

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 2024x Refresh 1	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	10
3.2.1	Easier checking of models	10
3.2.2	Minor improvements	11
3.3	Simulating a model	21
3.3.1	Support for the Ida solver from SUNDIALS	21
3.3.2	Animation tab	23
3.3.3	Scripting	25
3.3.4	Minor improvements	26
3.4	Installation	36
3.4.1	Improved license handling	36
3.4.2	Installation on Windows	40
3.4.3	Installation on Linux	41
3.5	Features under Development	42
3.6	Model Management	44
3.6.1	Encryption in Dymola	44
3.6.2	Extended Git support	44
3.6.3	Improvements in the 3DEXPERIENCE app “Design with Dymola”	46
3.7	Other Simulation Environments	54
3.7.1	Dymola – Scilab interface	54
3.7.2	Dymola – Matlab interface	55
3.7.3	Real-time simulation	55
3.7.4	Java, Python, and JavaScript Interface for Dymola	56
3.7.5	SSP Support	57
3.7.6	FMI Support in Dymola	60
3.7.7	eFMI Support in Dymola	70
3.7.8	Code and model export	73
3.8	Modelica Standard Library and Modelica Language Specification	73
3.9	Documentation	74

3.10	Appendix – Installation: Hardware and Software Requirements	74
3.10.1	Hardware requirements/recommendations	74
3.10.2	Software requirements	75

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 2024x Refresh 1:

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 75 for more information. **Notes:**

- From Dymola 2024 Refresh 1 (this version), Visual Studio C++ compilers older than version 2015 are no longer supported:
 - From Dymola 2024x Refresh 1 (this version), 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 2024x Refresh 1 has limited support for the MinGW GCC compiler, 32-bit and 64-bit. For more information about MinGW GCC, see section “Compilers” on page 75, the section about MinGW GCC compiler.

WSL GCC (Linux cross-compiler)

Dymola 2024x Refresh 1 has limited support for the WSL (Windows Subsystem for Linux) GCC compiler, 64-bit. For more information about WLS GCC, see section “Compilers” on page 75, the section about WSL GCC compiler.

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 78 for more information.

2 About this booklet

This booklet covers Dymola 2024x Refresh 1. 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 2024x Refresh 1

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2024x Refresh 1. In particular, Dymola 2024x Refresh 1 provides:

- Support for the Ida solver from SUNDIALS (previously a beta feature) (page 21)
- Support for Scilab (page 54)
- Support for FMI
 - Support for FMI 3.0:
 - Terminals and icons (*beta feature*) (page 60)
 - Extended co-simulation support:
 - Event mode co-simulation (page 60)
 - Intermediate update (page 61)
 - Early return (page 61)
 - Event handling (page 61)
 - Variable step size communication (page 61) (also FMI2)
 - FMU Export: SUNDIALS solver Ida available for FMU export (page 61)
 - FMU Export: Improved GUI for FMI type and algorithm selection (page 65)
- Support for SSP: A number of implementation improvements (page 57)
- eFMI Support: Support for Software Production Engineering on **3DEXPERIENCE** (page 70)
- “Features Under Development” – to be formally released in a later version (page 42)
- Easier checking of models (page 10)
- Improved handling of Modelica text formatting (page 11)
- Updated and enhanced export of animation files (page 23)
- Improvements in the **3DEXPERIENCE** app “Design with Dymola” (page 46)
- The Java interface and the Python interface updated (page 56 and page 57, respectively)
- Improved license setup for nodelocked license (page 36)
- Teleworking (remote login) supported (page 38)
- Discontinued support for the Microsoft Visual Studio 2012 compiler (page 40)
- Standardized Modelica library encryption (page 44)

3.1.2 New and updated libraries

New libraries

There are no new libraries in this Dymola version.

Name changes

The library Multiflash Media Library has been renamed to Thermodynamics Connector Library in this Dymola version.

Updated libraries

The following libraries have been updated:

- Aviation Systems Library, version 1.6.0
- Battery Library, version 2.7.0
- Brushless DC Drives Library, version 1.4.1
- ClaRa DCS Library, version 1.7.4
- ClaRa Grid Library, version 1.7.4
- ClaRa Plus Library, version 1.7.4
- Claytex Library, version 2024.1
- Claytex Fluid Library, version 2024.1
- Cooling Library, version 1.5.2
- Dassault Systemes Library, version 1.12.0
- Dymola Commands Library, version 1.17
- Dymola Embedded Library, version 1.0.3
- Dymola Models Library, version 1.8.0
- eFMI Library, version 1.0.3
- Electric Power Systems Library, version 1.6.4
- Electrified Powertrains Library (ETPL), version 1.9.0
- Fluid Dynamics Library, version 2.17.0
- Fluid Power Library, version 2024.1
- FTire Interface Library, version 1.3.0
- Human Comfort Library, version 2.17.0
- HVAC (Heating, Ventilation, and Air Conditioning) Library, version 3.2.0
- Hydrogen Library, version 1.4.0
- Multiflash Media Library, see Thermodynamics Connector Library
- Pneumatic Systems Library, version 1.7.0
- Testing Library, version 1.8.0
- Thermal Systems Library, version 1.13.0
- Thermal Systems Mobile AC Library, version 1.13.0

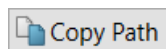
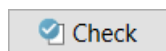
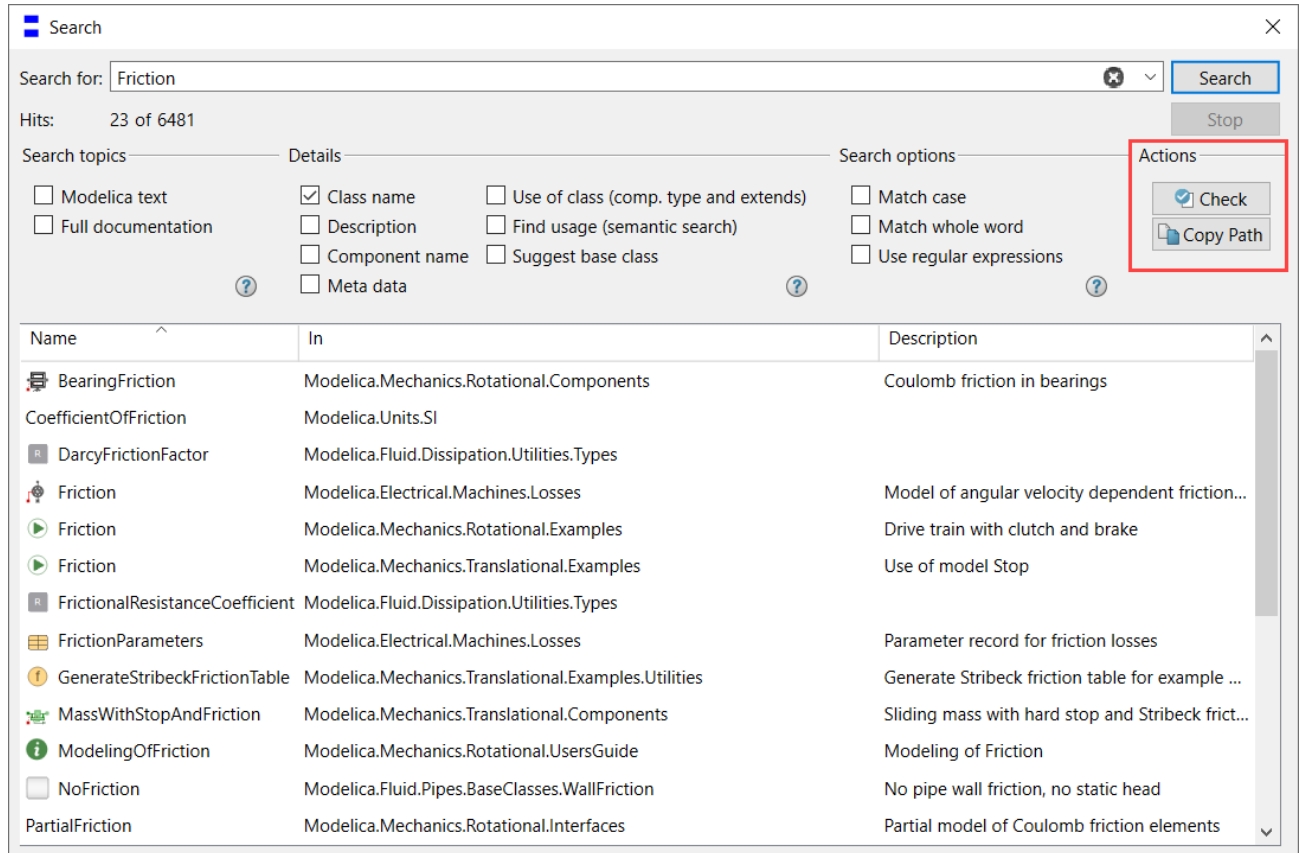
- Thermodynamics Connector Library (previously named Multiflash Media Library), version 1.2.0
- VeSyMA (Vehicle Systems Modeling and Analysis) Library, version 2024.1
- VeSyMA - Engines Library, version 2024.1
- VeSyMA - Powertrain Library, version 2024.1
- VeSyMA - Suspensions Library, version 2024.1
- VeSyMA2ETPL Library, version 2024.1
- Visa2Base, version 1.16
- Visa2Paper, version 1.16
- Visa2Steam, version 1.16
- Wind Power Library, version 1.1.4

