

Dymola

Dynamic Modeling Laboratory

Dymola Release Notes

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

Document version: 1

© Copyright 1992-2024 by Dassault Systèmes AB. All rights reserved.
Dymola® is a registered trademark of Dassault Systèmes AB.
Modelica® is a registered trademark of the Modelica Association.

Other product or brand names are trademarks or registered trademarks of their respective holders.

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	5
2	About this booklet	6
3	Dymola 2025x	7
3.1	Introduction	7
3.1.1	Additions and improvements in Dymola	7
3.1.2	New and updated libraries	8
3.2	Developing a model	11
3.2.1	Support for arrays with dimensions that are not known until simulation	11
3.2.2	Multi-lingual support for Dymola	11
3.2.3	Minor improvements	15
3.3	Simulating a model	26
3.3.1	Improved code generation for Modelica functions	26
3.3.2	Scripting	26
3.3.3	Minor improvements	27
3.4	Installation	31
3.4.1	Installation on Windows	32
3.4.2	Installation on Linux	34
3.5	Features under Development	35
3.6	Model Experimentation	39
3.6.1	Minor improvement: Save trajectory for last point sweeps available in GUI, and easier to match result file with sweep	39
3.7	Model Management	40
3.7.1	Encryption in Dymola	40
3.7.2	Extended Git support	41
3.7.3	Improvements in the 3DEXPERIENCE app “Design with Dymola”	42
3.8	Other Simulation Environments	42
3.8.1	Dymola – Matlab interface	42
3.8.2	Real-time simulation	42
3.8.3	Java, Python, and JavaScript Interface for Dymola	44
3.8.4	SSP Support	44
3.8.5	FMI Support in Dymola	45
3.8.6	eFMI Support in Dymola	53
3.9	New libraries	54
3.9.1	TIL libraries	54

3.10	Modelica Standard Library and Modelica Language Specification	65
3.11	Documentation	65
3.12	Appendix – Installation: Hardware and Software Requirements	66
3.12.1	Hardware requirements/recommendations	66
3.12.2	Software requirements	66

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 2025x:

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 66 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 2025x has limited support for the MinGW GCC compiler, 32-bit and 64-bit. For more information about MinGW GCC, see section “Compilers” on page 66, the section about MinGW GCC compiler.

WSL GCC (Linux cross-compiler)

Dymola 2025x has support for the WSL (Windows Subsystem for Linux) GCC compiler, 64-bit. For more information about WLS GCC, see section “Compilers” on page 66, 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 70 for more information. Note that you can use Clang as code generator.

2 About this booklet

This booklet covers Dymola 2025x. 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 2025x

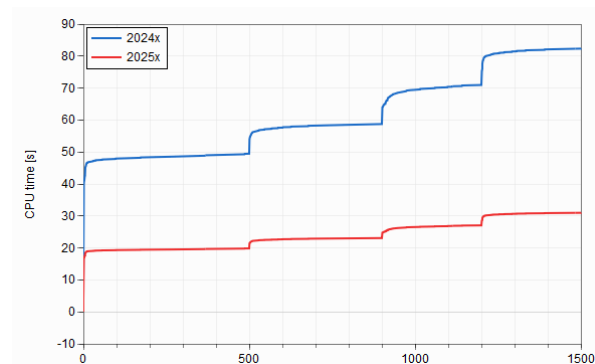
3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2025x. In particular, Dymola 2025x provides:

Improved code generation for Modelica functions

A number of improvements, that for some models with many function evaluations can make the simulation two times faster or more (see page 26):



Support for arrays with dimensions that are not known until simulation

Now a vector or matrix of arbitrary size can be read at simulation initialization into a Modelica parameter. The user benefit is considerable; it allows you to write Modelica models in a new way that does not require the use of external objects for tables (see page 11).

FMI Improvements

- **FMU export: Input interpolation for FMI 2 Co-simulation FMUs** To improve the efficiency of Co-simulation FMUs, smoothing of inputs allows the integrator to continue the simulation without any reset. To further help the integrator, you can also activate predictor compensation (see page 45).
- **FMU export: Variable-size parameter arrays supported in FMI 3 FMUs** Modelica models with variable-size parameter arrays are supported in exported FMUs too. Setting the structural parameter **size** on the parameter then allows the user to set **data** at FMU initialization (see page 47). Only vectors of type Boolean, Real, and Integer are supported, not matrices.

- **FMU import: GUI support for variable step-size Co-simulation** You now have a setting in the GUI that lets you activate variable step-size Co-simulation, to, for example, use small steps in the beginning of a process to handle transitions, and then apply longer steps (see page 49). The feature is available for FMI 2 and 3.

Dymola Modelica Compiler tool

One of features that are formally in a beta state, but available now, is the **Dymola Modelica Compiler** tool, a command line tool without GUI to better support scripting and remote execution in Dymola (see page 35).

SSP 2.0 support

The SSP specification version 2.0 is expected to be released in 2024. Dymola supports a number of 2.0 features – see page 44.

Some minor items

- Multi-lingual support for Dymola (page 11)
- Extended functionality for the Ccode and Ida solvers (page 30)
- Improved handling of multiple versions of a library (page 32)
- Displaying the used Java Runtime Environment location (page 32)
- Save trajectory for last point sweeps available in GUI, and easier to match result file and sweep (page 39)
- Extended Git support (page 41)
- Discontinued support for 32-bit simulation in Linux (page 34)
- Improved Simulation tab GUI (page 27)

3.1.2 New and updated libraries

New libraries

Seven new libraries are available in this Dymola distribution:

- TIL Library
- TIL Adsorption Library
- TIL Automotive Library
- TIL Heat Storage Library
- TIL Hydrogen Energy Systems Library
- TIL NTU Library
- TIL PCM Storages Library

For more information, see “New libraries” starting on page 54.

Libraries in Controlled Availability

The following libraries are now in Controlled Availability; they are replaced by some of the new TIL libraries above:

- Thermal Systems Library
- Thermal Systems Mobile AC Library

Updated libraries

The following libraries have been updated:

- Aviation Systems Library, version 1.6.1
- Battery Library, version 2.8.0
- Brushless DC Drives Library, version 1.4.2
- ClaRa DCS Library, version 1.7.5
- ClaRa Grid Library, version 1.7.5
- ClaRa Plus Library, version 1.7.5
- Claytex Library, version 2024.2
- Claytex Fluid Library, version 2024.2
- Cooling Library, version 1.5.3
- Dassault Systemes Library, version 1.13.0
- Design Library, version 1.2.2
- Dymola Commands Library, version 1.18
- Dymola Embedded Library, version 1.0.4
- Dymola Models Library, version 1.9.0
- Electric Power Systems Library, version 1.6.5
- Electrified Powertrains Library (ETPL), version 1.10.0
- Fluid Dynamics Library, version 2.18.0
- Fluid Power Library, version 2024.2
- FTire Interface Library, version 1.3.1
- Human Comfort Library, version 2.18.0
- HVAC (Heating, Ventilation, and Air Conditioning) Library, version 3.3.0
- Hydrogen Library, version 1.4.1
- Model Management Library, version 1.3.2
- Modelica_LinearSystems2, version 3.0.0
- Multiflash Media Library, see Thermodynamics Connector Library
- Optimization Library, version 2.2.7
- Plot3D Library, version 1.1.3
- Pneumatic Systems Library, version 1.7.1
- Process Modeling Library, version 1.2.0. **Note.** This library is currently not supported on Linux.

- Testing Library, version 1.9.0
- [Thermal Systems Library, in Controlled Availability]
- [Thermal Systems Mobile AC Library, in Controlled Availability]
- Thermodynamics Connector Library (previously named Multiflash Media Library), version 1.2.2. **Note.** This library is currently not supported on Linux.
- VeSyMA (Vehicle Systems Modeling and Analysis) Library, version 2024.2
- VeSyMA - Engines Library, version 2024.2
- VeSyMA - Powertrain Library, version 2024.2
- VeSyMA - Suspensions Library, version 2024.2
- VeSyMA2ETPL Library, version 2024.2
- Visa2Base, version 1.17
- Visa2Paper, version 1.17
- Visa2Steam, version 1.18
- Wind Power Library, version 1.1.5

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 Support for arrays with dimensions that are not known until simulation

You can now handle arrays whose array size is not known until the simulation is run (e.g., they depend on reading external tables). You do this by setting the annotation `annotation(__Dymola_UnknownArray=true);`. This also works in combination with array aliases.

This means that a vector or matrix of arbitrary size can be read at simulation initialization into a Modelica parameter. The user benefit is considerable; it allows you to write Modelica models in a new way that does not require the use of external object for tables.

For more information, please see the new white paper “Parameter Arrays in Dymola (Managing external parameter sets)”. You can find it by the command **Tools > Help Documents**, in the White Paper section.

Limitations:

- Only parameters are supported.
- They are not included in the result file, and can thus not be plotted.
- They cannot be set interactively.
- The total number of array elements is statically limited.
- Only arrays of simple types are supported, not array of records.

3.2.2 Multi-lingual support for Dymola

Introduction

Dymola 2025x has multi-lingual support as defined in the Modelica language specification, allowing you to translate Modelica libraries to other natural languages.

The idea is that you create a translation file that you put in the library package. You have to set a flag to activate the usage of the translation file. Dymola by default use the current language setting of your system.

It is important to note that the texts that are translated are “resulting” texts, that is, texts that cannot be edited (texts that cannot be opened in an editor from where they are seen). Texts that can be edited are not translated, the reason being that those texts you should be able to use when creating a new translation.

Creation of the translation file

You can create a translation file template by the built-in function `generateTranslationFile(className, fileName)` where `className` is the name of the library/package to be translated, and `fileName` is the name of the translation file template to be generated, including the path to it. The translation template name must be the name of the library/package to be translated, with `.pot` as file extension.

You should typically put the template file in the library/package you want to translate, in a folder `Resources/Language`.

The file created is a template. Open it, type the translation for each string, and save it as a language-specific translation file, as `<packageName>.<languageCode>.po`. The language code used is defined according to ISO 639-1.

Example of translation file creation

Let us get started with an example. Create a simple library `MyTestLib`. When creating it, add the description `This is a translation test library`. Also, make sure that the library is not saved as only one file. (This is already the case by default.) In the library, create a model `MyModel1`, with the description `This is a test model`.

Save the library in a suitable location, in this example we use `E\MyExperiments`, and then, in the folder of the library, create the folder `Resources`, with a subfolder `Language`.

To create the translation file template, in the input command line in Dymola, enter:

```
generateTranslationFile("MyTestLib",  
"E:/MyExperiments/MyTestLib/Resources/Language/MyTestLib.pot")
```

This will generate the file `MyTestLib.pot` in the folder `E:\MyExperiments\MyTestLib\Resources\Language\`.

Now, let us assume we want a translation to Swedish (language code `sv`). Open the translation file template using e.g. Notepad, and save it as `MyTestLib.sv.po`. In the file, enter the translation for the two texts, and save the file.

```
1 # Add comments and copyright here
2 #, fuzzy
3 msgid ""
4 msgstr ""
5 "Project-Id-Version: MyTestLib \n"
6 "Report-Msgid-Bugs-To: \n"
7 "POT-Creation-Date: 2024-07-04 13:16+0000\n"
8 "PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
9 "Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
10 "Language-Team: \n"
11 "Language: \n"
12 "MIME-Version: 1.0\n"
13 "Content-Type: text/plain; charset=UTF-8\n"
14 "Content-Transfer-Encoding: 8bit\n"
15
16 msgctxt "MyTestLib"
17 msgid "This is a translation test library."
18 msgstr "Detta är ett översättningstestbibliotek."
19
20 msgctxt "MyTestLib.MyModell"
21 msgid "This is a test model."
22 msgstr "Detta är en testmodell"
23
```

(As can be seen, there are a number of useful lines in the beginning of the text file that can be used to supply information about the translation. However, we concentrate here on the specific descriptions translation.)

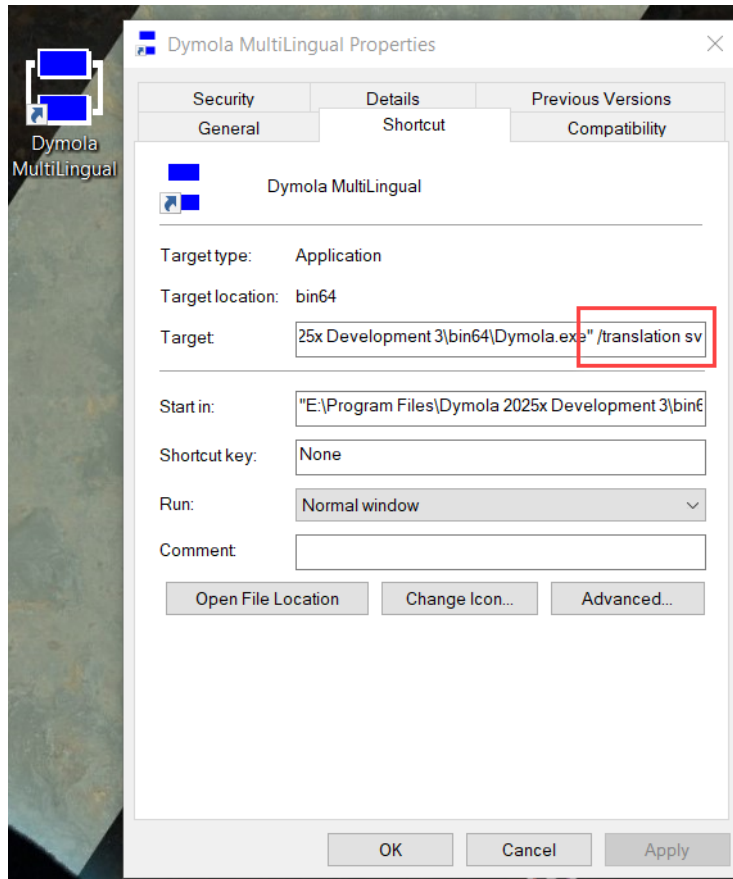
Using the translation file

To use the translation file, that is, to apply the translation of texts in the library, the first thing to consider is the language used in Dymola. By default, the system language is used. If you want to use another language, you must first start Dymola with a specific language command prompt argument `/translation <languageCode>`.

Now you can open the relevant library/package, and then set the flag `Advanced.UI.SupportMultiLingual = true`. Finally, you may need to refresh the library/package.

