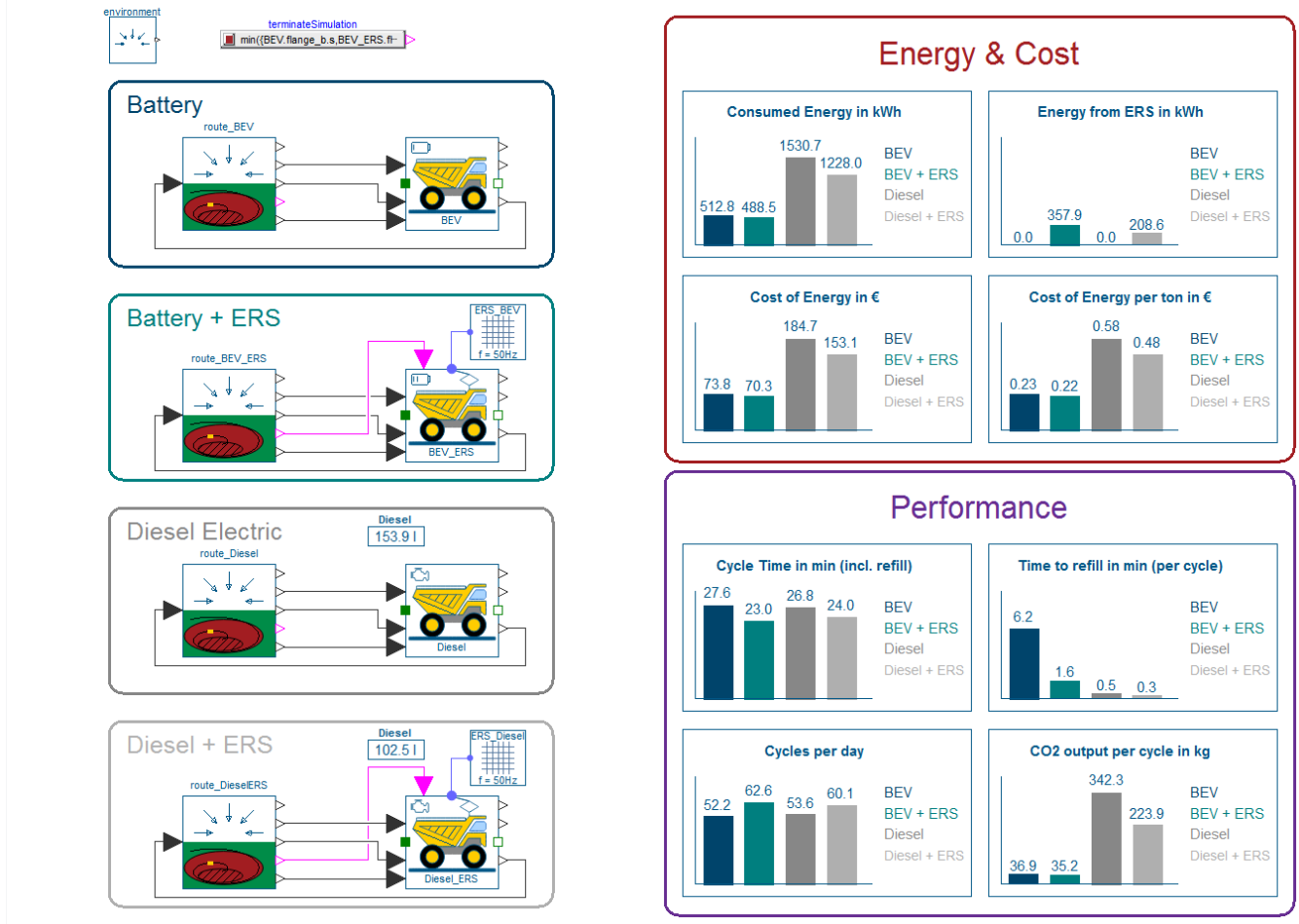
- Entry level simulation with low data requirements
- High computational performance