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 Easier checking of models

In Dymola 2024x Refresh 1, you can directly check models by a new **Check** button in the **File > Search** dialog:



When you click the **Check** button, all models in the search results displayed are checked for syntactic and semantic errors. This makes it easier for you to check a changed model in all locations where it is used. (Even if a model is “locally” correct, it may fail in a certain context.)

To extend the checking, you can use the button **Copy Path** to copy the paths of the found models to a vector. That vector you can call by a suitable function if needed. An example of the result of such a copy is:

```
{"Modelica.Mechanics.Rotational.Components.BearingFriction", ...,  
"Modelica.Fluid.Pipes.BaseClasses.WallFriction"}
```

3.2.2 Minor improvements

Support for ModelicaServices 4.1.0

For a future Modelica Standard Library version 4.1.0, the library ModelicaServices version 4.1.0 is required. ModelicaServices 4.1.0 is already available in the Dymola distribution.

In ModelicaServices 4.1.0, errors in machine constants have been corrected; this can however effect some older models. In such cases, you get warnings for `Modelica.Constants`. You can instead use the previous values of the constants by setting the flag:

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

(The default value of the flag is `false`.) If you want to try the new Modelica Standard Library (MSL) version 4.1.0, it is available as a beta release at <https://www.modelica.org/>. Note that you need to perform some actions in Dymola to be able to work with MSL 4.1.0, since this is a newer version than the one in the distribution. For information, see the manual “*Dymola User Manual 1A: Introduction, Getting Started, and Installation*”, use the index entry “*Modelica Standard Library: newer than the one in the distribution*” to find the information.

Class name numbering changed

The naming of `Unnamed`-classes and `Submodel` for creating subsystems using the command **Split Model** now use letters, not digits. The goal is to reduce the possible confusion between classes and components, when creating multiple components of these classes.

As an example, creating three `Unnamed`-classes, they will have the names `Unnamed`, `UnnamedA`, and `UnnamedB`. Now, if you instantiate `UnnamedA` three times, the component names will be `UnnamedA`, `UnnamedA1`, and `UnnamedA2`.

Improved keeping of Modelica text formatting

These improvements are not only esthetic but also important for maintaining consistency when a version control system is used.

General formatting improvements

The Modelica text formatting is now better preserved for the following cases:

- Short classes and external calls.
- Public/protected comments.
- Parenthesis for annotations/extends.
- Some keywords needed for the Modelica Standard Library.
- Empty lines in modifier lists.
- Enumerations: comments, and indentation of the enumeration itself.
- Comments in `package.mo` after annotation (two issues: merging of package order and placing the annotation last).

Updating the Modelica text formatting by using a new built-in function

You can update the formatting of Modelica text to follow certain parts of the Modelica standard by using the new built-in function `updateModelicaFormatting`.

```
function updateModelicaFormatting "update Modelica text formatting to follow the
    standard by removing formatting for"
    input String className;
    input Boolean topClass = true "Ensure that entire contents of mo-file is not
        indented.";
    input Boolean publicProtected = true "The keywords public/protected should not
        be indented.";
    input Boolean declarations = true "Declarations should be indented two spaces.";
    input Boolean trailingSpace = true "Ensure that there are no trailing spaces.";
    output Boolean ok;
    external "builtin";
end updateModelicaFormatting;
```

Notes:

- `className` is usually the package name, but you can also specify a certain class inside the package.
- `topClass` can be set to `false` if you want to preserve indented top-level classes. (Some packages in the Modelica Standard Library use indented top-level classes.)

As an example, consider:

```
model Unnamed
    Real x;
    public
        input Real z;
    Real y = 2;
equation

end Unnamed;
```

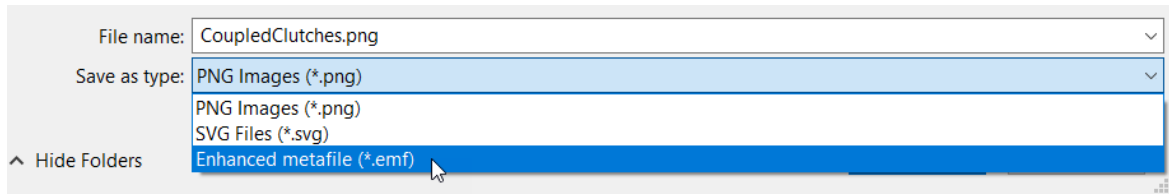
which by `updateModelicaFormatting("Unnamed")` is turned into:

```
model Unnamed
    Real x;
public
    input Real z;
    Real y = 2;
equation
end Unnamed;
```

Improved enhanced metafile support

Saving the active window in enhanced metafile (.emf) format

The enhanced metafile image format (.emf) is now better supported. You can now export the active window as an enhanced metafile (.emf) image, by the command **Tools > Image**. The dialog to save the image now has an enhanced metafile alternative:



Note that you since a long time can export the following active windows as enhanced metafile items to the clipboard by the command **Tools > To Clipboard:**

- Diagram layer
- Icon layer
- Plot windows

Improved rendering

The rendering of metafile images has been improved:

- Filled ellipses are now rendered filled.
- Ellipses are now rendered smooth.