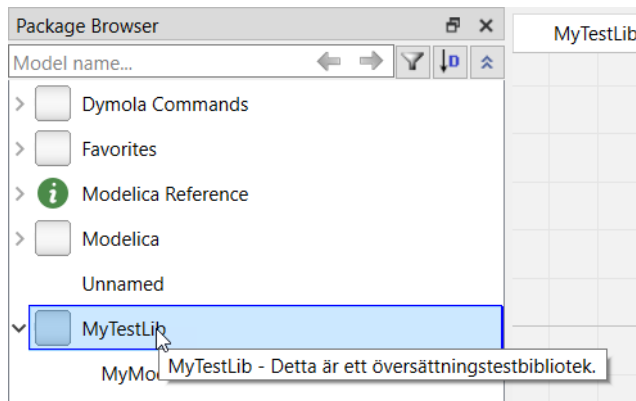
Example of using the translation file

To be able to start Dymola with a language command for the Swedish language (sv), you can create a shortcut to Dymola and then add the language command in the “Target” line:

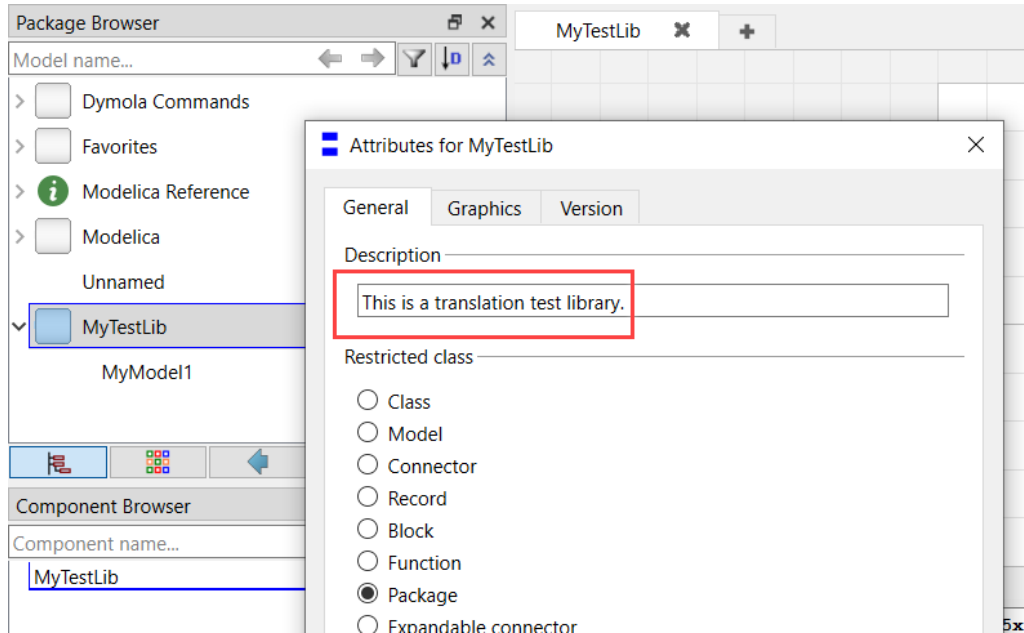


Now, start Dymola with this shortcut, open the package `MyTestLib`, and set the flag `Advanced.UI.SupportMultiLingual = true`.

Refreshing the package (right-click the package in the package browser and select **Refresh**), and then pausing over the package in the package browser now gives:



However, note that right-clicking the package in the package browser and selecting **Attributes** gives:



The reason for this is that the attribute text can be edited, and is therefore not translated.

3.2.3 Minor improvements

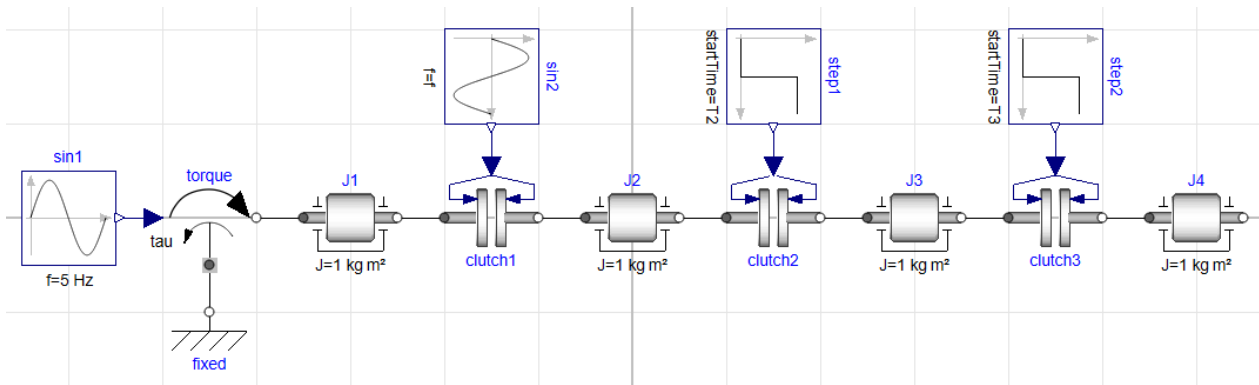
Selective model extension improvements

The selective model extension feature, defined in the Modelica language specification, has been improved:

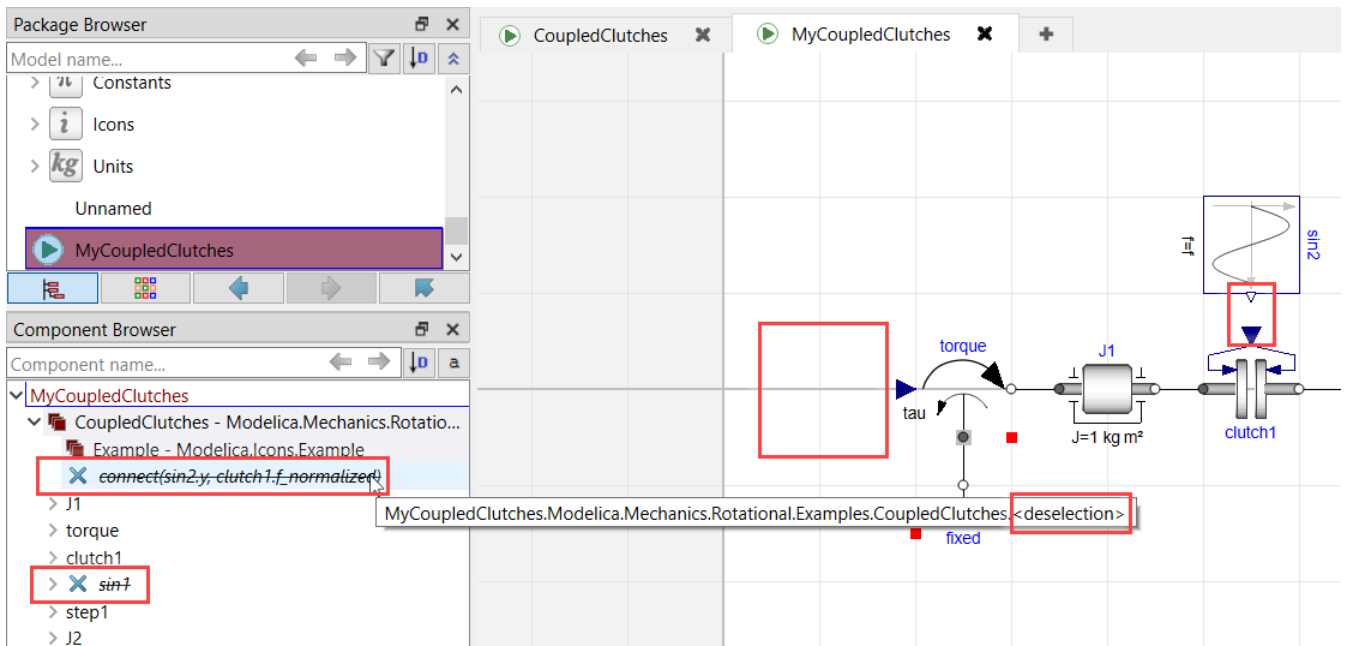
- Deselected connections are now displayed in the component browser.
- Option to regret deselection of connections in the component browser GUI.
- Better visibility of deselected items in the component browser.

A simple scenario to illustrate these improvements:

Open the demo Coupled Clutches and extend from it. Change the size to 150% for easier selection of connections.



Select the component **sin1**, click the **Delete** button, and, in the dialog box that appears, select **Deselect** to deselect it. Do the same for the connection between **sin2** and **clutch1**. The result will be:

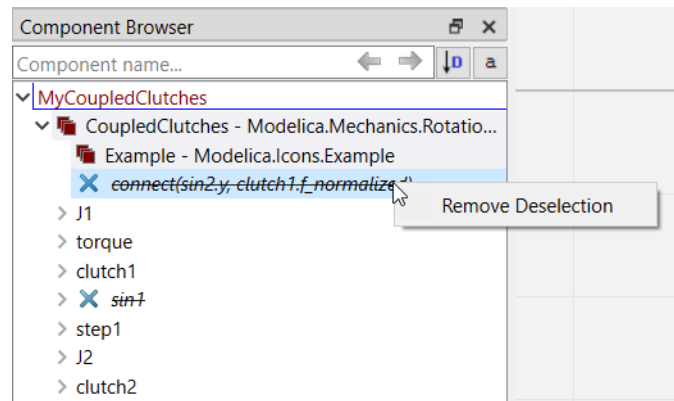


Some notes:

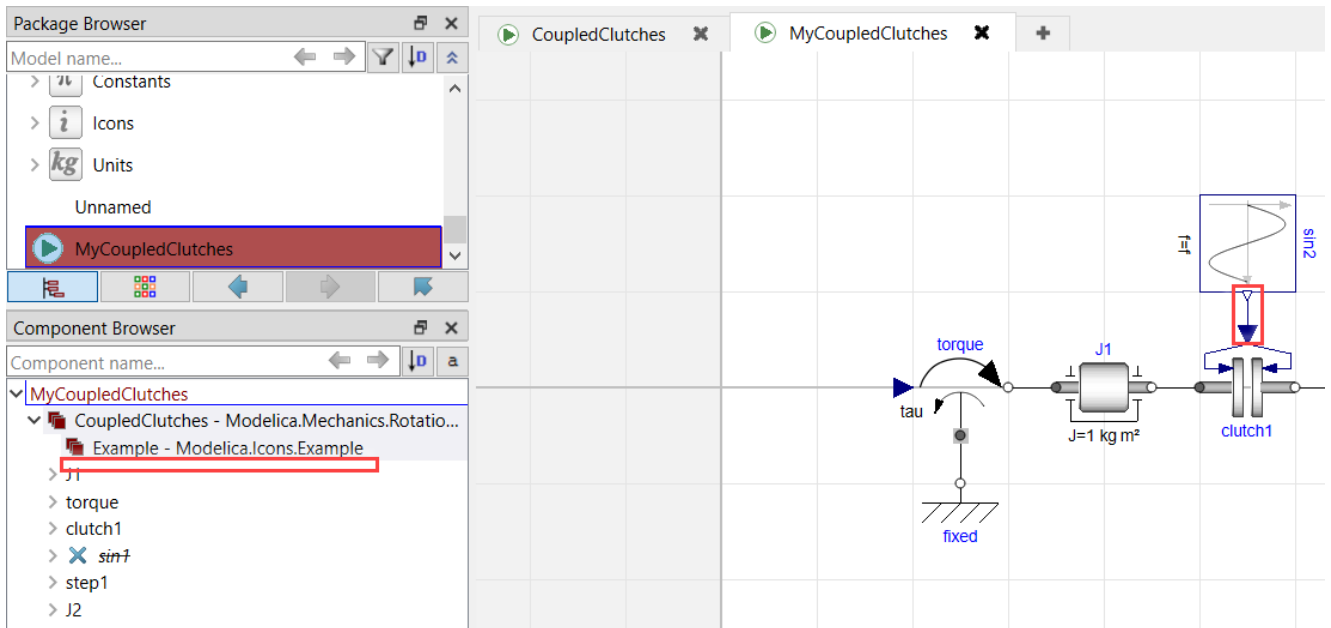
- The deselected connection is now graphically presented in the component browser, with a tooltip. Note that *only* deselected connections are displayed in the component browser.
- The general visibility what items are deselected is improved, by the cross before the item, and the strikethrough of the text.

- You can only apply selective model extension to inherited components. If you have added other components, and you select such a component and click **Delete**, that component is just deleted, without any dialog box appearing.

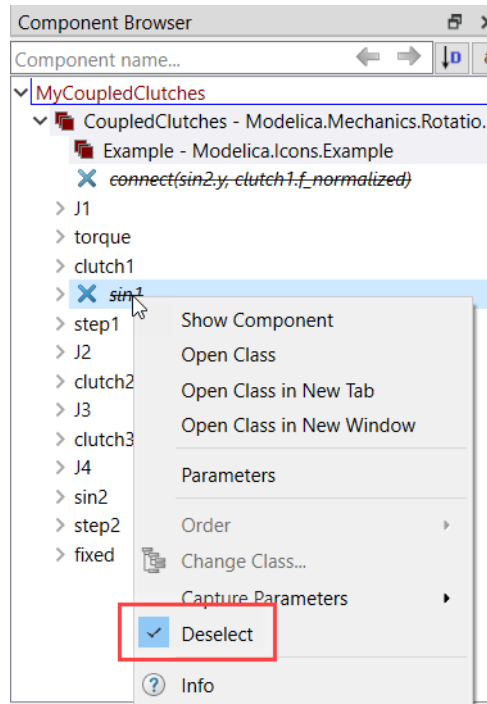
To regret the deselection of the connection, you can right-click it and select **Remove Deselection**:



The result is:



Note that the GUI for regretting the deselection of a *component* has not changed:



To regret the deselection, click **Deselect** in the context menu of the component, to change the status. (The menu entry works in toggle mode.)

Improved efficiency of functions

The code in functions in general has been improved to be more efficient.

Improved unit checking

In general, when-conditions are now unit checked.

Concerning the suggested Modelica language improvement of unit checking (<https://github.com/modelica/ModelicaSpecification/pull/3491>), the following are implemented, in addition to many minor improvements:

- The unit checking consider the value of evaluable parameters, and only consider active equations.
- Units for elements of array variables (when using subscripts) are supported:
 - Getting the unit: All if the array has the same unit for all elements.
 - Fully (getting and propagating): literal (allowing per element unit), for-loops (all elements will have the same unit).

For the second item, you can consider three cases:

- If you have `Real x[:]` (each unit = "m"), then `Real y = x [...]` gives the unit "m" for `y` since all elements in `x` have the same unit.

- If you have `Real z1[:], z2[:];` and `z1[1] = time;` that gives that `z1[1].unit = "s"`.
- If you have a loop `for i in ... loop z2[i] = time;` then you get `z2[:].unit = "s"`.

Further improved support for package-extends

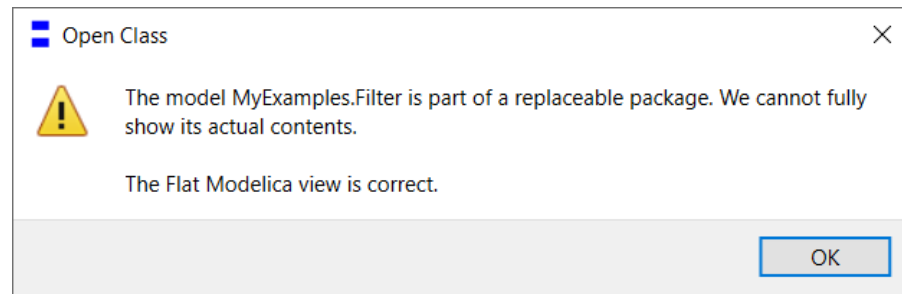
Consider the following package:

```
package MyExamples
  extends Modelica.Blocks.Examples;
  annotation (uses(Modelica(version="4.0.0")));
end MyExamples;
```

The scripting command `translateModel` since previous versions supports translation of models inherited by package extension, including incorporation of their simulation setup. An example is `translateModel("MyExamples.PID_Controller")`.