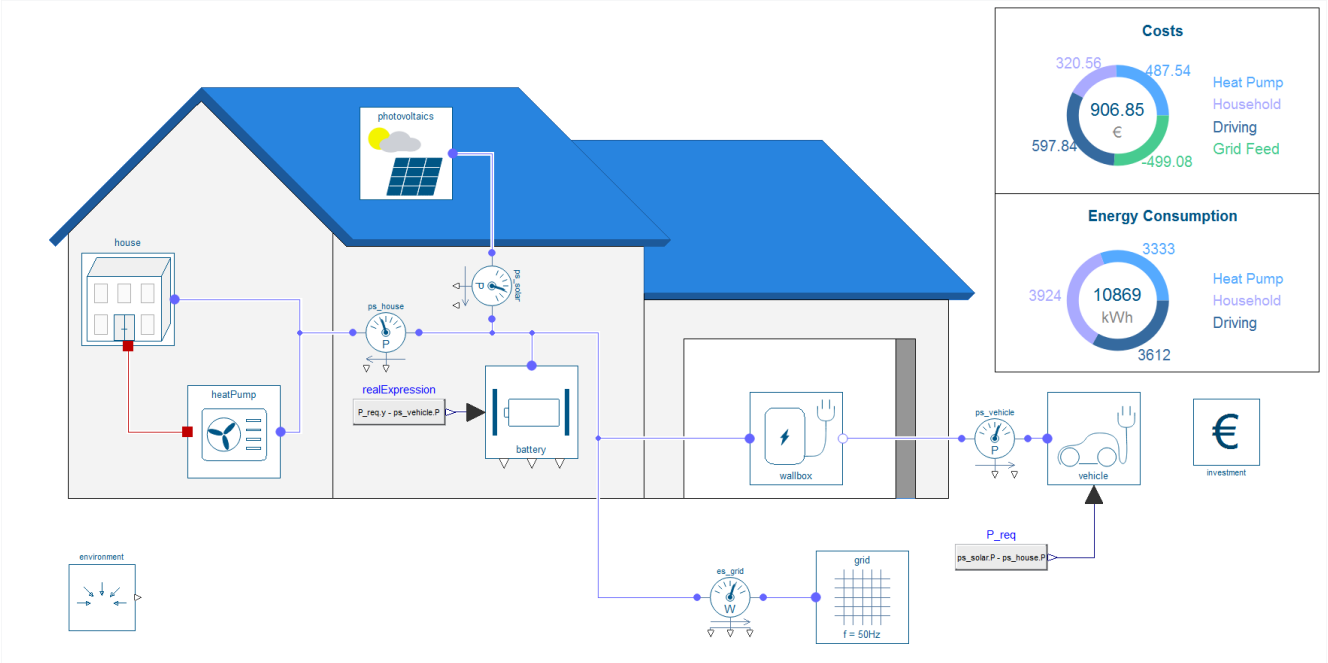
Applications and customer requests include:

- Marine & Offshore
- Mine Electrification
- Vehicle2Grid
- Rail Industry
- Business Services

The library is a commercial and licensed Modelica library.

A couple of examples of applications; a haul truck and a smart home:


environment
terminateSimulation
min({BEV.flange_b.s,BEV_ERS.fl-
Battery
route_BEV
BEV
Battery + ERS
ERS_BEV
f = 50Hz
route_BEV_ERS
BEV_ERS
Diesel Electric
Diesel
153.9 l
route_Diesel
Diesel
Diesel + ERS
Diesel
102.5 l
ERS_Diesel
f = 50Hz
route_DieselERS
Diesel_ERS
Energy & Cost
Consumed Energy in kWh
512.8
488.5
1530.7
1228.0
BEV
BEV + ERS
Diesel
Diesel + ERS
Energy from ERS in kWh
0.0
357.9
0.0
208.6
Cost of Energy in €
73.8
70.3
184.7
153.1
Cost of Energy per ton in €
0.23
0.22
0.58
0.48
Performance
Cycle Time in min (incl. refill)
27.6
23.0
26.8
24.0
Time to refill in min (per cycle)
6.2
1.6
0.5
0.3
Cycles per day
52.2
62.6
53.6
60.1
CO2 output per cycle in kg
36.9
35.2
342.3
223.9



3.10 Modelica Standard Library and Modelica Language Specification

The current version of the Modelica Standard Library is version 4.1.0. The current version of the Modelica Language Specification is 3.6.

3.11 Documentation

General

In the software, distribution of Dymola 2026x Dymola User Manuals of version “September 2025” will be present; these manuals include all relevant features/improvements of Dymola 2026x presented in the Release Notes, except the “under development” ones (if present).

Updated Modelica Tutorial

There is a Modelica tutorial that is reachable by the command **Tools > Help Documentation**. The tutorial was originally for the old Modelica version 1.4, and has been so until the previous version. The tutorial has now been updated; the text and the examples are now correct with respect to the latest Modelica version, Modelica 3.6. Note however that the new features

introduced after Modelica 1.4 are not described. In other words, the tutorial is describing features present in Modelica 1.4, but with text and examples of Modelica 3.6.

3.12 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2026x are listed.

3.12.1 Hardware requirements/recommendations

Hardware requirements

- At least 2 GB RAM
- At least 1 GB disc space

Hardware recommendations

At present, it is recommended to have a system with an Intel Core 2 Duo processor or better, with at least 2 MB of L2 cache. Memory speed and cache size are key parameters to achieve maximum simulation performance.

A dual processor will be enough if not using multi-core support; the simulation itself, by default, uses only one execution thread so there is no need for a "quad" processor. If using multi-core support, you might want to use more processors/cores.