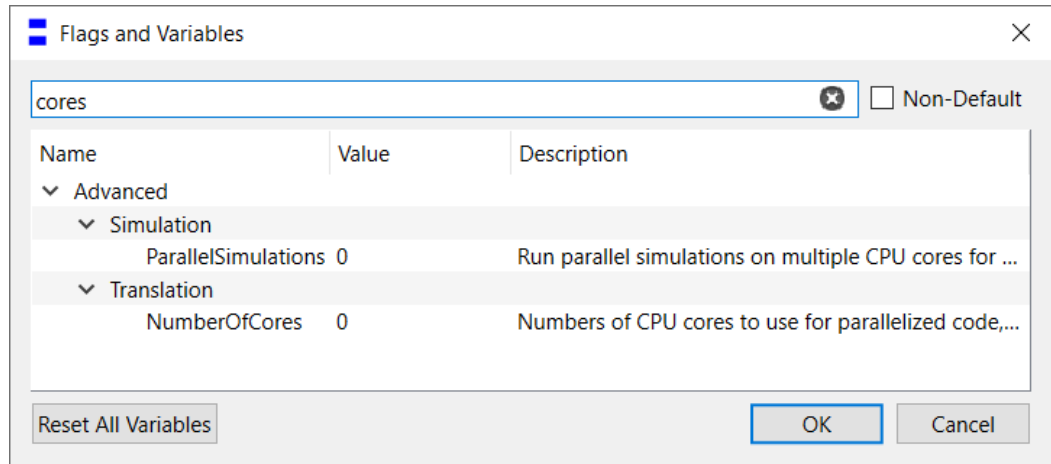
Still, the lowest part of arrows may be cut off, to fix this, if needed, you can:

1. Draw a suitable rectangle.
2. Fill white and no border.
3. Bring to back.

Improved Flags and Variables dialog

The **Flags and Variables** dialog, used to specify flags and variables, reached by the command **Tools > Options > Variables...**, have been improved in three ways:

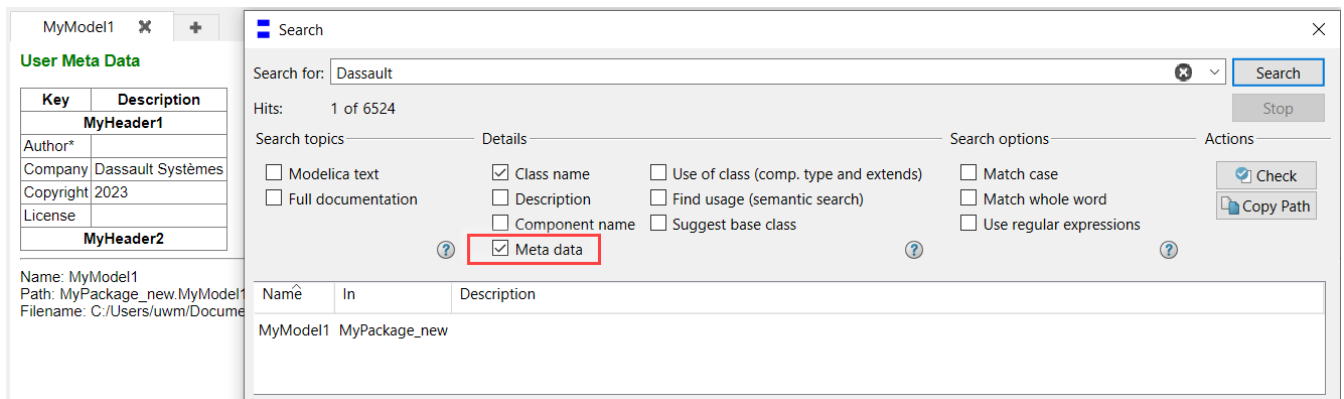
- Filtering now matches *both* the flag/variable name and the description text, making more variables that are relevant searchable. In the example below, the word `cores` matches first a description of a variable, and second a variable name.
- The tooltip for the variable name now displays the full variable name, even if a short name exists for historical reasons.
- The context menu entry **Copy Name** now copies the full variable name, even if a short name exists for historical reasons.



Meta data searchable

Meta data is now searchable by the command **File > Search**, and selecting **Meta data** in the **Details** group. The search matches meta data category descriptions and meta data values, but not meta data keys.

A very simple example:



The **Meta data** option is by default not activated.

Caching function evaluations during translation

By default, function evaluations are now cached during translation, to improve the translation speed. You can disable this feature by setting the flag:

```
Advanced.Translation.CacheFunctionEvaluation = false
```

(The flag is by default true.)

(This feature was a beta feature in the previous Dymola version. The flag has changed name and default value in Dymola 2024x Refresh 1.)

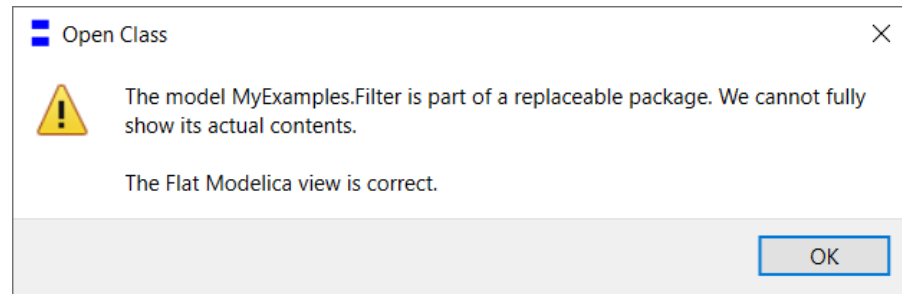
Improved support for package-extends

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")`. However, in Dymola 2024x Refresh 1, the display in the package browser is also correct.

You can select a model inside the extended package in the package browser. By default, you get a warning:



This warning can be disabled by setting the flag:

```
Advanced.UI.WarnIfModelInPackageExtends = false
```