In Dymola 2024x Refresh 1, some improvements were implemented:

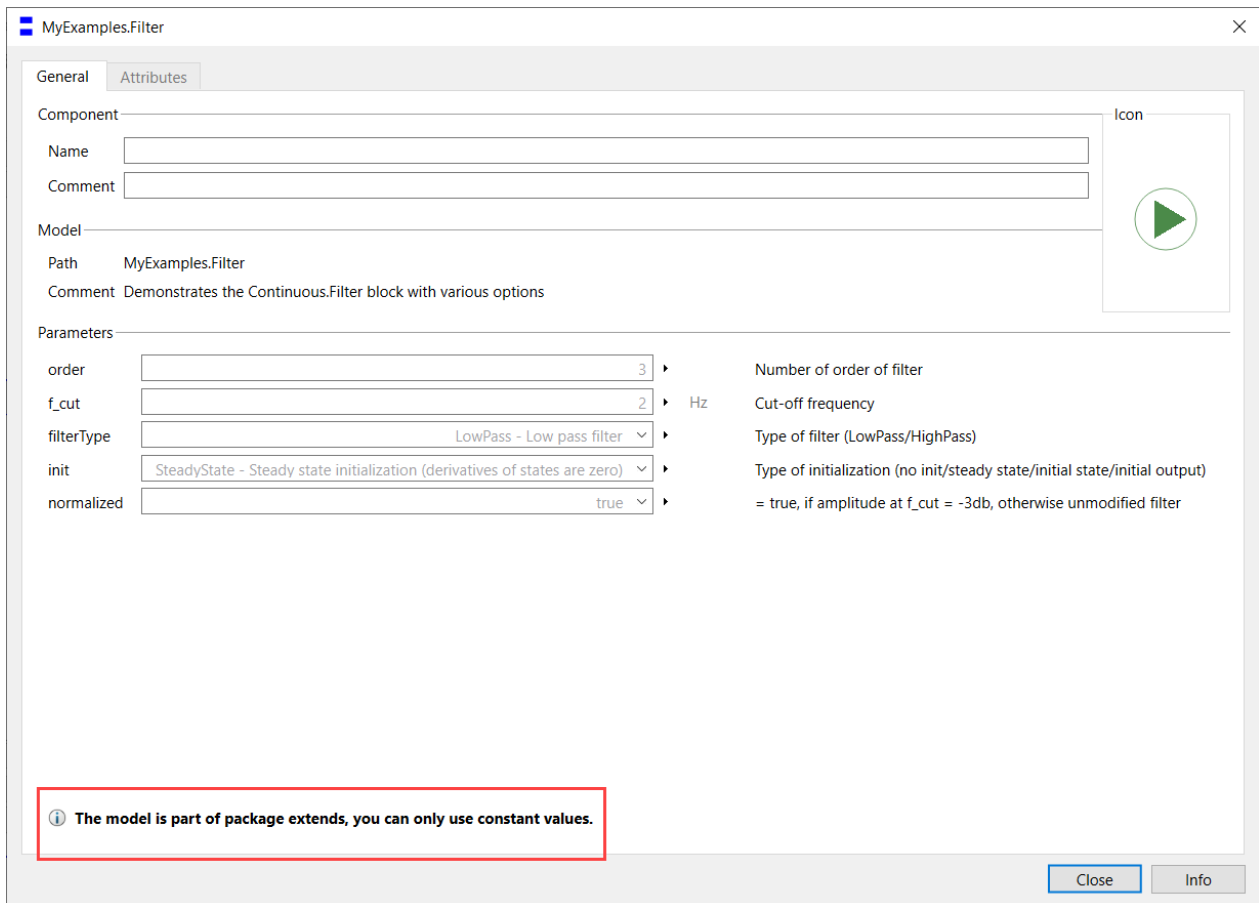
- The display in the package browser was corrected.
- When you selected a model inside the extended package in the package browser, you by default got a warning:



In Dymola 2025x, the support for package-extends have been further improved:

You can now create and edit parameters of models in package-extends also in the diagram layer, and you can also apply the command **Change Class** for replaceable components. (Previously you could only work with the text layer.)

For the parameter values, you can only use constant values. This is indicated when you open the parameter dialog of a model inside the extended package. An example:



The diagram layer now also displays the current values.

The above warning when selecting a model inside the extended package in the package browser is by default now not displayed. To display it, you can set the flag

```
Advanced.UI.WarnIfModelInPackageExtends = true
```

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

Better parsing error handling

Parsing errors have been improved, in some cases syntax errors can be reported for multiple tokens, e.g. the function call `Real z= foo(2 = x) ;` will report an issue for `2 =` where it is unclear if it should be `foo(a=x)` or `foo(2+x)` (or even `foo(function a=x)`).

Additional tokens expanded in text strings

In Dymola 2025x, two more tokens can be expanded in text strings in the diagram or icon layer:

Token	Expanded to
<code>%componentdescription</code>	The description of the component, if one has been specified. Otherwise, the string is empty.
<code>%classdescription</code>	The description of the class, if one has been specified. Otherwise, the string is empty.

Note that curly brackets are allowed when specifying tokens, for example `%{componentdescription}`.

Clarification of the “Copy attributes” setting and flag

If you display the attributes of a class by right-clicking it in the package browser and select **Attributes**, you have, in the **Graphics** tab, a setting that was previously called **Copy attributes to Icon layer**. To better indicate what this setting does, it has now been renamed:

Attributes for Unnamed

General Graphics Version

Attributes for Diagram layer

☒ Copy coordinate system, grid and scale factor to the Icon layer ?

Coordinate system

	Min	Max
Horizontal	-100	100
Vertical	-100	100

Grid

Horizontal	2
Vertical	2

Component scale factor

Scale 0.1

The size of an inserted component is scale · coordinate system.
Leave this field **empty** to use the default component size.

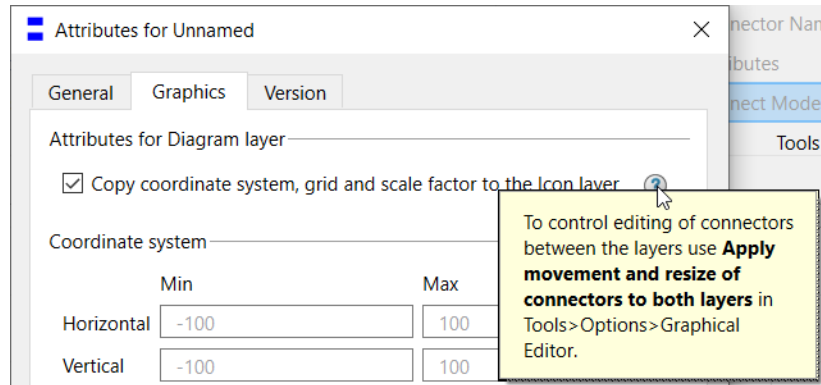
Aspect ratio

☒ Preserve aspect ratio when resizing components of this class

OK Cancel

The description of the corresponding flag `Advanced.Editor.AttributesBothLayers` has been changed accordingly.

Note that this setting/flag does not control copying of connectors between the layers. To do that, you have to use the general setting **Apply movement and resize of connectors to both layers**. You find this setting using the command **Tools > Options**, the **Graphical Editor** tab. This information is also given using the question mark ? :



(The corresponding flag is `Advanced.Editor.MoveConnectorsTogether`.)

Improved grid value calculation

If you, in the figure above, change the coordinate system to e.g. $\{-1, 1\}$ instead of the default $\{-100, 100\}$, then a better (smaller) grid value is now calculated automatically.

Modelica records displayed in the documentation layer

Modelica records are now displayed in the documentation layer. As an example, consider the record:

```
record MyRecord "RecordTest"
  parameter Real T1=2.1 "Simple value";
  parameter Real T2[:, :]={{1.1, 1.2}, {2.1, 2.2}} "Matrix";
  record Rec1_record
    parameter Real T11(
      unit="m/s",
      displayUnit="km/h") = 3.083333333333333 "Scaled value";
    parameter Real T12(unit="m/s") = 12.1 "With unit";
  end Rec1_record;
  Rec1_record Rec1;
end MyRecord;
```

Looking at the documentation layer for this record, you get:

Package Browser

Model name...

- > Dymola Commands
- > Favorites
- > Modelica Reference
- > Modelica
- MyPackage
 - MyModel
 - MyRecord
 - Rec1_record

MyRecord

RecordTest

Contents

Type	Name	Default	Description
Real	T1	2.1	Simple value
Real	T2[:, :]	{{1.1, 1.2}, {2.1, 2.2}}	Matrix
	Rec1_record	Rec1	

Name: MyRecord
Path: MyPackage.MyRecord
Filename: C:/Users/uwm/Documents/Dymola/MyPackage/MyRecord.mo

Clicking the link “**Rec1_record**”, you get:

Package Browser

Model name...

- > Dymola Commands
- > Favorites
- > Modelica Reference
- > Modelica
- MyPackage
 - MyModel
 - MyRecord
 - Rec1_record

Rec1_record

Contents

Type	Name	Default	Description
Real	T11	3.083333333333333	Scaled value [m/s]
Real	T12	12.1	With unit [m/s]

Name: Rec1_record
Path: MyPackage.MyRecord.Rec1_record
Filename: declaration window

Note. There is no indication if the record consists of parameters or variables or both, all items are presented under “Contents”.

Option to display variables in the documentation layer of models

If you set the flag `Advanced.HTML.VariablesInModels = true`, the variables of a model are displayed in the documentation layer. (The flag is by default `false`.)

Consider a simple pendulum model:


```

model Pendulum
  parameter Real m=1;
  parameter Real L=1;
  parameter Real g=9.8;
  parameter Real J=m*L^2;
  Real phi(start=0.1);
  Real w;
  equation
    der(phi) = w;
    J*der(w) = -m*g*L*sin(phi);
  end Pendulum;

```

Now, the default documentation layer (also, what was available in previous versions) is:

Parameters

Type	Name	Default	Description
Real	m	1	
Real	L	1	
Real	g	9.8	
Real	J	m*L^2	

Name: Pendulum

Path: Pendulum

Filename: C:/Users/uwm/Documents/Dymola/Pendulum.mo

Setting the flag above, you instead get:

Parameters

Type	Name	Default	Description
Real	m	1	
Real	L	1	
Real	g	9.8	
Real	J	m*L^2	

Contents

Type	Name	Description
Real	phi	
Real	w	

Name: Pendulum

Path: Pendulum

Filename: C:/Users/uwm/Documents/Dymola/Pendulum.mo

Note. This flag does not affect the display of records as described in the section above.

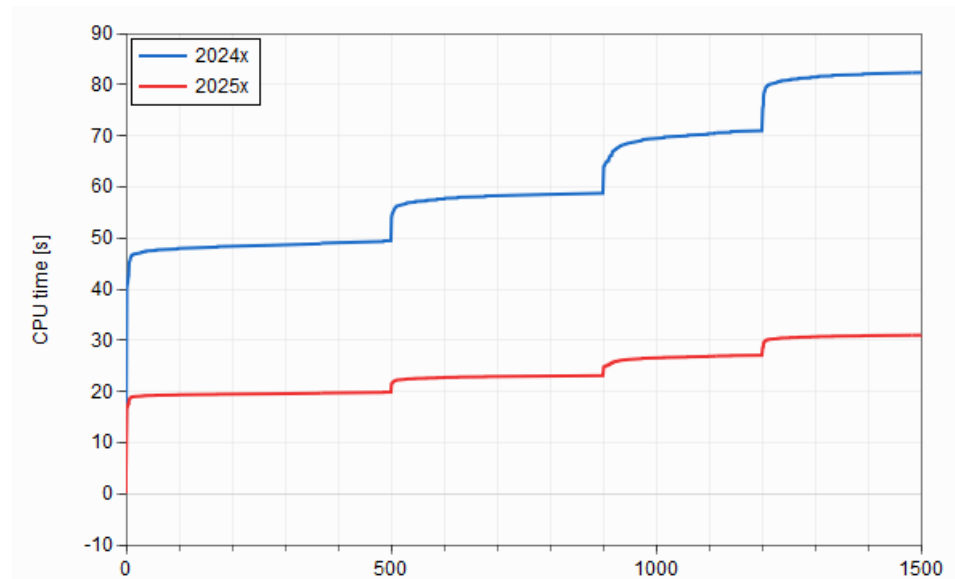
3.3 Simulating a model

3.3.1 Improved code generation for Modelica functions

The code-generation for Modelica functions has been improved:

- Unnecessary checks are reduced.
- Additional constants are computed only once in functions.
- Literal arrays are handled more efficiently.

For some models, there is no effect on the simulation time from these improvements, but for some models with many function evaluations, the simulation can be two times faster, or even more. One example is the VaporCycle from the ThermoFluidsStreams library, available from GitHub:



3.3.2 Scripting

The built-in function `importSSP` improved

The built-in function `importSSP` now has a new Boolean input argument `includeAllVariables`. By default, the argument is `true`. Setting it to `false` means that the FMUs that are included in the imported SSP file are imported as black-box FMUs. (That is, only inputs, outputs, and parameters are included in the imported FMUs.)

The `importSSP` function will now also print its `importFMU` commands as log messages.

The built-in function `importSSV` improved

The built-in function `importSSV` previously had an argument `modelName`. The name is now changed to `baseName` since the function can also create and modify parameter records, and those are not models. For more information, see section “Improved handling of SSV import” on page 45.

The built-in function `RequestOption` improved

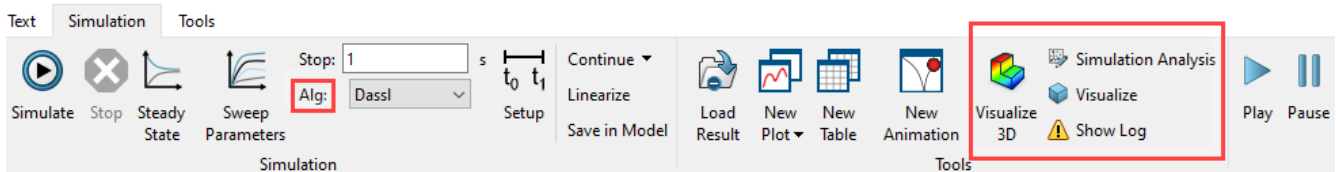
The built-in function `RequestOption` now has a new Boolean input argument `verbose`. By default, the argument is `false`. Setting it to `true` returns additional information if the checking out of the requested option fails. An example:

```
RequestOption("BinaryModelExport", verbose=true)
No such feature exists.
Feature:      BinaryModelExport
License path: C:/Users/uwm/AppData/Roaming/DassaultSystemes/Dymola/dymola.lic;
FlexNet Licensing error:-5,147
= false
```

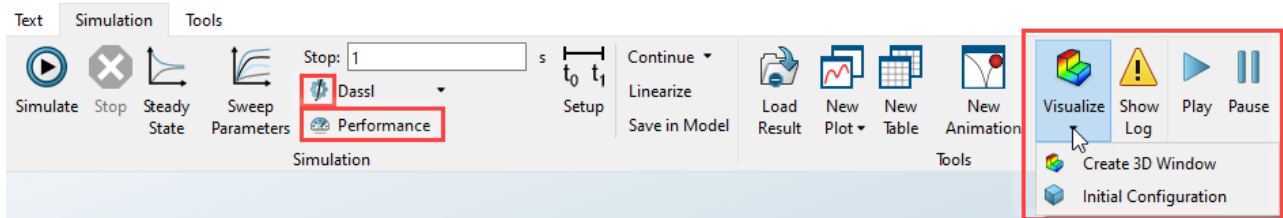
3.3.3 Minor improvements

Improved Simulation tab GUI

The simulation tab has been improved. In previous Dymola versions, part of it looked like:




In Dymola 2025x, the corresponding part looks like:



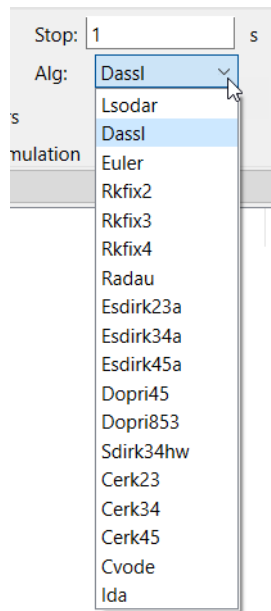
The changes are:

- The command **Simulation Analysis** has been renamed to **Performance** (with a new icon), and have been given a more prominent position in the tab.
- The two **Visualize** commands have been combined to one command button **Visualize** with a pull-down menu. For each subcommand, the name is slightly changed:

- The former command **Visualize 3D** is now named **Create 3D Window**.
- The former command **Visualize** is now named **Initial Configuration**.
- The **Show Log** command can now occupy more space.
- The former text **Alg.** for algorithm selection has been exchanged to the icon . Also, the corresponding menu for algorithm selection has been improved, see next section.