Memory size may be significant for translating big models and plotting large result files, but the simulation itself does not require so much memory. Recommended memory size is 6 GB of RAM.

3.12.2 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions

Dymola 2026x is supported, as 64-bit application, on Windows 10 and Windows 11. Since Dymola does not use any features supported only by specific editions of Windows ("Home", "Professional", "Enterprise" etc.), all such editions are supported if the main version is supported.

Compilers

Please note that for the Windows platform, a Microsoft C/C++ compiler, or a GCC compiler, must be installed separately. The following compilers are supported for Dymola 2026x on Windows:

Microsoft C/C++ compilers, free editions:

Note. When installing any Visual Studio, make sure that the option "C++/CLI support…" is also selected to be installed. (Also, note that Bundled Visual Studio toolsets are not supported. The workaround is to install the older build tools separately instead of bundled in e.g. a Visual Studio 2022 installation.)

- Visual Studio 2015 Express Edition for Windows Desktop (14.0)
- Visual Studio 2017 Desktop Express (15) **Note!** This compiler only supports compiling to Windows 32-bit executables.
- Visual Studio 2017 Community 2017 (15)
- Visual Studio 2017 Build Tools **Notes:**
    - The recommended selection to run Dymola is the workload "Visual C++ build tools" + the option "C++/CLI Support…"
    - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
    - This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2017 alternative: **Visual Studio 2017/Visual C++ 2017 Express Edition (15)**.
    - For more information about installing and testing this compiler with Dymola, see the document Installing Visual Studio Build Tools for Dymola.pdf.
- Visual Studio 2019 Community (16). Note that you can select to use Clang as code generator for this compiler.
- Visual Studio 2019 Build Tools **Notes:**
    - The recommended selection to run Dymola is the workload "C++ build tools" + the option "C++/CLI Support…"
    - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
    - This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2019 alternative: **Visual Studio 2019/Visual C++ 2019 (16)**.
    - For more information about installing and testing this compiler with Dymola, see the document Installing Visual Studio Build Tools for Dymola.pdf.
    - You can select to use Clang as code generator for this compiler.
- Visual Studio 2022 Community (17). Note that you can select to use Clang as code generator for this compiler.
- Visual Studio 2022 Build Tools **Notes:**
    - The recommend selection to run Dymola is the workload "Desktop development with C++" + the option "C++/CLI Support…"

- Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
- This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2019 alternative: **Visual Studio 2022/Visual C++ 2022 (17)**.
- For more information about installing and testing this compiler with Dymola, see the document Installing Visual Studio Build Tools for Dymola.pdf.
- You can select to use Clang as code generator for this compiler.

Microsoft C/C++ compilers, professional editions:

Note. When installing any Visual Studio, make sure that the option "C++/CLI support…" is also selected to be installed. (Also, note that Bundled Visual Studio toolsets are not supported. The workaround is to install the older build tools separately instead of bundled in e.g. a Visual Studio 2022 installation.)

- Visual Studio 2015 (14.0)
- Visual Studio Professional 2017 (15)
- Visual Studio Enterprise 2017 (15)
- Visual Studio Professional 2019 (16). Note that you can select to use Clang as code generator for this compiler.
- Visual Studio Enterprise 2019 (16). Note that you can select to use Clang as code generator for this compiler.
- Visual Studio Enterprise 2022 (17). Note that you can select to use Clang as code generator for this compiler.
- Visual Studio Professional 2022 (17) Note that you can select to use Clang as code generator for this compiler.

Clang compiler

If you first select to use Visual Studio 2019 or Visual Studio 2022 as compiler, you can then select to use Clang as code generator instead of native Visual Studio.

Intel compilers

Note!

Important. The support for Intel compilers are discontinued from the previous Dymola 2022 version.

MinGW GCC compiler

Dymola 2026x has limited support for the MinGW GCC compiler. The following versions have been tested and are supported:

- For 32-bit GCC: version 6.3 and 8.2. **Important!** The support for the MinGW 32-bit GCC compiler is deprecated in this version, Dymola 2026x. The support will be removed in Dymola 2026x Refresh 1.
- For 64 bit GCC: version 7.3 and 8.1. Normally, later versions will also be compatible.

To download any of these free compilers, please see the manual "*Dymola User Manual 1: Introduction, Getting Starting, and Installation*", the chapter "Appendix – Installation" where the latest links to downloading the compilers are available. Needed add-ons during installation etc. are also specified here. Note that you need administrator rights to install the compiler.

Also, note that to be able to use other solvers than Lsodar, Dassl, and Euler, you must also add support for C++ when installing the MinGW GCC compiler. Usually, you can select this as an add-on when installing GCC MinGW.