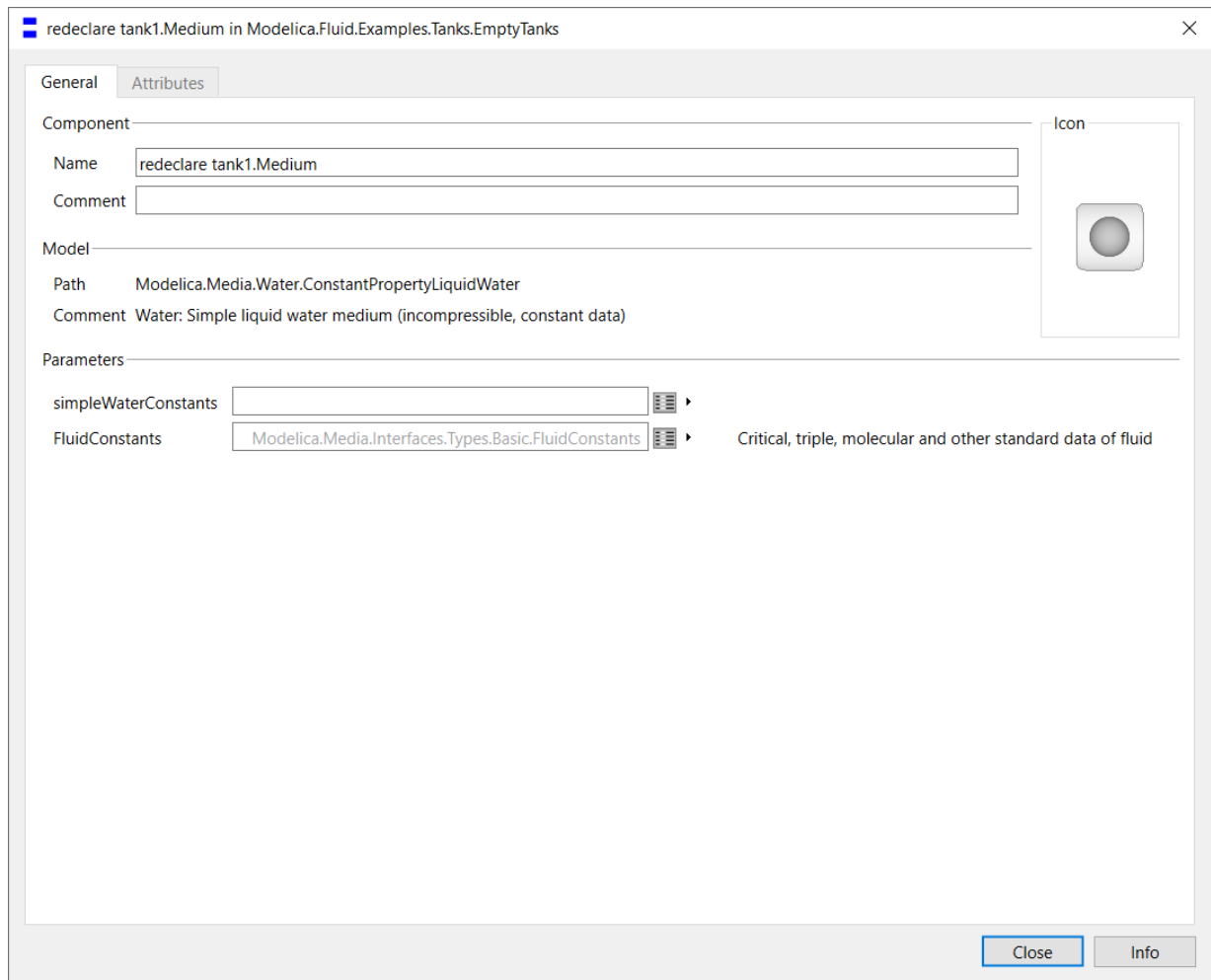
(The default value of the flag is `true`.) The flag is saved between sessions.

Note that the warning might be good to have, to indicate that you work with a model inherited by package extension. Such inherited models are shown read-only. We do not currently support graphical editing in the diagram layer, for example, to modify component parameters. Such editing have to be done in the text layer of the extending package and will *not* be reflected in the diagram layer of the modified model.

Allowing modification of package constants

In Dymola 2024x Refresh 1 you can modify package constants, since they are now displayed in the parameter dialog.

As an example, consider the tank **tank1** in the demo `Modelica.Fluid.Examples.Tanks.EmptyTanks`. Opening the parameter dialog for this tank and then clicking the **Edit** button after the **Medium** input box, you get, in Dymola 2024x:



Doing the same in Dymola 2024x Refresh 1, the package constants are also displayed:


Modelica.Fluid.Examples.Tanks.EmptyTanks

General | Attributes

Component

Name:

Comment:




Icon: 

Model

Path:

Comment:

Parameters

simpleWaterConstants	<input type="text"/>	 ▶
mediumName	<input type="text" value="'SimpleLiquidWater'"/>	▶
substanceNames	<input type="text" value="{mediumName}"/>	 ▶
extraPropertiesNames	<input ",="" 0)"="" type="text" value="fill("/>	 ▶
reference_p	<input type="text" value="1.01325"/>	▶ bar
reference_T	<input type="text" value="25"/>	▶ °C
reference_X	<input type="text" value="fill(1/nX, nX)"/>	▶ kg/kg
p_default	<input type="text" value="1.01325"/>	▶ bar
T_default	<input type="text" value="(Modelica.Units.Conversions.from_degC(20)) - 273.15"/>	▶ °C
h_default	<input type="text" value="specificEnthalpy_pTX(p_default, T_default, X_default)"/>	▶ J/kg
X_default	<input type="text" value="reference_X"/>	▶ kg/kg
C_default	<input type="text" value="fill(0, nC)"/>	▶
nX	<input type="text" value="nS"/>	▶
nXi	<input type="text" value="if fixedX then 0 else if reducedX then nS - 1 else nS"/>	▶

Name of the medium

Names of the mixture substances. Set substanceNames=(mediumName) if only one substance.

Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused

Reference pressure of Medium: default 1 atmosphere

Reference temperature of Medium: default 25 deg Celsius

Default mass fractions of medium

Default value for pressure of medium (for initialization)

Default value for temperature of medium (for initialization)

Default value for specific enthalpy of medium (for initialization)

Default value for mass fractions of medium (for initialization)

Default value for trace substances of medium (for initialization)

Number of mass fractions

Number of structurally independent mass fractions (see docu for details)

Close Info

Notes:

- Be careful, these constants being shown does not mean you are supposed to change them if not having good arguments to do so.
- Be careful, use common Media packages to avoid duplicating the information.

Warnings for conditional components disabled

In Dymola 2024x Refresh 1, the warnings for conditional components have been disabled also in pedantic mode. To activate them, if needed, you can set the flag:


```
Advanced.Modelica.CheckConditionallyEnabled = true
```

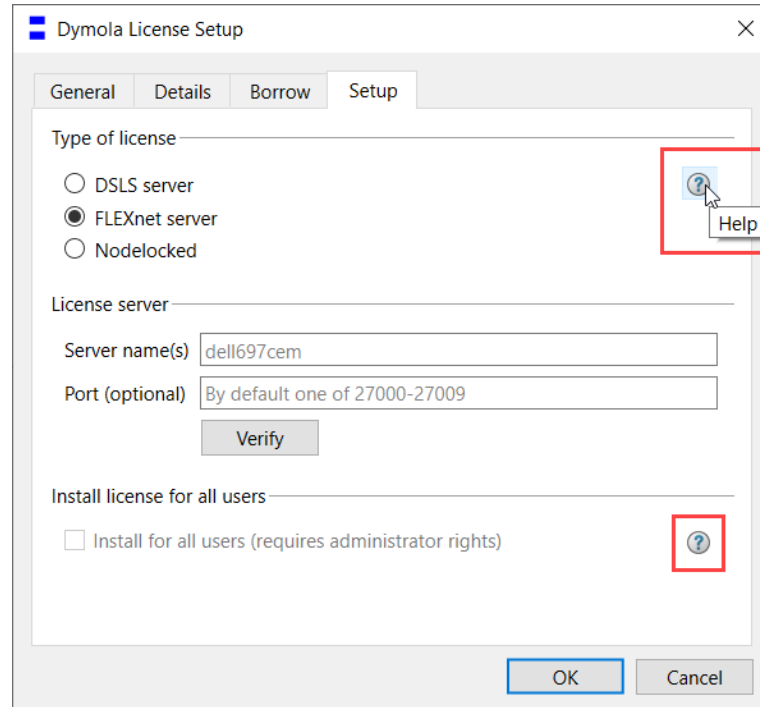
Note that if you set the flag to `true`, the warnings are always given, also when not in pedantic mode.


(The flag is by default `false`.)