Improved selection of integration method from the Simulation tab

You can select the integration algorithm in the simulation setup, but you can also select it directly in the simulation tab. In Dymola 2024x Refresh 1, the menu to select algorithm directly looked like:

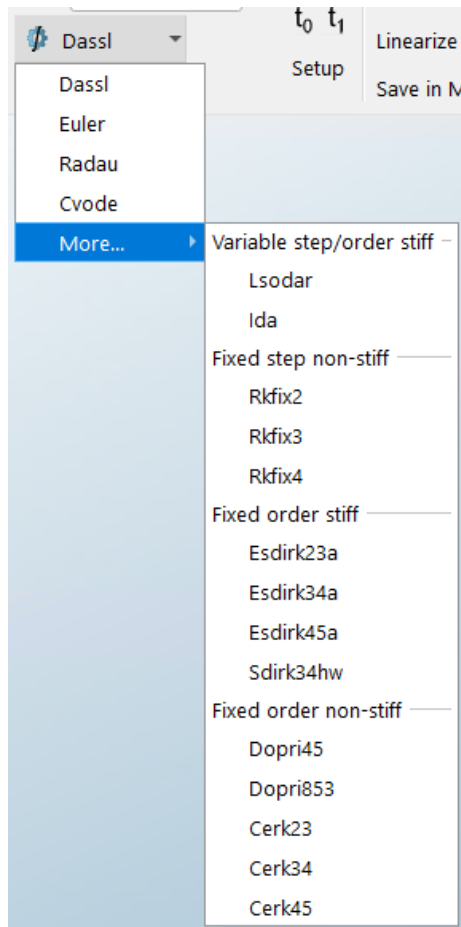


This option has been improved in Dymola 2025x. The menu now looks like:

, in several ways:

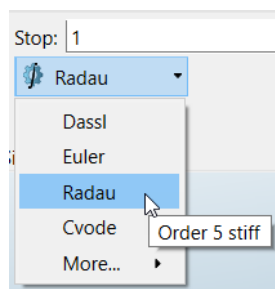
menu has been improved by giving you the most common methods to select from directly, while the more specialized ones are now under a **More...** menu entry, now also grouped under headers of the characteristics (these headers cannot be seen in Windows 10).

In Dymola 2025x, it looks like:

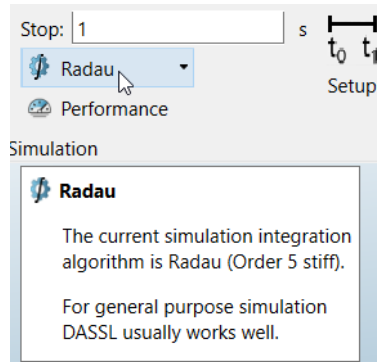



The improvements are:

- The most common algorithms are directly selectable; the others are collected under **More....**
- The algorithms under **More...** are somewhat reorganized, and grouped under headings of the characteristics of the groups.
- There are now tooltips for the selections:



- There are also more information in the tooltip for the selected algorithm:



The text **Alg:** before the selected algorithm has been changed to the icon .

In the simulation setup, reached by the command **Simulation > Setup**, the **General** tab, you still have all the algorithms in one **Algorithm** list. However, the description string has been improved for the following algorithms:

- Lsodar
- Dassl
- Ccode
- Ida

Extended functionality for the Ccode and Ida solvers

Support for analytical Jacobians

You can now use analytical Jacobians also when using the Ccode or Ida solver. You do that by activating the setting **Generate Analytic Jacobian for the ODE problem** in the simulation setup (reached by the command **Simulation > Setup**, the **Translation** tab).

You can use analytical Jacobians both when simulating in Dymola, and when exporting FMUs.

This means that now all implicit solvers in Dymola supports analytical Jacobians. (For the explicit solvers to support Jacobians is not relevant since they miss system Jacobians.)

Support for analyzing numeric integration

The Ccode and Ida solvers now support using the setting **Which states that dominate error** in the simulation setup (reached by the command **Simulation > Setup**, the **Debug** tab).

This means that numeric integration analysis can be applied for models using these solvers. The numeric integration analysis is handled by the command **Simulation > Performance**, the **Numeric Integration** tab. This also means that all solvers now supports all applicable simulator performance features, that is, the features reached by the command **Simulation > Performance**. Notes:

- The feature numeric integration analysis is not applicable to Euler and Rkfix* as they don't use an error estimate.
- Simulator performance was called simulation analysis in previous Dymola versions, see “Improved Simulation tab GUI” on page 27.

Option to use central difference for Jacobian evaluation for the Dassl solver

For the Jacobian matrix in the iteration algorithm in Dassl, you can use a central difference approximation for evaluation, instead of the default forward difference approximation. You activate this option by setting the flag:

```
Advanced.Simulation.CentralDifferenceJacobianDassl = true
```

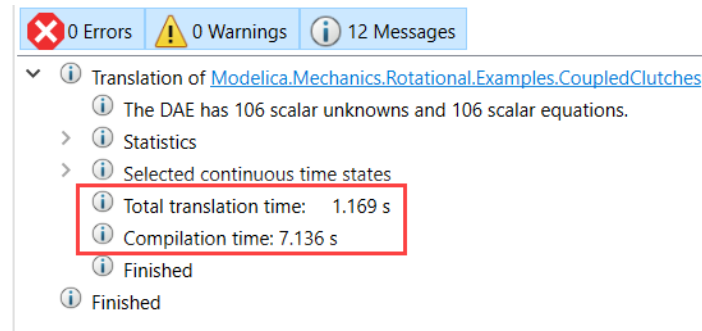
(The flag is by default `false`.)

Using this option can improve the accuracy, but the simulation might become less efficient.

Measuring the time for translating and compiling a model

In Dymola 2025x you can set the flag `Advanced.Translation.Log.Timing = true` to output the time for translating and compiling a model. (The default value of the flag is `false`.)

The times are printed in the translation log:



A typical use can be to compare the times between Dymola versions for a specific model.

Note that the translation time might include reading parts of Modelica libraries the first time the model is translated.

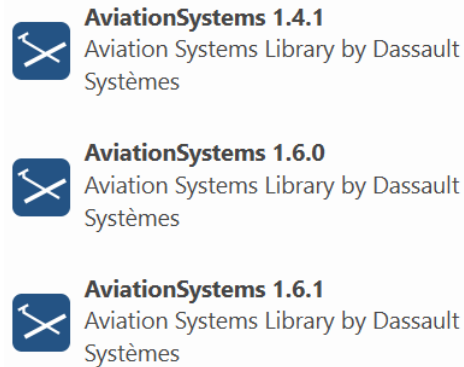
3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 66.

3.4.1 Installation on Windows

Improved handling of multiple versions of a library

The **File > Libraries** menu has been improved to better support users with multiple versions of a library. If there are multiple versions of a library, resulting in duplicated menu entries in the **File > Libraries** menu, Dymola will attempt to disambiguate the versions by appending the version number. For example:



This new functionality can be disabled by setting the flag

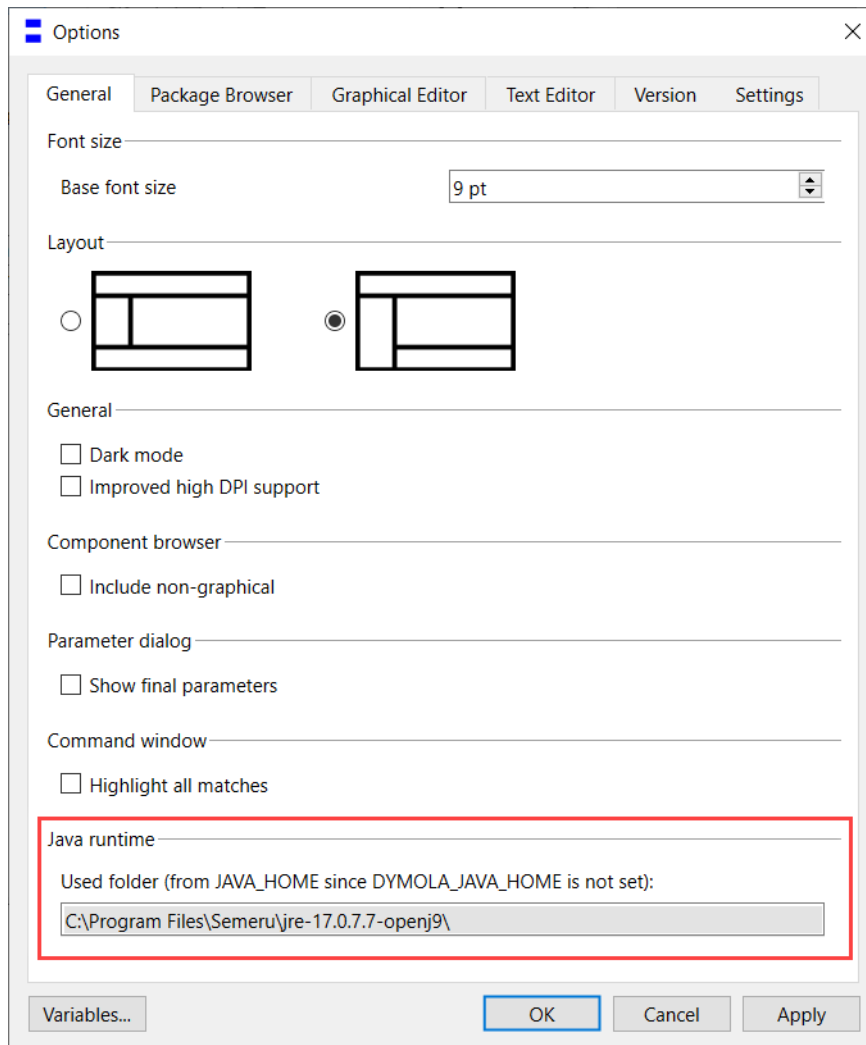
```
Advanced.File.DisambiguateLibraries = false
```

thereby restoring the behavior of earlier Dymola versions.

(The flag is by default `true`.)

Displaying of used Java Runtime Environment

You can see the used Java Runtime Environment by the command **Tools > Options**, the **General** tab:



You can set the Java Runtime Environment folder location to use in two ways:

- The environment variable **DYMOLA_JAVA_HOME**.
- The environment variable **JAVA_HOME**.

Having two environment variables allow you to have more than one Java Runtime Environment installed, and use them for different applications.

For the Dymola related applications Design with Dymola and eFMI, these applications first look if the first one above is set, if not, they look for the other variable. This means that this part of the dialog can look like any of the two images below:

Java runtime

Used folder (from DYMOLA_JAVA_HOME):

C:\Program Files\Semeru\jdk-17.0.7.7-openj9

Java runtime

Used folder (from JAVA_HOME since DYMOLA_JAVA_HOME is not set):

C:\Program Files\Semeru\jdk-8.0.402.6-openj9\

If none of the environment variables is set, the location field is left empty.

If you are to specify the Java Runtime Environment location, use as first choice DYMOLA_JAVA_HOME since this is specific for Dymola and related applications, the other environment variable is a more general one that might be used by other applications (with other version demands) as well.

Updated Qt version

Dymola 2025x is built with Qt 6.7.1. (For Dymola 2024x Refresh 1, the version used was Qt 6.6.3.)

3.4.2 Installation on Linux

Improved handling of multiple versions of a library

See the corresponding section for Windows above.

Discontinued support for 32-bit simulation

From this Dymola version, Dymola 2025x, the 32-bit simulation on Linux is no more supported. This means:

- No more dependencies to the 32-bit libc.
- Discontinued support for 32-bit FMUs.

(Note that the support for 32-bit simulation on Windows is continued in Dymola 2025x.)

Updated Qt version

Dymola 2025x is built with Qt 6.7.1. (For Dymola 2024x Refresh 1, the version used was Qt 6.6.3.)

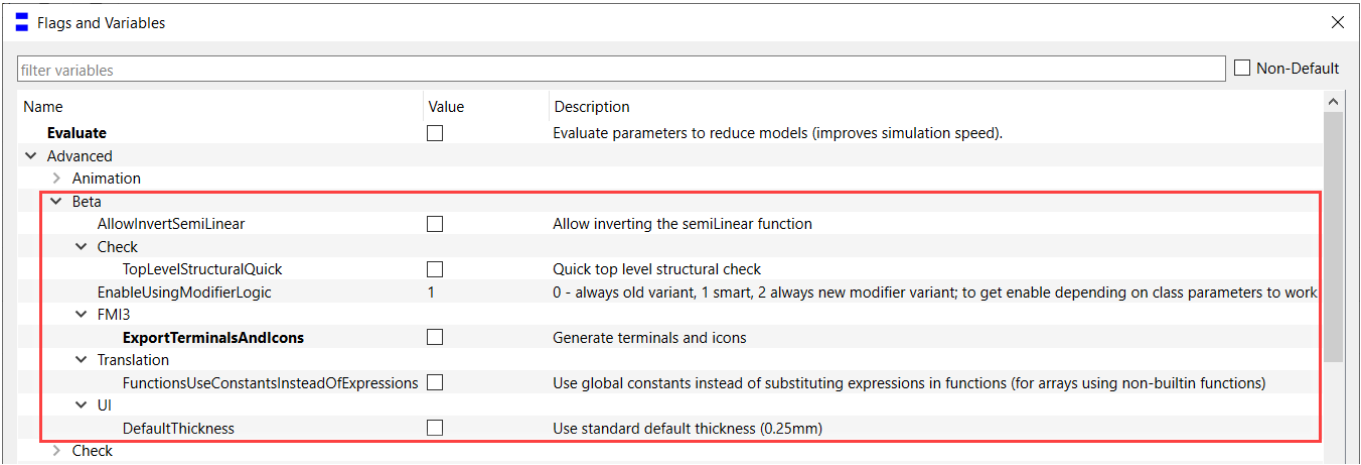
3.5 Features under Development

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 2025x:



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 2025 x, 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`.