Current limitations with 32-bit and 64-bit Min GW GCC:

- Embedded server (DDE) is not supported.
- Support for external library resources is implemented, but requires that the resources support GCC, which is not always the case.
- FMUs must be exported with the code export option[1] enabled.
- For 32-bit simulation, parallelization (multi-core) is currently not supported for any of the following algorithms: RadauIIa, Esdirk23a, Esdirk34a, Esdirk45a, and Sdirk34hw.
- Compilation may run out of memory also for models that compile with Visual Studio. The situation is better for 64-bit GCC than for 32-bit GCC.

In general, 64-bit compilation is recommended for MinGW GCC. In addition to the specific GCC 32-bit limitations above, 32-bit tends to be less numerically robust. Note that the support for the 32-bit GCC compiler is deprecated in this version, Dymola 2026x. The support for 32-bit GCC will be removed in Dymola 2026x Refresh 1. (64-bit GCC will still be supported.)

WSL GCC compiler (Linux cross-compiler)

Dymola on window supports cross-compilation for Linux via the use of Windows Subsystem for Linux (WSL) GCC compiler. The default WSL setup is 64-bit only and Dymola adopts this limitation. Notes:

- WSL is usually not enabled on Windows, so you need to enable WSL on your computer and install needed software components.
- You must download and install a suitable Linux distribution, including a C compiler. We recommend Ubuntu 20 since it is the most tested version for Dymola. In particular, the integration algorithms RadauIIa, Esdirk23a, Esdirk34a, Esdirk45a, and Sdirk34hw have been confirmed to work with Ubuntu 20, but not with Ubuntu 18.
    - **Note** however that if you want to exchange FMUs with the Systems Simulation Design app (sometimes referred as "SID"), you should use Ubuntu 18.04 instead.
- The WSL Linux environment can compile the generated model C code from Dymola in order to produce a Linux executable dymosim or a Linux FMU. (To generate Linux FMUs, you must use a specific flag as well.)
- Note that you can select to use Clang as code generator for this compiler.

[1] Having the code export options means having any of the license features **Dymola Binary Model Export** or the **Dymola Source Code Generation**.

Dymola license server

For a Dymola license server on Windows, all files needed to set up and run a Dymola license server on Windows using FLEXnet, except the license file, are available in the Dymola distribution. (This includes also the license daemon, where Dymola presently supports FLEXnet Publisher version 11.16.2.1. This version is part of the Dymola distribution.)

As an alternative to FLEXnet, Dassault Systèmes License Server (DSLS) can be used. Dymola 2026x supports DSLS R2026x. Earlier DSLS versions cannot be used.

Note that running the Dymola license server on virtual machines is not a supported configuration, although it might work on some platforms.

Linux

Supported Linux versions and compilers

Dymola 2026x runs on Red Hat Enterprise Linux (RHEL) version 8.6, 64-bit, with gcc version 11.2.1, and compatible systems. (For more information about supported platforms, do the following:

- Go to https://doc.qt.io/
- Select the relevant version of Qt, for Dymola 2026x it is Qt 6.8.1.
- Select Supported platforms)

Any later version of gcc is typically compatible. In addition to gcc, the model C code generated by Dymola can also be compiled by Clang. (To be able to select Clang, it must be installed, e.g. on Red Hat Enterprise Linux (RHEL): `sudo yum install clang` – on Ubuntu: `sudo apt install clang`.)

You can use a dialog to select compiler, set compiler and linker flags, and test the compiler by the **Verify Compiler** button, like in Windows. This is done by the command **Simulation > Setup**, in the **Compiler** tab.

Dymola 2026x is supported as a 64-bit application on Linux. Corresponding support for 64-bit export and import of FMUs is included.

Notes:

- Dymola is built with Qt 6.8.1 and inherits the system requirements from Qt. However, since Qt 6.8.1 no longer supports embedding of the XCB libraries, these must now be present on the platform running Dymola. To know what to download and install, see the table in https://doc.qt.io/qt-6/linux-requirements.html for the list of versions of the ones starting with “libxcb”. Note that the development packages (“-dev”) mentioned outside the table are not needed.
- For FMU export/import to work, zip/unzip must be installed.
- Support for 32-bit simulation on Linux (including 32-bit export and import of FMUs) is discontinued from Dymola 2025x.

Note on libraries

- The following libraries are currently not supported on Linux:

- Process Modeling Library
- Thermodynamics Connector Library
- UserInteraction Library

Dymola license server

For a Dymola license server on Linux, all files needed to set up and run a Dymola license server on Linux, except the license file, are available in the Dymola distribution. (This also includes the license daemon, where Dymola presently supports FLEXnet Publisher 11.16.2.1.)

As an alternative to FLEXnet, Dassault Systèmes License Server (DSLS) can be used. Dymola 2026x supports DSLS R2026x. Earlier DSLS versions cannot be used.

Note that running the Dymola license server on virtual machines is not a supported configuration, although it might work on some platforms.

' "