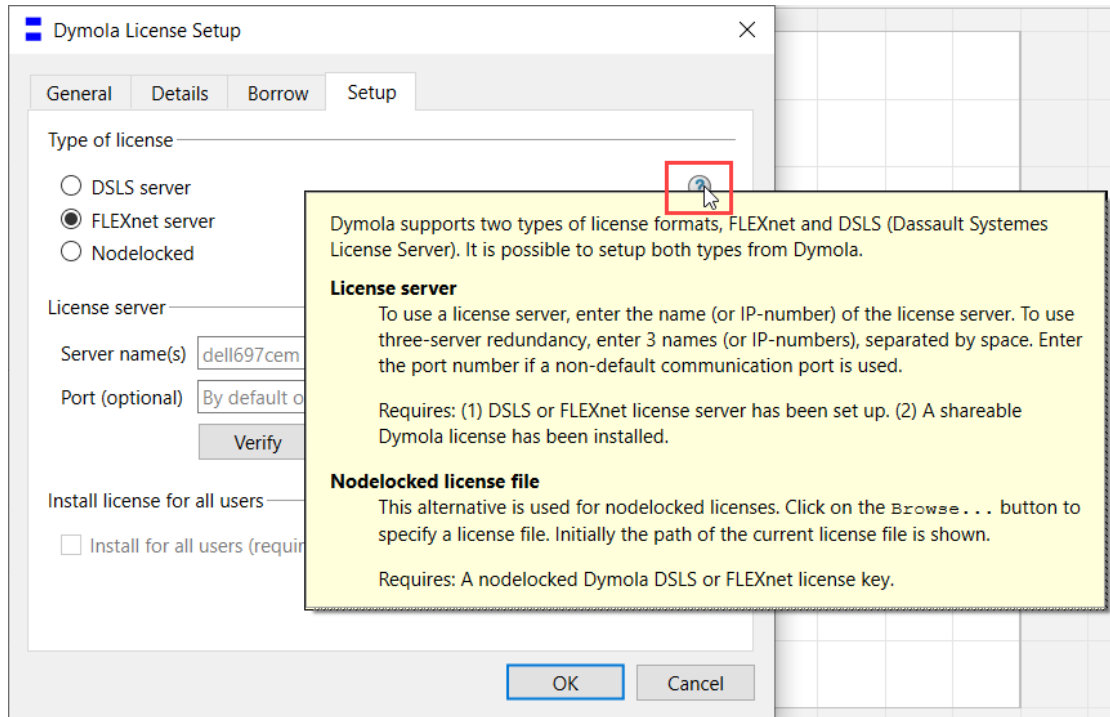
More accessible help texts


Since long, help texts have been available by right-clicking in the area of a group in a dialog and selecting **What's This?**

Now, for some dialogs, the texts are also available by new **Help** buttons . An example for the **Setup** tab:



Clicking the first  gives:



Advanced users that don't need these **Help** buttons  or prefer using the context menu to see the texts can hide the new buttons by setting the flag:

```
Advanced.UI.DialogHelpButton = false
```

(The flag is by default `true`.)

Note that sometimes a button covers more than one group; in the example above the button covers both **Type of license** and **License server**.

MathJax handling improved

General MathJax support

MathJax is used for rendering of equations in MathML and LaTeX/TeX in the Dymola documentation.

In previous Dymola versions, the support of MathJax was handled by linking to a content provider over Internet. In Dymola 2024x Refresh 1, MathJax is loaded when Dymola starts, and a cached copy is used for documentation.

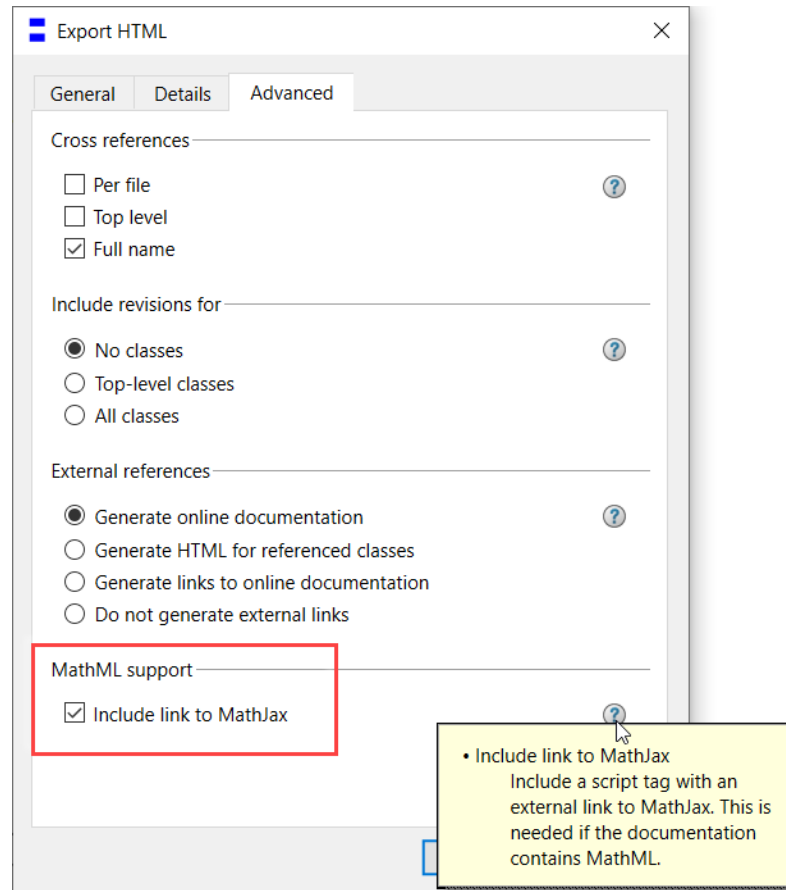
If you start Dymola with the command line command `-nocachedmathjax`, MathJax is not cached, and the following official link to MathJax is used instead:

<https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-svg.js>

Exporting documentation to file

You can export documentation by GUI (the command **Tools > [Export] HTML**) and any of the two built-in functions `exportDocumentation` and `exportHTMLDirectory`.

If you export documentation containing MathML/LaTeX, you can select to include a script tag with MathJax, by activating **Include link to MathJax** in the **Advanced** tab of the command **Tools > [Export] HTML**:



Activating this setting corresponds to setting the flag

```
Advanced.HTML.IncludeLinkMathJax = true
```

(The flag is by default `false`; the setting is by default not activated)

If the setting is activated, the above link is used.

3.3 Simulating a model

3.3.1 Support for the Ida solver from SUNDIALS

(This feature was a beta feature in the previous Dymola release. Now being unconditionally implemented, the corresponding Beta flag has been removed.)

The SUNDIALS Ida solver is now available as a part of the SUNDIALS 6.4.1. (For SUNDIALS 6.4.1, see also “Upgrade to SUNDIALS 6.4.1 and SuperLU_MT 3.1” on page 36.)

Ida is a modern variable-stepsize, variable-order solver of hybrid DAEs. In the core, it implements a BDF integrator and is therefore most similar to Dassl and Cvode.

Just as Dassl, Ida allows integration of DAEs. This is a benefit over SUNDIALS Cvode, which is a pure hybrid ODE solver.