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
<code>-h</code>	Prints a summary of available operations.
<code>-o file-name</code>	Opens the named Modelica file into DMC.
<code>-x command</code>	Runs the given command line. For example: Dos: <code>dmc -x "simulateModel('\"Foo\"', stopTime=2);"</code> Linux: <code>dmc -x 'simulateModel("Foo", stopTime=2);'</code> (For Linux, the shell-quoting may differ between Linux versions.)
<code>-r file-name</code>	Runs a script file (.mos)
<code>-p port</code>	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 2025x 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, `kernel=True` needs to be set even when providing the path to the DMC executable.

```
dymola = DymolaInterface("C:/Program Files/Dymola  
2025x/bin64/dmc.exe", kernel=True)
```

(This feature appeared in Dymola 2025x (this release).)

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 (this release).)

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

FMI 3.0 introduces a new file to support terminals and icons. Dymola can use this to store input and output variables. You can export terminals and icons in FMUs by setting the flag `Advanced.Beta.FMI3.ExportTerminalsAndIcons = true`. (The flag is by default `false`.)

(This feature appeared in Dymola 2023x Refresh 1.)

For arrays using non-built-in functions, an option to use global constants instead of substituting expressions in the functions

This is a part of improving the code generation for Modelica functions. This option may result in an even faster function execution. However, it might require additional start values in the model. To use this option, set the flag `Advanced.Beta.Translation.FunctionsUseConstantsInsteadOfExpressions = true`. (The default value of the flag is `false`.)

(This feature appeared in Dymola 2025x (this release).)

Option to use default thickness according to specification

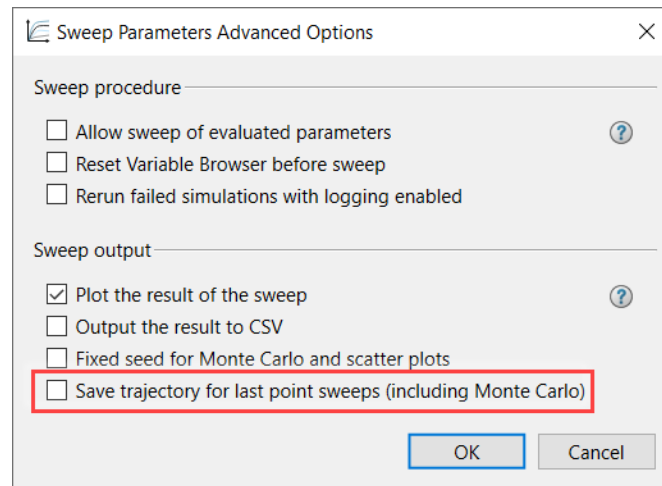
There is a conclusion in the Modelica Association that the default line thickness should always be 0.25. In Dymola, the default line thickness depends on the context. To adapt to the standard, you can set the flag `Advanced.Beta.UI.DefaultThickness = true`. (The flag is by default `false`.)

(This feature appeared in Dymola 2025x (this release).)

3.6 Model Experimentation

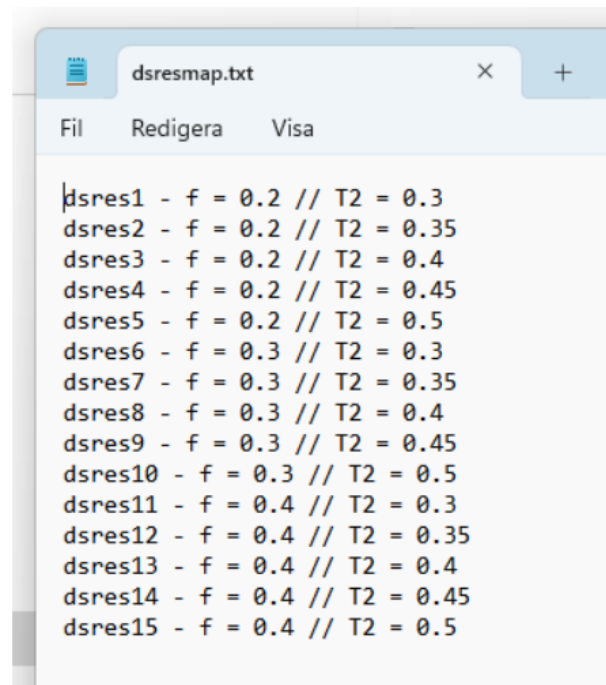
3.6.1 Minor improvement: Save trajectory for last point sweeps available in GUI, and easier to match result file with sweep

In Dymola 2025x, you can activate the feature to save the trajectory for the last point sweeps (including Monte Carlo) by a new setting in the sweep parameters advanced options dialog. You can reach this dialog by the command **Simulation > Sweep Parameters**, and then clicking the **Options** button in the dialog that appears:



If you activate this setting and run the sweeps, the result files are saved as `dsres*`, where `*` is a number. An additional text file `dsresmap.txt` is created; detailing which `dsres` file corresponds to which simulation. An example

dsres1.mat
dsres2.mat
dsres3.mat
dsres4.mat
dsres5.mat
dsres6.mat
dsres7.mat
dsres8.mat
dsres9.mat
dsres10.mat
dsres11.mat
dsres12.mat
dsres13.mat
dsres14.mat
dsres15.mat
dsresmap.txt



```
dsres1 - f = 0.2 // T2 = 0.3  
dsres2 - f = 0.2 // T2 = 0.35  
dsres3 - f = 0.2 // T2 = 0.4  
dsres4 - f = 0.2 // T2 = 0.45  
dsres5 - f = 0.2 // T2 = 0.5  
dsres6 - f = 0.3 // T2 = 0.3  
dsres7 - f = 0.3 // T2 = 0.35  
dsres8 - f = 0.3 // T2 = 0.4  
dsres9 - f = 0.3 // T2 = 0.45  
dsres10 - f = 0.3 // T2 = 0.5  
dsres11 - f = 0.4 // T2 = 0.3  
dsres12 - f = 0.4 // T2 = 0.35  
dsres13 - f = 0.4 // T2 = 0.4  
dsres14 - f = 0.4 // T2 = 0.45  
dsres15 - f = 0.4 // T2 = 0.5
```

Note that the feature of saving the trajectory for last point sweeps was available also in previous versions, by opening the underlying function call, and, in the integrator part, selecting Advanced, and activating the Write trajectory option.

However, the text file `dsresmap.txt` is new for Dymola 2025x. It is created also if you use the Write trajectory option in the underlying function call.

3.7 Model Management

3.7.1 Encryption in Dymola

Standardized Modelica library encryption: Minor improvements

Support for opening `.mol` files with the command **File > Libraries** or via the script `libraryinfo.mos`

You can open an encrypted library `.mol` file by the **File > Libraries** command. You can also include `.mol` files in the script `libraryinfo.mos`, meaning that those encrypted libraries are automatically loaded when the corresponding top package is opened.

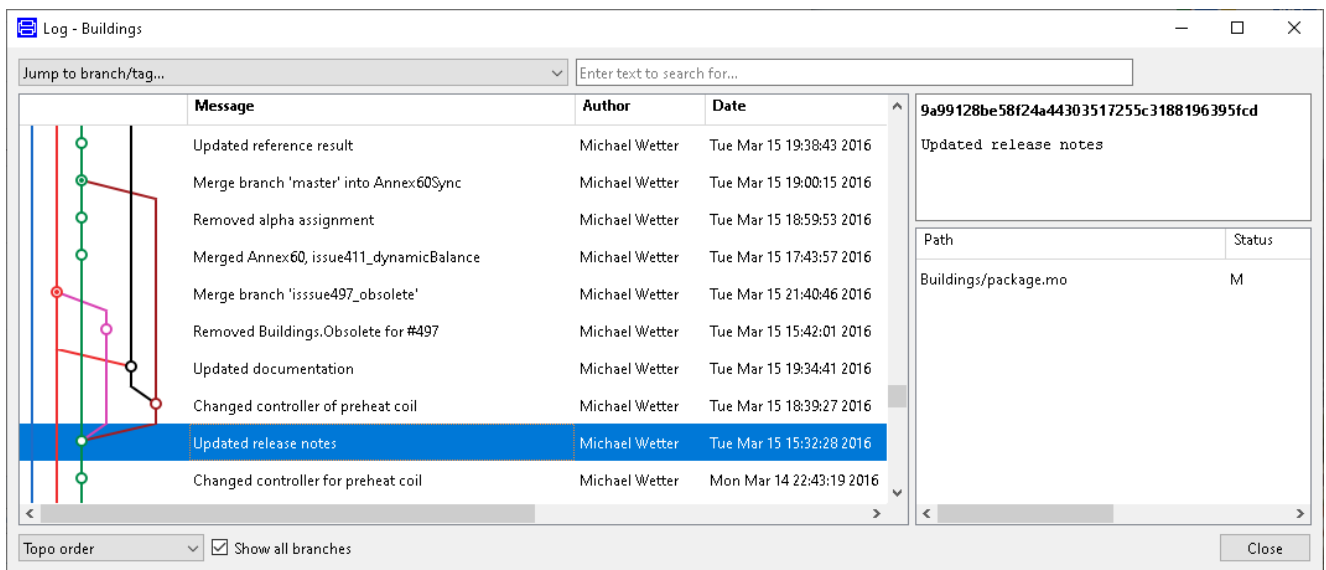
This also means that you can have `.mol` files in package folders, just like `.mo` and `.moe` files.

3.7.2 Extended Git support

Graphical visualization of the versioning log

The context command **Version > Log**, available when right-clicking a library or model in the package browser, listed in previous Dymola versions the versioning history only as text. In Dymola 2025x, the log also contains graphical visualization of the versioning history, including support for presenting branch history graphically.

The log is displayed in a separate window. Each commit is a line; the branching history is displayed by a graph. An example:

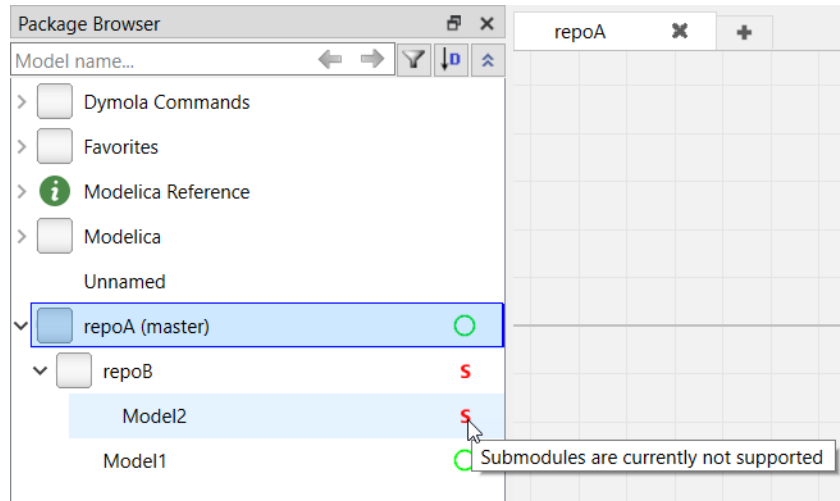


Minor improvement: Better handling of Git submodules

It is possible to store Modelica packages in Git submodules, that is, as example, you may have the top-level Modelica package in a Git module, and Modelica sub-packages in Git submodules.

In previous Dymola version, you could not open a Modelica package or model from a Git submodule. In Dymola 2025x you can open them, and you can also edit them and store them locally.

However, Dymola can currently not use any Git versioning commands on Modelica models or packages in Git submodules. This is indicated by a red **S** for such models and packages. An example:



Pausing over the **S**, you get the information tooltip as seen in the above image.

3.7.3 Improvements in the 3DEXPERIENCE app “Design with Dymola”

For this version, the only new feature is the displaying of the used Java Runtime Environment in the Options dialog, see “Displaying of used Java Runtime Environment” on page 32.

3.8 Other Simulation Environments

3.8.1 Dymola – Matlab interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2019a (ver. 9.6) up to R2024a (ver. 24.1). 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.

3.8.2 Real-time simulation

Compatibility – dSPACE

Dymola 2025x officially supports the DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2025x generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases:

- dSPACE Release 2019-A with Matlab R2019a
- dSPACE Release 2019-B with Matlab R2019b
- 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 R2023a and R2023b
- dSPACE Release 2024-A with Matlab R2023a, R2023b, and R2024a

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_rtmp_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_rtmp_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_usrtrcmmerge` 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 R2019a (Simulink Real-Time ver. 6.10) to R2024a (Simulink Real-Time ver. 24.1). Only Microsoft Visual C compilers have been tested.

3.8.3 Java, Python, and JavaScript Interface for Dymola

New or improved built-in functions available

A number of new and improved built-in functions are available in the interfaces.

For more information, see the corresponding sections in “Scripting” starting on page 26.

3.8.4 SSP Support

Support for SSP 2.0

The SSP specification version 2.0 is expected to be released in 2024.

The following features are supported by Dymola:

- Support for FMI 3:
 - Basic types, such as `Float32` (mapped to the closest Modelica type, in this case `Real`).
 - Scheduled execution (accepted in the SSP file, but scheduled execution is not supported by Dymola so a warning is given).
 - Local variable defined at connector with `kind="local"`.
 - Clocked signals (partial - not in SSP but for FMI 3 FMUs).
 - Array connectors (partial - not in SSP but for FMI 3 FMUs).
 - Optional support for exporting the rotation attribute (which is not standard SSP 2.0).
- Optional inner coordinates of system connector (when inner view is different from outer view).
 - By default, the coordinates are mapped to both icon and diagram layer (as is also the default in Modelica).
 - If `systemInnerX` and `systemInnerY` are specified, they will be used for the diagram layer in Dymola.
- Initial meta-data support from "SSP Traceability" project.
- Support for representing Modelica components and connectors in SSP.

Support for black-box import of FMUs when importing an SSP file

The built-in function `importSSP` now has a new Boolean input argument `includeAllVariables` that activates black-box import of included FMUs when set to `false`. See also section “The built-in function `importSSP` improved” on page 26.

Support of extended MIME attributes for Modelica types

In previous Dymola versions, the MIME attribute `text/x-modelica` was used to indicate models or connectors represented by Modelica types. The source attribute was used to designate the model name. This is not entirely supported by SSP, especially the extension to add a source attribute to binary types.

In SSP 2.0 and Dymola 2025x, an extended MIME path attribute is used instead, as specified in RFC 6838 (see <https://www.rfc-editor.org/rfc/rfc6838>). It is now possible to add additional attributes as

```
text/x-modelica; path=Modelica.Electrical.Interfaces.Pin
```

This is now the new behavior.

Improved handling of SSV import

The built-in function `importSSV` had in previous versions an argument `modelName`. This was the name of the model that the parameters were applied to.