Among other features, the Dymola Ida implementation supports:

- Sparse linear solvers for large-scale systems, by setting `Advanced.SparseActivate = true`.
- Dymola DAE mode for efficient integration of dynamic models with large algebraic loops, by setting `Advanced.Define.DAEsolver = true`. Just as the other Dymola DAE solvers, Ida DAE mode uses a variant of the efficient and accurate event handling as described in Henningsson, Olsson, and Vanfretti: *DAE Solvers for Large-Scale Hybrid Models* (<https://dx.doi.org/10.3384/ecp19157491>).
- Efficient minor events for fast handling of events that don’t affect the dynamics.

In Dymola 2024x, the solver can be selected in the simulation setup (reached by the command **Simulation > Setup**), the **General** tab:

Simulation Setup

General Translation Output Debug Compiler Realtime FMI Export FMI Import

Experiment

Model Unnamed

Result Unnamed

Max simulation run time

For batch Not set s

Per simulation Not set s

Simulation interval

Start time 0 s

Stop time 1 s

Steady State

Output interval

Interval length 0 s

Number of intervals 500

Integration

Algorithm Dassl

Tolerance

Fixed Integrator Step

Esdirk34a - order 4 stiff

Esdirk45a - order 5 stiff

Dopri45 - order 5

Dopri853 - order 8

Sdirk34hw - order 4 stiff

Cerk23 - order 3

Cerk34 - order 4

Cerk45 - order 5

Cvode - variable order

Ida - variable order

Store in Model ☒ Automatically store General and Inline integration settings

OK Cancel

The Ida solver can be used in the following cases:

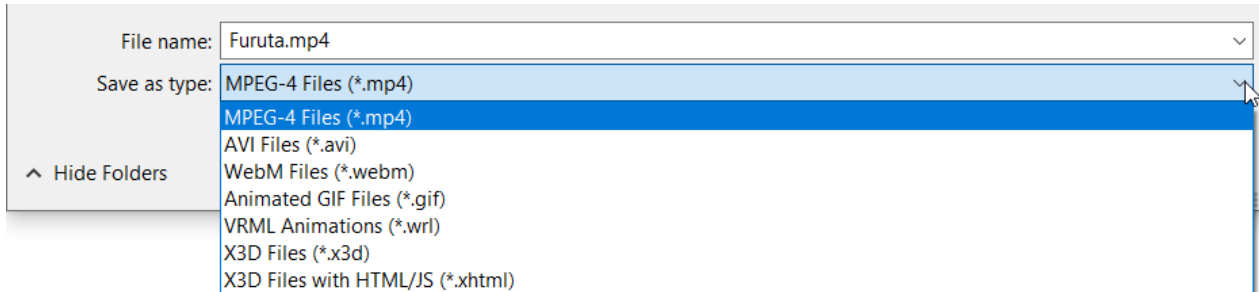
- Simulation directly in Dymola
- FMU export with or without including source code. For the FMU export, see section “SUNDIALS solver Ida available for FMU export” on page 61.

3.3.2 Animation tab

Improved and enhanced export of animation files

The animation export has been improved, to better support previous file formats, and adding new file formats.

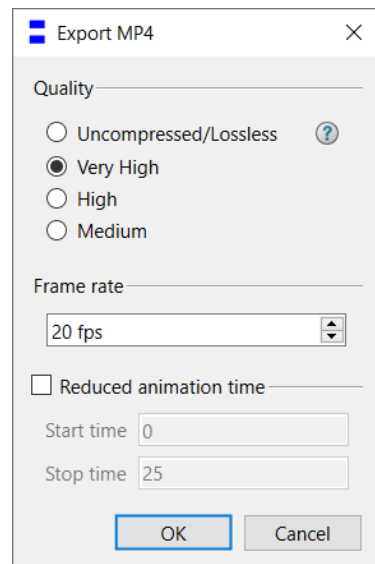
The following figure shows the selectable file formats when exporting an animation from Dymola on Windows. (To export an animation, you can use the command **Export Animation** from the **Animation** tab, or the command **Tools > [Export] Animation**.)



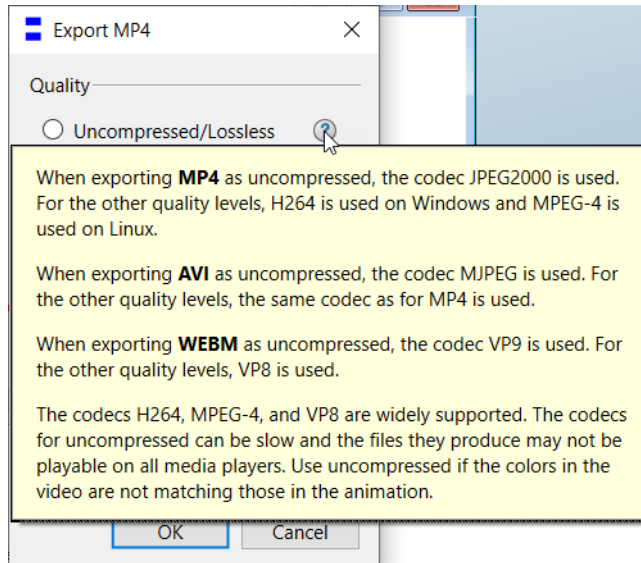
The new supported file formats are:

- MPEG-4 (.mp4) – this is the default selection
- WebM (.webm)

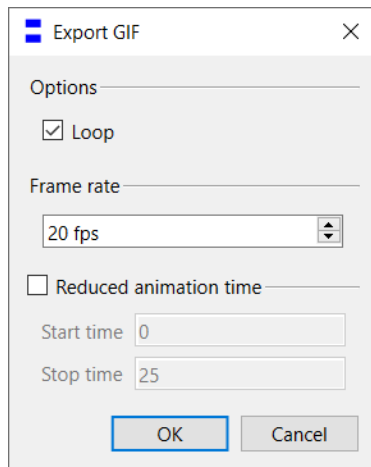
When selecting any of three first formats in the selection menu above, you get a dialog like:



The “?” button displays information about the codecs used etc.:



For Animated GIF Files the dialog looks like:



For the last three export alternatives in the selection menu above, the export is done without displaying any additional dialog.

Note that the corresponding built-in function `exportAnimation` has also been improved accordingly. Please see section “The built-in function `exportAnimation` improved” on page 25.

3.3.3 Scripting

New built-in function `DymolaLicenseInfo()`

A new built-function for returning license information has been added. For more information, see section “New built-in function `DymolaLicenseInfo()` that returns license information” on page 39.

The built-in function `SetDymolaCompiler` improved

The built-in function `SetDymolaCompiler` has been improved in three ways:

The function now output diagnostics if you try to specify an unknown compiler, or if you miss to specify major compiler settings.

A new Boolean input argument `mergeSettings` is added. The default value is `true`, meaning that previous settings are remembered when changing any input arguments, unless they are overwritten by the new call. This makes it easy to, for example, changing a present Visual Studio compiler setup to a Windows Subsystem for Linux one by just calling `SetDymolaCompiler("wsl")` or to change a Visual Studio version by calling `SetDymolaCompiler("vs", {"MSVCDir=..."})`.

The function is now also supported on Linux. However, only the first argument is supported. That is, you can use any of the function calls: `SetDymolaCompiler("gcc")` or `SetDymolaCompiler("clang")`.

The built-in function `GetDymolaCompiler` improved

This built-in function is now also supported on Linux.

The built-in function `exportAnimation` improved

The built-in function `exportAnimation` has been improved, to cover the improvements and enhancements in exporting animation files. Those are described in section “Improved and enhanced export of animation files” starting on page 23.