In Dymola 2025x, the argument name is changed to `baseName`, since the function now also can be used to create a parameter record, or to modify an existing record. For more information, please see Chapter 3 in the new white paper “Parameter Arrays in Dymola (Managing external parameter sets)”. You can find it by the command **Tools > Help Documents**, in the White Paper section.

3.8.5 FMI Support in Dymola

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

FMI Export: FMU input interpolation for FMI 2 Co-simulation FMUs

Introduction

Dymola 2025x supports optional FMU input interpolation for Co-simulation FMUs. By smoothing any continuous-time Real input to the FMU, the integrator is allowed to continue the simulation without any reset. The result is potentially improved efficiency. To further help the integrator when smoothing inputs, you can also activate predictor compensation.

The smoothing is supported for FMU Co-simulation export using any of the solvers:

- Ccode or Ida
- Dymola solver Dassl, Lsodar, Euler, or Rkfix* (Dymola solver Ccode or Ida are not supported.)

The predictor compensation is supported for Ccode, Ida and the Dymola solver Dassl.

The features are also available for FMU source code export.

Important: These features are currently only supported for FMI version 2.

These features have been developed in collaboration with TLK-Thermo.

Input smoother

To activate the input smoother, set the flag `Advanced.FMI.UseInputSmoother = true`. (The flag is by default `false`.)

With the flag activated, exported Co-simulation FMUs with any of the solvers specified in the item list in the introduction above will include the input smoother. This means that any

continuous-time Real input will be smoothed. Discrete inputs like Integers, Booleans, and discrete-time Real inputs are not affected by the input smoother, but are handled normally.

When setting a smoothed input, the new value is not immediately set, but stored inside the FMU, to be used in the next `doStep` call. When any or all the inputs are set and `doStep` is called, the stored inputs will then be used. Internally, the FMU will ramp up all the smoothed inputs from the input values set before the previous `doStep`, to the input values set before the current `doStep`. The input values set before the current `doStep` are thus reached at the end of the `doStep`.

Linear interpolation is used and therefore continuous, piecewise linear input signals are achieved. Due to the increased smoothness of the input signals, the integrator is allowed to continue the simulation without any reset. The benefit is potentially larger step-sizes, fewer model evaluations, and fewer Jacobian computations. In summary, this means potentially improved efficiency.

The downside is that the smoothed input signals are delayed. Effectively, the delay is about half a communication interval as we are ramping up to the new inputs during the step. Therefore, the input smoother should be used with care when a co-simulation set-up is sensitive to delays.

When the input smoother is enabled, the FMU feature `canInterpolateInputs` is set to `false` on the exported FMU. This means that when you import such an FMU, you are not allowed to call `fmi2SetRealInputDerivatives` for that FMU; that is, you are not allowed to set the flag `Advanced.FMI.SetInputDerivatives = true` in a model that uses any such FMU. (If you do anyway, the call will fail with an error, and the simulation terminates.)

Predictor compensation

For the solvers `Ida`, `Cvode` and the Dymola solver `Dassl`, if you have activated the input smoother for FMU export, you can also activate a predictor compensation to further help the integrator.

To activate the predictor compensation, set the flag `Advanced.FMI.UsePredictorCompensation = true`. (The flag is by default `false`.)

Without input smoothing, the integrator will be re-initialized at the beginning of any `doStep` after any input has changed value. However, with the input smoother this is not done if only continuous-time Real inputs have gotten new values. Although such inputs are smoothed to piecewise linear, continuous signals, this is not enough smoothness for optimal integrator performance.

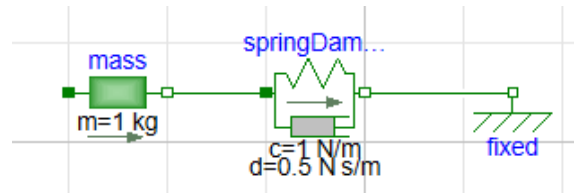
To alleviate the problem, the integrator predictor can be modified using predictor compensation. The improved predictor results in better error estimates and helps in solving for the next integrator step. In general, we may see even larger step-sizes, even fewer model evaluations, and even fewer Jacobian computations.

The downside is that the predictor compensation requires additional model evaluations that may offset the above benefits. These additional evaluations are listed as a separate item in the simulation log if `fmi_loggingOn` is enabled.

This feature is available for Ccode, Ida, and the Dymola solver Dassl. (It may be noted that Euler and Rkfix* could not benefit from any predictor compensation since they have no predictor.)

Test case

As a simple test, consider the mass-spring-damper system with one mass and one spring-damper component.



The model is split into a co-simulation problem with the mass in one FMU and the spring-damper in the other. The input smoother is only applied to the mass FMU, as the other FMU does not contain any continuous states.

Simulation statistics from the mass FMU (the solver used is Ccode):

Feature	Number of f-function evaluations	Number of Jacobian-evaluations	Predictor compensation f-function evaluations
Normal Co-simulation	5471	499	-
Input smoother	2708	40	-
Input smoother and predictor compensation	853	14	998

FMI export: Support for variable-size parameter arrays in FMI 3 FMUs

Models with variable-size parameter arrays can be exported as FMUs. Setting the structural parameter **size** on the parameter then allows the user to set **data** at FMU initialization. The file name can be specified as a string parameter of the FMU.

The feature is supported for FMI 3. Only vectors of type Boolean, Real, and Integer are supported, not matrices.

The feature is by default activated, but can be disabled by setting

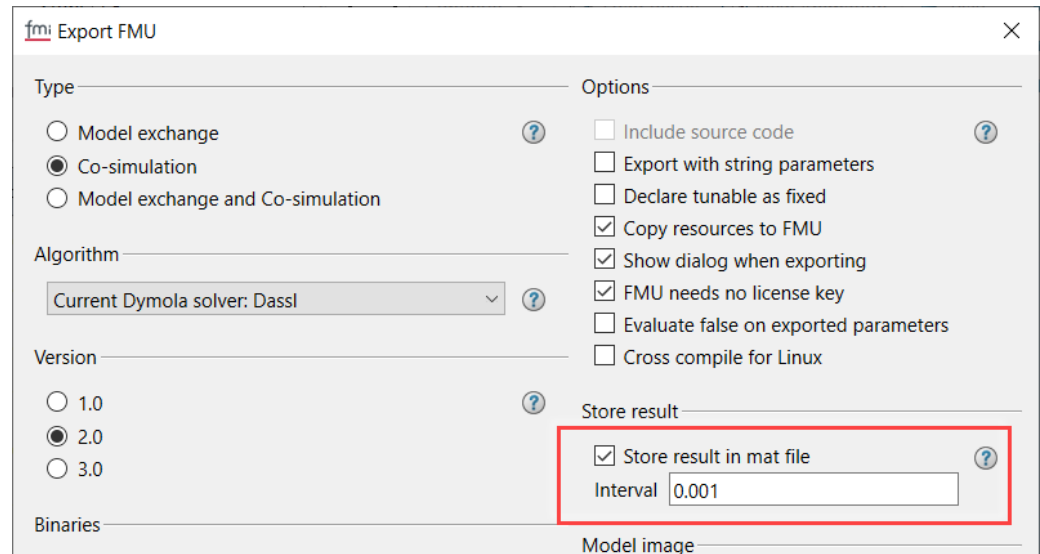
```
Advanced.FMI3.ExposeDynamicArrays = false
```

(The flag is by default `true`.)

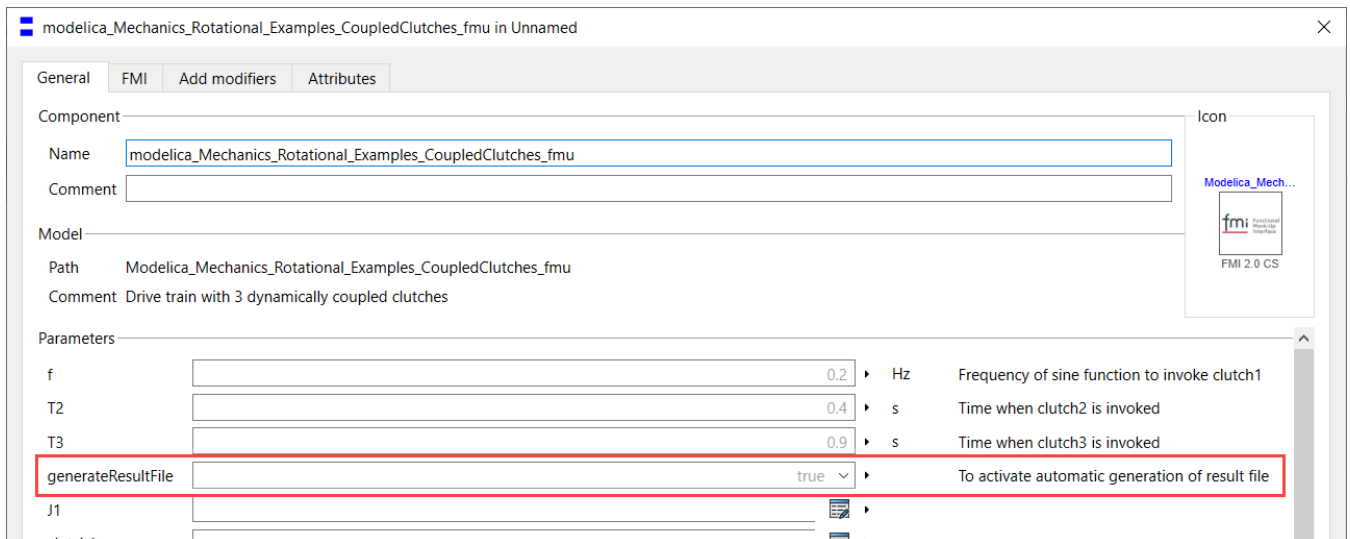
For more information, please see Chapter 4 in the new white paper “Parameter Arrays in Dymola (Managing external parameter sets)”. You can find it by the command **Tools > Help Documents**, in the White Paper section.

FMI export: Option to disable result file generation for an FMU

If you had activated the setting to store the result in a .mat file when you created the FMU, this was in in previous Dymola versions hard coded in the generated FMUs.



In Dymola 2025x, for FMI 2 and FMI 3, if you have activated this setting (as above or when using `translateModelFMU`) you get this setting as a parameter in the generated FMU:



This means that you can select to not generate result files when executing the individual instantiated FMUs. This can be an advantage when you don't want to generate result files for subsystems unless you want to debug them.

FMI import: GUI support for variable step-size Co-simulation

There is a need for variable communication interval for co-simulation, to, for example, use small steps in the beginning of a process to handle transients, and then apply longer steps.

From Dymola 2024x, you can change `dostepTrigger` to a variable that can be externally triggered. You can do this by setting the flag `Advanced.FMI.ExternalDoStepTrigger = true`. (The flag is by default `false`.)

In Dymola 2025x, this flag is also available as the setting **Enable variable communication interval** in the GUI when importing an FMU by the command **File > Open > Import FMU...** You must select **Co-simulation** as Preferred type:

fmi Import FMU

FMU file

Browse...

Name of imported FMU's model

Preferred type

☐ Model exchange

☒ Co-simulation

Options

☐ Prompt before replacing an existing Modelica model

☐ Generate graphics for the diagram layer

☐ Translate value reference to variable name

☒ Structured declaration of variables

☒ Enable variable communication interval

Insert in package

Variables to import

☒ All variables

☐ Black box (parameters, inputs, outputs)

☐ These variables:

Select...

OK Cancel

(By default, the setting is not activated.)

Important. The feature `canHandleVariableCommunicationStepSize` must be activated in the FMU that is imported to be able to activate variable step communication. In Dymola, this feature is by default activated in an exported FMU.

This feature is available for FMI 2 and FMI 3.

Having imported an FMU with this option activated, the `doStepTrigger` is available as a parameter `fmi_DoStepTrigger` in the parameter dialog of the imported FMU, in the **FMI** tab, like:

modelica_Mechanics_Rotational_Examples_CoupledClutches_fmu in Unnamed

General FMI Add modifiers Attributes

Instance name

fmi_instanceName "Modelica_Mechanics_Rotational_Examples_CoupledClutches_fmu" ▶

Enable logging

fmi_loggingOn false ▶

Step time

fmi_DoStepTrigger sample(fmi_StartTime, fmi_CommunicationStepSize) ▼

fmi_StartTime 0.0 ▶

fmi_StopTime 1.5 ▶

fmi_NumberOfSteps 500 ▶

fmi_CommunicationStepSize (fmi_StopTime - fmi_StartTime)/fmi_NumberOfSteps ▶

stepSizeScaleFactor 1 ▶ Number of doSteps called between two CommunicationStepSize

Example

For an example, we will assume that we have an exported FMU of the CoupledClutches demo, which has the `canHandleVariableCommunicationStepSize` capability (because we exported the FMU from Dymola).

To be able to import the FMU and use variable step size, use the command **File > Open > Import FMU...**, browse for the CoupledClutches FMU, select **Co-simulation** as Preferred type, and activate **Enable variable communication interval**. The dialog can look like:

fmi Import FMU

FMU file

Name of imported FMU's model

Preferred type
☐ Model exchange
☒ Co-simulation

Options
☐ Prompt before replacing an existing Modelica model
☐ Generate graphics for the diagram layer
☐ Translate value reference to variable name
☒ Structured declaration of variables
☒ Enable variable communication interval

Insert in package

Variables to import
☒ All variables
☐ Black box (parameters, inputs, outputs)
☐ These variables:

Now, consider the equation $\text{der}(x) = 1 - x$; as the model we use to determine the `doStepTrigger` for the Coupled Clutches FMU.

```

model TestingExternalDoStepTrigger
  Boolean trigger(start=false, fixed=true);
  Real x(start=0, fixed=true);

  Modelica_Mechanics_Rotational_Examples_CoupledClutches_fmu fmu(
    fmi_DoStepTrigger = change(trigger));
equation
  der(x) = 1 - x;
  when {time < 0.5 and sample(0.0, 1/500), sample(0.5, 1/20), x >= 0.6} then
    trigger = not pre(trigger);
  end when;
end TestingExternalDoStepTrigger;

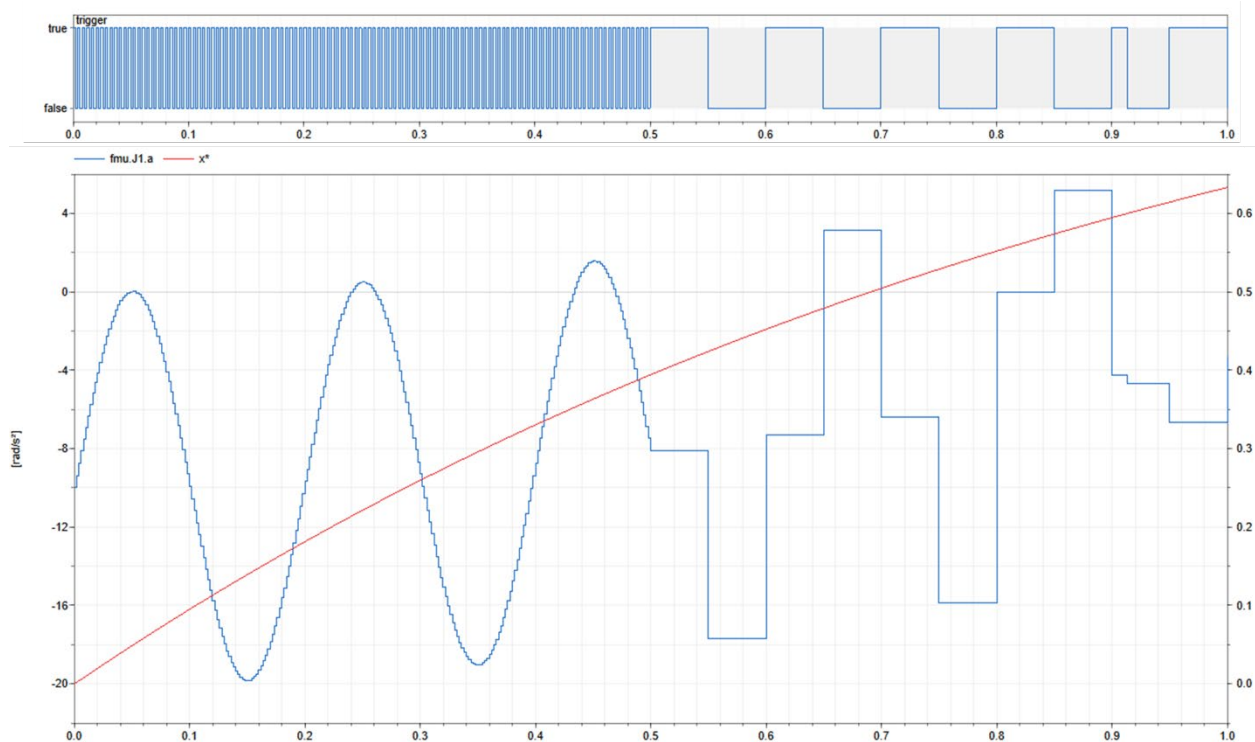
```

For the Coupled Clutches simulation, this model will enable fast sampling in the beginning and slow sampling at the end, along with a state-dependent step. The idea here is to change dynamically the `fmi_DoStepTrigger` variable depending on the events of the simulation. This is achieved by the `change()` function, which returns `true` whenever the `trigger` variable is changed, which happens inside the `when` statement.

To take steps at specific time events, we can use the `sample` function. In our example model, `sample(0.0, 1/500)` will return `true` and trigger a time event every 1/500 seconds, such that in the first half of the simulation, the condition `time < 0.5` and `sample(0.0, 1/500)` becomes `true`. The result is that the FMU takes a step at time instants 1/500 apart at time between 0 and 0.5. By similar logic, after 0.5 seconds, we trigger a step every 1/20 seconds, with `sample(0.5, 1/20)`.

We can also trigger a step by checking state events of the co-simulated model, like `x > 0.6`.

The plot of the simulation below shows these triggers and their effect on the signal **fmu.J1.a** from the FMU.



Support for FMI 1 will be discontinued in a future release

The support for FMI 1 will be discontinued in a future release of Dymola.

3.8.6 eFMI Support in Dymola

The following improvements have been implemented:

Algorithm Code

- Support for untypical, error-case start values in manifests and initialization (NaN, $\pm \infty$)
- Support for many more GALEC builtin functions (3D interpolation, all integer division and remainder variants, ...)

Behavioral Model

- Derived experiment-packages now support boolean and integer GALEC block-interface in- and outputs.

Production Code (backed by Software Production Engineering on 3DEXPERIENCE)

- Support for production code checks with Cppcheck and clang-tidy
 - MISRA C:2012, MISRA C:2023, SEI CERT C Coding Standard
 - Strict preconfigurations, with very few exceptions (exceptions for clang-tidy: identifier length, magic numbers, and altera rules for FPGA & CUDA programming)
 - HTML log with syntax highlighting for Cppcheck results
 - Profiles for open source and premium Cppcheck

An example of production code check:

Cppcheck report - [project name]

error

warning

portability

performance

style

information

cppcheck

clang-tidy

File:

Filter:

Defect summary

☒ Toggle all

Show # Defect ID

☐ 39 premium-misra-c-2023-8.7

☐ 36 misra-c2023-8.7

☒ 5 premium-cert-dcl39-c-SPE_interpolation1D

☒ 5 premium-cert-dcl39-c-SPE_interpolation2D

☒ 4 premium-misra-c-2023-10.1

☒ 2 premium-cert-int30-c

☒ 1 checkersReport

☒ 1 misra-c2023-14.1

☐ 1 premium-licenseExpiresSoon

94 total

Statistics

Line	Id	CWE	Severity	Message
	checkersReport	information	Active checkers: 504/1035	
	E:\modelica-libraries\Dymola-embedded-libraries\DymolaEmbedded\Resources\BuiltinFunctions\galec.stl.c			
89	premium-misra-c-2023-10.1		style	Operands shall not be of an inappropriate essential type.
589	premium-misra-c-2023-10.1		style	Operands shall not be of an inappropriate essential type.
622	premium-misra-c-2023-10.1		style	Operands shall not be of an inappropriate essential type.
651	premium-misra-c-2023-10.1		style	Operands shall not be of an inappropriate essential type.
1421	misra-c2023-14.1		style	A loop counter shall not have essentially floating type
1730	premium-cert-dcl39-c-SPE_interpolation1D		style	Avoid information leakage when passing a structure across a trust boundary
1749	premium-cert-int30-c		warning	Ensure that unsigned integer operations do not wrap
1750	premium-cert-dcl39-c-SPE_interpolation1D		style	Avoid information leakage when passing a structure across a trust boundary
1775	premium-cert-dcl39-c-SPE_interpolation1D		style	Avoid information leakage when passing a structure across a trust boundary
1793	premium-cert-dcl39-c-SPE_interpolation1D		style	Avoid information leakage when passing a structure across a trust boundary
1802	premium-cert-dcl39-c-SPE_interpolation1D		style	Avoid information leakage when passing a structure across a trust boundary
1916	premium-cert-dcl39-c-SPE_interpolation2D		style	Avoid information leakage when passing a structure across a trust boundary
1946	premium-cert-int30-c		warning	Ensure that unsigned integer operations do not wrap

Created by Cppcheck 2.4.9.0.1 (Sourceforge, RC)

```

1742         if (((Integer) 1) == ep_saturated) /* no interpolation needed */
1743         {
1744             result = SPE_interpolation1D(
1745                 error_signal
1746                 , x2
1747                 , x2_data
1748                 , nx2_saturated
1749                 , sy_data[last * x2_size] <--- Ensure that unsig
1750                 , in_saturated <--- Avoid information leakage wh
1751                 , x2_size
1752                 , NULL);
1753     }

```

General user-interface

Generating Algorithm Code containers will not change anymore the settings in the **Output** tab of the simulation setup (reached by the command **Simulation > Setup**, the **Output** tab). In particular, it will not set **Write variables to result file only at stop time**.

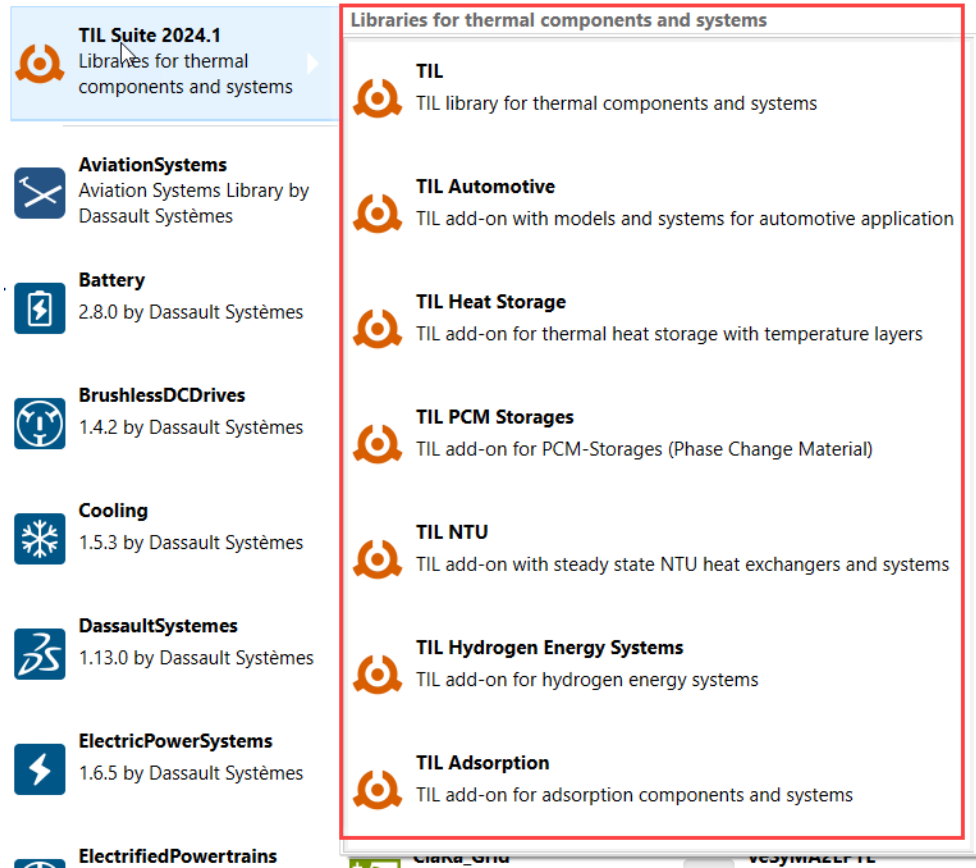
Improved feedback when linking an **3DEXPERIENCE** account via `DymolaEmbedded.UsersGuide.Requirements.link_3DEXPERIENCE_account()`; a success message is printed in the Commands window, any error causes the function to fail and the remote service now can be additionally tested for availability. Likewise, `DymolaEmbedded.UsersGuide.Requirements.linked_3DEXPERIENCE_credentials()` now can check availability of the **3DEXPERIENCE** service. The new `DymolaEmbedded.UsersGuide.Requirements.unlink_3DEXPERIENCE_account()` function can be used to delete any existing linking.

3.9 New libraries

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

3.9.1 TIL libraries

Seven new libraries are included in this Dymola distribution. In the **File > Libraries** menu, they are collected under **TIL Suite 2024.1**:



TIL Suite

Starting with Dymola 2025x, ThermalSystems libraries are replaced by TIL Suite libraries.

The libraries in the suite seen in the menu above are listed below.

Notes:

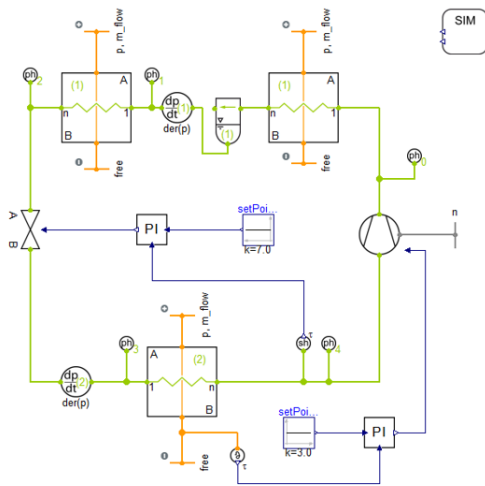
- There are libraries in the suite that are not displayed in the menu, but automatically opened when needed. Examples are TIL Media Library and TIL Cabin Library.
- The names in the package browser when opening a library may not be exactly the same as the library name in the menu.
- For conversion of existing customer models and license, see section “Conversion of existing customer models” on page 63.
- For information of the corresponding Dymola products, that is, the commercial packaging of the below libraries, see “Dymola products for the TIL Suite” on page 63.

TIL Base Library

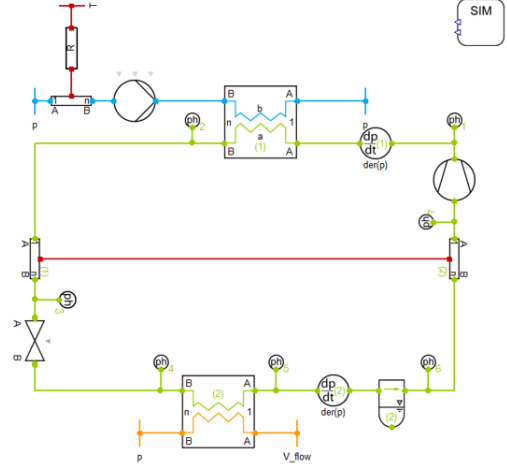
The TIL Base library consists of highly versatile and numerically optimized component models. Thanks to the included substance property library TIL Media, simulations with various media can be performed. The library is intended for the stationary and transient simulation, design and optimization of large and complex systems such as:

- Refrigeration cycles, including refrigeration mixtures
- Heat pump systems e.g. with ejectors
- Hydraulic networks
- Rankine cycles
- Heating, ventilation and air-conditioning systems
- Ab- and adsorption systems (see also TIL Adsorption Library)
- Fuel cell systems (see TIL Hydrogen Library)

Air Conditioning Cycle



Heat Pump System



In addition to the components already included in TIL Base, numerous Add-On libraries are available which allow for more extensive or detailed simulations:

- TIL Automotive Library
- TIL Heat Storage Library
- TIL Phase Change Material (PCM) Storages Library
- TIL Number of Transfer Units (NTU) Library
- TIL Hydrogen Energy Systems Library
- TIL Adsorption Library

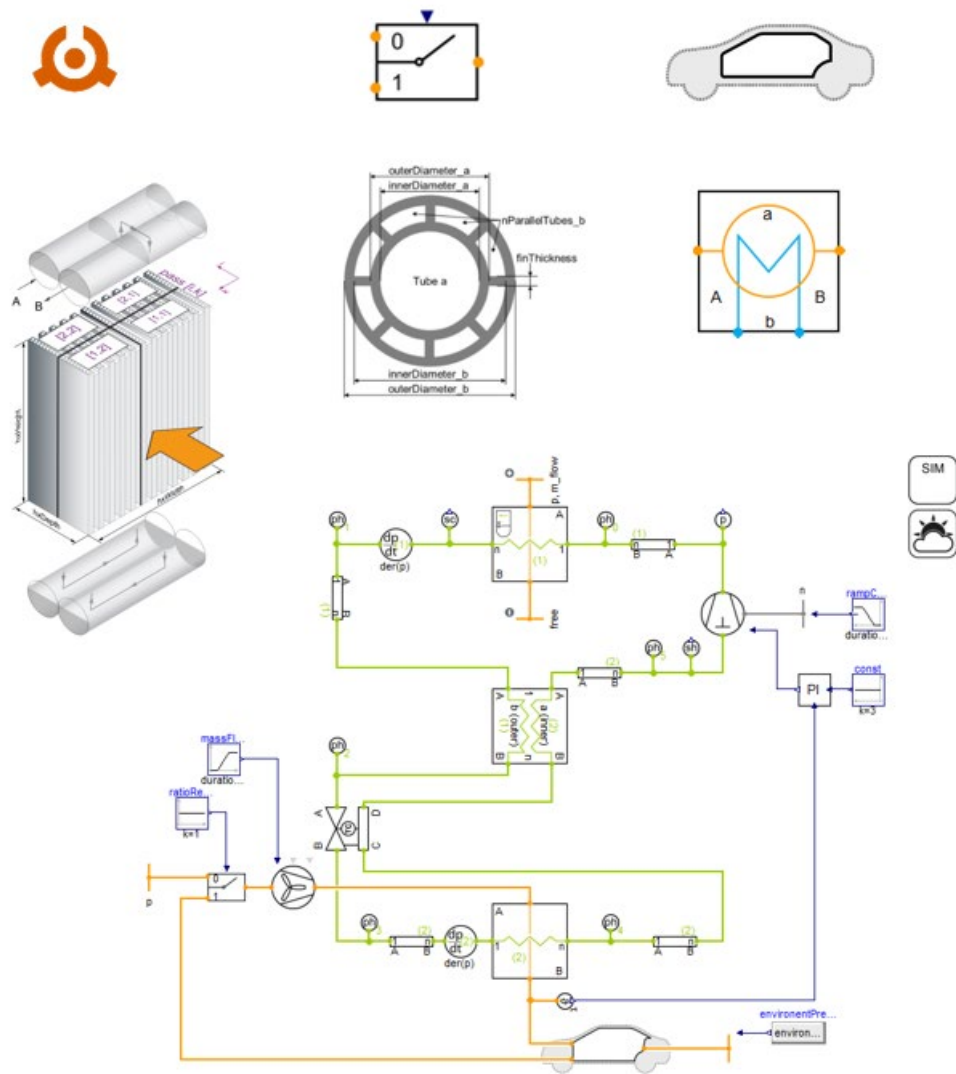
Model details can be viewed in the Modelica code. The library follows the object-oriented concepts of inheritance and polymorphism, allowing the extension of existing models and simplifying the implementation of custom components.