The input argument `path` has been extended with the file formats MP4 and WEBM.

The following input arguments have been added:

- `framerate` – The integer value of the frame rate in fps, default value 20.
- `quality` – An integer value for the quality of the animation exported. This argument is only applicable to files of MP4, AVI, and WEBM formats. The possible values are:
 - 0 – Uncompressed/lossless
 - 1 – Very high (This is the default value.)
 - 2 – High
 - 3 – Medium
- `repeat` – a Boolean value to specify if an animated Gif file should loop by default. The default value is `true`.

The built-in function `translateModelFMU` improved

The built-in function have been improved to take into account the new available Ida solver. The new solver effects the input argument `fmiType` if `cs` or `all` is selected for this argument. What is selected is, in priority order:

- If an inline method is selected by using the flag `Advanced.Translation.InlineMethod`, that inline method is selected. (The default of the flag is that no inline method is selected.)
- If the Ida algorithm is selected by setting the flag `Advanced.FMI.Ida = true`, Ida is selected. (The default of the flag is `false`.)
- If the flag `Advanced.FMI.Ida` has the default value `false`, Ccode is selected as algorithm.

Note that the selection of `csSolver` for `fmiType` now is implemented in another way in the GUI; there is now no specific corresponding FMI type selection. Instead you have to select **Current Dymola solver** as **Algorithm** – this will give you the wanted selection. For more information about the GUI, see section “Improved GUI for exporting an FMU” on page 65.

Updating Modelica text formatting

You can update the formatting of the Modelica text to follow certain parts of the Modelica standard by the new built-in function `updateModelicaFormatting`. For more about this, see section “Updating the Modelica text formatting by using a new built-in function” on page 12.

3.3.4 Minor improvements

Grouping for nonlinear systems of equations with numeric Jacobian

When a model contains algebraic loops, Dymola may generate systems of equations to solve such loops. The solving procedure involves calculations of the system Jacobian. Normally, Dymola will find these Jacobians analytically by differentiating the residual equations. However, sometimes such differentiation fails, and then Dymola falls back on using numerical approximations.

It is possible to speed up the numeric Jacobian approximation using grouping to reduce the number of residual evaluations. This feature is by default enabled, you can disable it by setting the flag:

```
Advanced.Translation.NonlinearSystemGrouping = false
```

(The default value of the flag is `true`.) For large and sparse systems, also consider enabling sparse solvers, to also speed up the Jacobian factorization.

Other than the potential efficiency gain, enabling grouping should normally have a very small effect on the overall solution procedure for the algebraic loop. In general, the Jacobian approximation should be more exact as all structural zeros are guaranteed to be zero also in the approximation.

Note. For the approximation of the ODE Jacobian, grouping is always used when applicable, independent of the value of the above flag.

(This feature was a beta feature in the previous Dymola version. The flag has changed name and default value in Dymola 2024x Refresh 1.)

Generating analytical Jacobians when dependencies are traced through Co-simulation FMUs with loops

You can generate analytical Jacobians also when dependencies are traced through Co-simulation FMUs with loops. You do that by setting the flag

```
Advanced.Translation.ODEJacobianForDiscrete = true
```

(The flag is by default `false`.)

(This feature was a beta feature in the previous Dymola version. The flag has changed name in Dymola 2024x Refresh 1.)

Option to apply additional array aliasing

You can select to allow additional aliasing for entire array variables (after expansion) in cases where normal aliasing cannot be used. You do this by setting the flag

```
Advanced.Translation.ArrayAlias = true
```

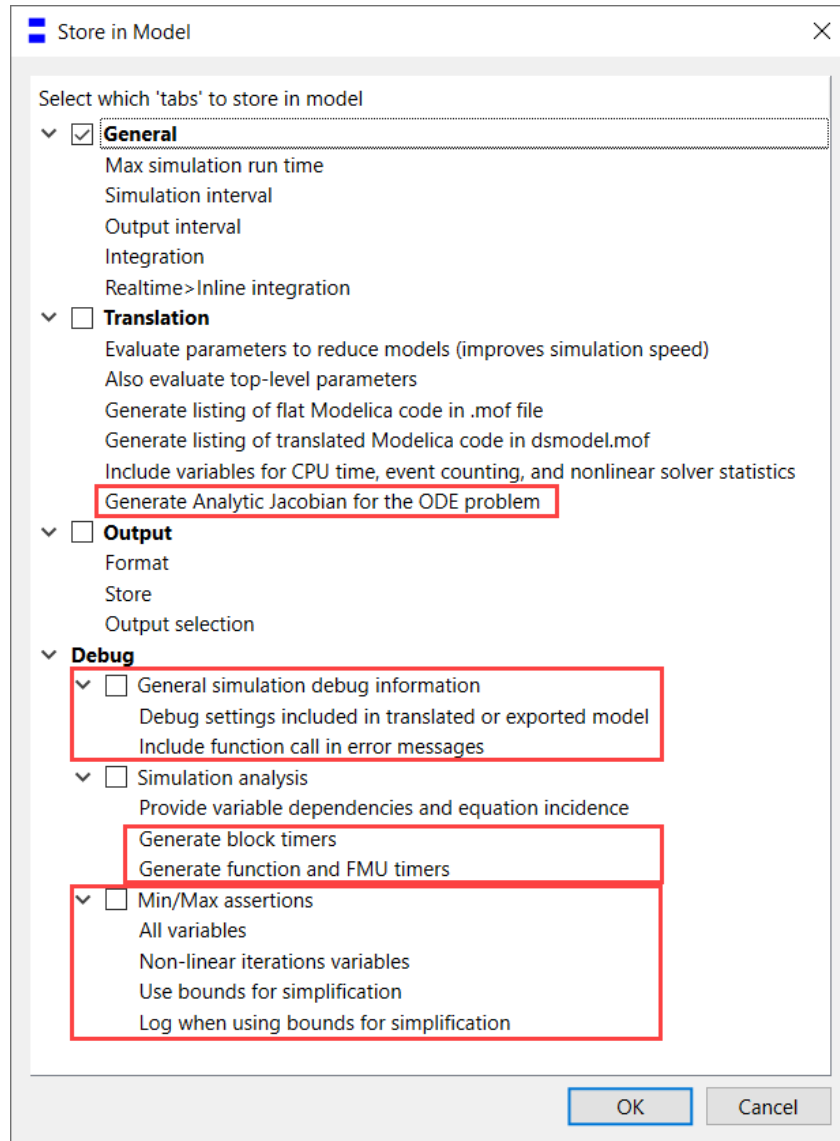
(The default value of the flag is `false`.)

Applying this option might reduce the result file about 10 %, depending on the model. **Note.** In rare cases, this aliasing can affect the selection of start-values.