Directly simulatable testers are available not only for single component models, but also for complex systems. The clear visual language of the model icons, together with the documentation and well-structured GUIs of the model components, further simplify the implementation of your own systems.

TIL Automotive Library

The TIL Automotive library is an Add-On to the TIL Base library and focuses on mobile air conditioning systems. It provides additional models for:

- Car, coach and train cabins
- Detailed MPET heat exchanger
- Internal heat exchanger
- Common example systems for mobile AC cycles with different refrigerants



The Cabin models enable the energy balancing of e.g. a car's single air volume, its walls, windows and built-in components. Furthermore, inner and outer thermal effects, the air conveyance separated in fresh and circulated air and the moisture ratio are considered. Besides that, the impact of passengers, e.g. the water production, as well as the human comfort level (PMV/PPD) can be calculated. In addition, drive cycles and environmental conditions for transient simulations are available.

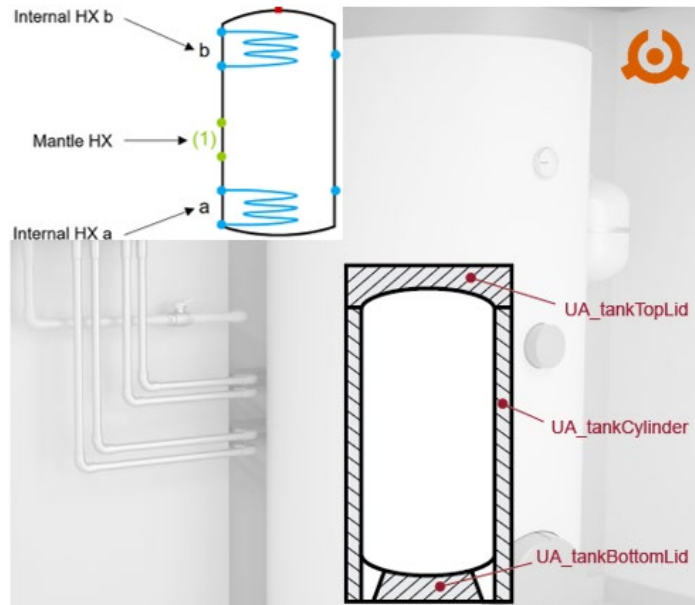
The detailed Multiple Port Extruded Tube (MPET) heat exchanger model enables the discretization of each pass along the refrigerant flow direction and orthogonal to the flow direction. The model can have multiple layers, and the flow pattern through the passes is configurable by the user. The detailed MPET model provides ports on the air side which can

be used for inhomogeneous air temperature and mass flow rates. The optional receiver model makes this heat exchanger applicable to evaporator and condenser geometries.

The internal heat exchanger is a TubeAndTube model, which provides a simple interface to enter the IHX geometry.

TIL Heat Storage Library

The TIL Heat Storage library is an Add-On to the TIL Base library with hot water tanks and examples. The hot water storage tank has stratified temperature layers and is used for example in residential heat pump systems.



The thermal storages and subcomponents are parametrized with geometry information, heat transfer and pressure drop models. The hot water storage tank has optional internal and mantle heat exchangers and detailed models for buoyancy, walls and insulation. In addition, the number and position of the fluid ports and temperature sensors in the tank can be parametrized individually by the user.

TIL Phase Change Material (PCM) Storages Library

The TIL PCM Storages library is an Add-On to the TIL Base library with different types of geometries and fluid combinations for thermal PCM storages respectively heat exchangers.



PCM storages utilize the properties of solid-liquid equilibrium (SLE) media, such as cold ice storage. The storages are parametrized with geometry information, heat transfer and pressure drop models. The PCM storages respectively heat exchangers can be used in in Liquid or VLEFluid systems together with TIL components. PCM storages have an optional external heat port to connect and model the ambient heat transfer.

TIL Number of Transfer Units (NTU) Library

TIL NTU Library is an Add-On to the TIL Base Library. It includes steady state heat exchangers using a “Number of Transfer Units” (NTU) method and example systems.

The NTU heat exchangers can be used for extremely fast steady state simulations. A simple parametrization with one overall heat transfer value, a fixed pinch point temperature difference or a defined transferred heat flow rate is possible.

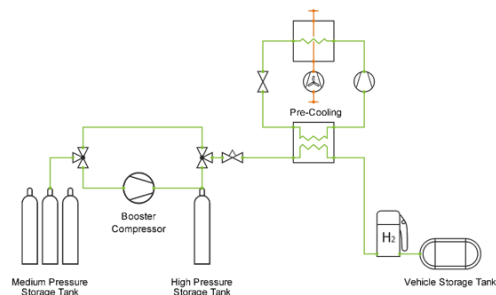
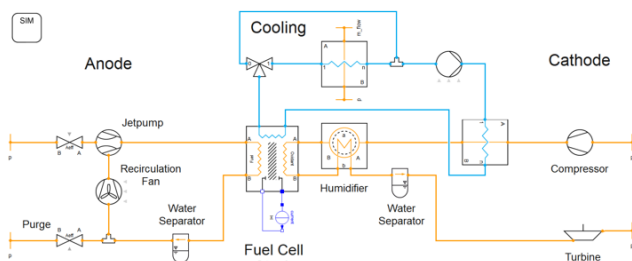
TIL Hydrogen Energy Systems Library

The TIL Hydrogen Energy Systems library is an Add-On to the TIL Base library. The Add-On Hydrogen Energy Systems can be used to model and simulate systems along the entire hydrogen value chain (production, storage and distribution, application).

Fuel Cell Systems



Hydrogen Fueling Stations



H2 production and utilization:

- Models for the analysis of electrolysis processes (SOEC, rSOC)
- Ideal chemical reactors, e.g. for methanation
- Other power-to-X processes

H2 storage and distribution:

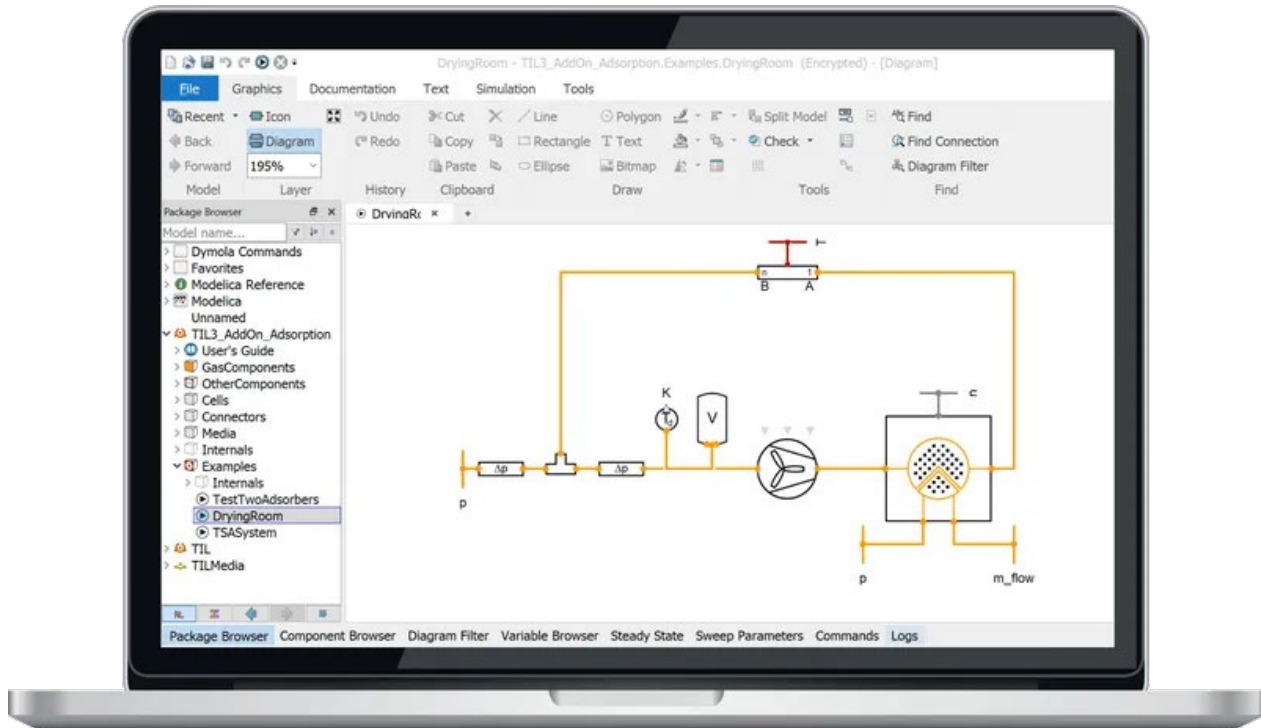
- Hydrogen storage processes, e.g. compressed gas storage
- Liquefaction according to the Linde process
- Hydrogen refueling stations and vehicle storage systems for the simulation of H2 refueling processes (components and example systems)

H2 application:

- Polymer Electrolyte Membrane fuel cells
- Solid Oxide fuel cells
- Entire fuel cell systems including peripheral components of the supply paths, e.g. humidifiers and flow machines, as well as cooling

TIL Adsorption Library

TIL Adsorption is a specialized model library for the simulation and analysis of adsorption processes for drying applications, gas separation or direct air capture. All models are based on the physical principle of adsorption. The library can be individually extended with material data for different adsorbents and offers a complete tool chain to optimize the process design based on simulation and small-scale experiments.



TIL Adsorption includes:

- Universal basic adsorption equilibrium model
- Several isotherm models such as Dubinin, Langmuir, Toth and Sips
- Media models of various adsorbent-adsorptive pairings
- Models for simulating adsorbents with adsorption-desorption hysteresis and multicomponent adsorption
- Component models for adsorbers and various adsorption wheels
- Sample systems, including temperature swing adsorption (TSA) system, dry room and breakthrough curve calibration

TIL Adsorption can:

- Optimize operating strategy to minimize energy consumption and maximize regeneration efficiency
- Help to understand the influence of different cycle parameters, pressure drop and different operating conditions
- Evaluate different regeneration methods such as heating, purge gas, depressurization or vacuuming
- Assist in the selection of the proper adsorbent material
- Improve the sizing of absorbers and adsorption wheels

Conversion of existing customer models and licenses

General

Existing customer models will be converted (the name in bracket is the name of the library in the menu):

- ThermalSystems -> TIL (TIL Base Library)
- TSMedia -> TILMedia (present in the suite, but not in the library list in the menu, opened when needed)
- TSMobileAC.Cabin -> TIL3_AddOn_Cabin (present in the suite, but not in the library list in the menu, opened when needed)
- TSMobileAC -> TIL3_AddOn_Automotive (TIL Automotive)

Upgrading an existing model

Existing models using ThermalSystems can be upgraded to TIL Suite by use of automatic conversion scripts. To do this, the following steps must be taken:

1. Open **TIL Suite 2024.1 > TIL Automotive**.
2. Open the model that is using ThermalSystems. The changes from the conversion are logged.
3. Verify the result by running **Check** on your model.

Upgrading your license from ThermalSystems to TIL Suite

The ThermalSystems library will be replaced by the TIL Suite, but your existing license for ThermalSystems is not automatically updated. To make a cost-free license upgrade, please contact your Dymola sales channel. Existing licenses for ThermalSystems will be replaced by TIL Base Library plus TIL Automotive Library.

Note that the ThermalSystems is available in the Dymola 2025x installer, and existing license keys used for earlier versions of Dymola can be used until the license has been upgraded.

Dymola products for the TIL Suite

The seven libraries displayed for the TIL Suite in the **File > Libraries** menu (and two more) are commercially packed into four products.

Below a table of the products and the corresponding libraries. You need this information to know what product you have to buy to get a certain library. All products contain the TIL Base Library and the TIL Media Library. The other libraries are in bold.

Dymola product (trigram in brackets)	Included libraries from the TIL Suite
Dymola – TIL Base Library (FNY)	TIL Base Library TIL Media Library
Dymola – TIL Mobile Air Conditioning Library (HMY)	TIL Base Library TIL Media Library TIL Automotive Library TIL Cabin Library TIL NTU Library
Dymola – TIL Hydrogen Library (HNY)	TIL Base Library TIL Media Library TIL Adsorption Library TIL Hydrogen Energy Systems Library
Dymola – TIL Thermal Storages Library (TTY)	TIL Base Library TIL Media Library TIL Heat Storage Library TIL PCM Library

3.10 Modelica Standard Library and Modelica Language Specification

The current version of the Modelica Standard Library is version 4.0.0. The current version of the Modelica Language Specification is 3.6.

Dymola 2025x is compatible with the future Modelica Standard Library 4.1.0 version, and even includes a vendor-specific ModelicaServices 4.1.0.

3.11 Documentation

General

In the software, distribution of Dymola 2025x Dymola User Manuals of version “September 2024” will be present; these manuals include all relevant features/improvements of Dymola 2025x presented in the Release Notes, except the “under development” ones (if present).

New white paper

A new white paper is available, “Parameter Arrays in Dymola (Managing external parameter sets)”. You can find it by the command **Tools > Help Documents**, in the White Paper section.

3.12 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2025x 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 2025x 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 2025x 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 2025x 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
- For 64-bit GCC: version 7.3 and 8.1

Hence, at least the versions in that range should work fine.

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 option¹ 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 limitations above, it tends to be more numerically robust.

Note that the support for the MinGW GCC compiler will be discontinued in a future release of Dymola.

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.

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 2025x supports DSLS R2025x. Earlier DSLS versions cannot be used.

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

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 2025x 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 2025x it is Qt 6.7.1.
- 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 2025x 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 6.7.1 and inherits the system requirements from Qt. However, since Qt 6.7.1 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 2025x supports DSLS R2025x. 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.

"