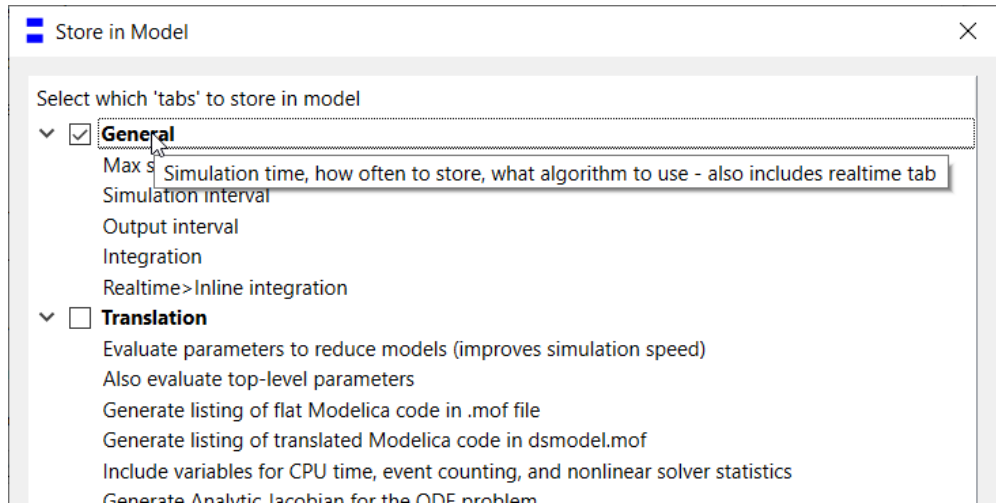
Improved storing of simulation flags in model

In the simulation setup, reached by the command **Simulation > Setup**, you can click the button **Store in Model** to decide what flags of the simulation setup should be stored in the model.

This dialog has now been enhanced with a number of flags, framed in the image below.



Note that you have tooltips for the sections, an example:



Function indicating if solver is fixed-step or variable

Some models are designed to run only with a fixed-step solver. An example is when a finite impulse response (FIR) filter is applied to the last consecutive values.

The function `Dymola.Simulation.isVariableStep()` outputs `true` if the solver is detected to be a variable-step solver, `false` otherwise.

An example of use:

```
model M
  Real x;
equation
  der(x)=1-x;
  assert(not Dymola.Simulation.IsVariableStep(), "Must use
    fixed step");
end M;
```

Logging of external function calls during translation improved

Already in Dymola 2024x, you could log external function calls during translation by setting the flag:

```
Advanced.Translation.Log.FunctionCallEvaluation = true
```

(The flag is by default `false`.)

An example of function calls during translation is setting of scalar parameter values from CSV files. This can mean many calls to read the matrix size, and then to read the matrix, in order to read just one value. The logging enables you to find out if this is an issue, that is, if you have numerous such calls.

The logging has been improved in Dymola 2024x Refresh 1 to cache and display more information and cover more cases.

Allowing non-evaluated parameter expressions for the attributes min, max, nominal, and unbounded

Non-evaluated parameter expressions for the attributes min, max, nominal, and unbounded are now allowed by default. You can disable this feature by setting the flag:

```
Advanced.Translation.SupportNonLiteralAttributes = false
```

(The flag is by default true.)

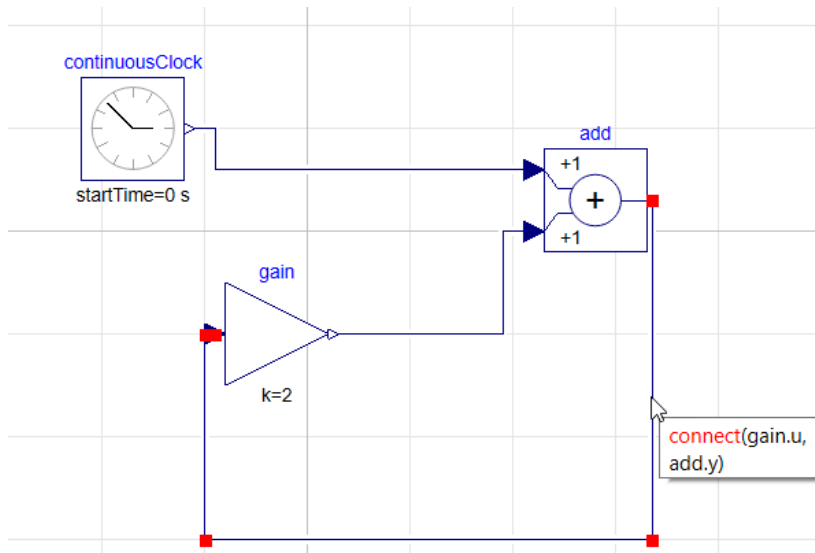
(This feature was a beta feature in the previous Dymola version. The flag has changed name and default value in Dymola 2024x Refresh 1.)

Option to get diagnostics when reversing causality

To use this option, activate the flag `Advanced.Translation.CheckLoopCausality`. (The flag is by default false.) We recommend also setting `Evaluate = false` when using this option, otherwise some case might be missed.

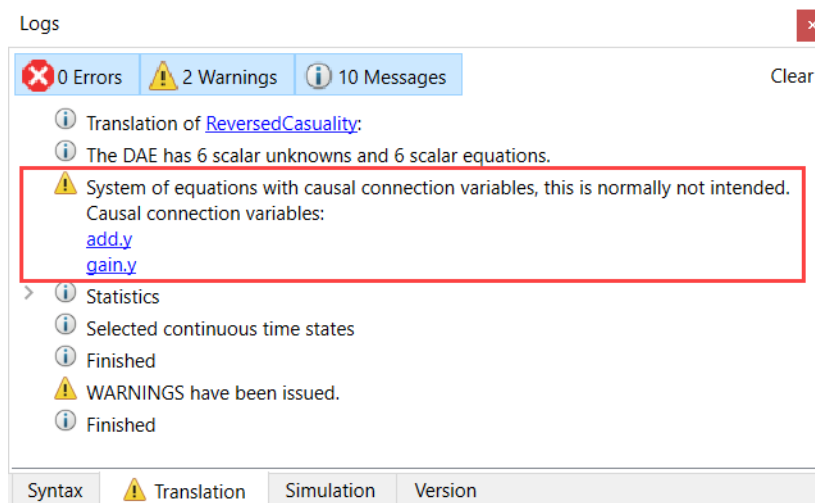
An example model might be:

```
model ReversedCasualty
  Modelica.Blocks.Math.Gain gain(k=2)
    annotation (Placement(transformation(extent={{-16,-30},{4,-10}})));
  Modelica.Blocks.Math.Add add
    annotation (Placement(transformation(extent={{46,-4},{66,16}})));
  Modelica.Blocks.Sources.ContinuousClock continuousClock
    annotation (Placement(transformation(extent={{-44,10},{-24,30}})));
equation
  connect(gain.y, add.u2) annotation (Line(points={{5,-20},{38,-20},{38,0},{
    44,0}}, color={0,0,127}));
  connect(add.u1, continuousClock.y) annotation (Line(points={{44,12},{-18,12},
    {-18,20},{-23,20}}, color={0,0,127}));
  connect(gain.u, add.y) annotation (Line(points={{-18,-20},{-20,-20},{-20,
    -60},{67,-60},{67,6}}, color={0,0,127}));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false)), Diagram(
    coordinateSystem(preserveAspectRatio=false)),
    experiment(StopTime=2, __Dymola_Algorithm="Dassl"));
end ReversedCasualty;
```



Here, the connection between the add block and the gain block constitutes a reversed causality issue.

Activating the flag, setting `Advanced.Translation.CheckLoopCausality=true`, and setting `Evaluate=false`, and then simulating, you will get, in the translation log:



(This feature was a beta feature in the previous Dymola version. The flag has changed name in Dymola 2024x Refresh 1.)

Differentiating discrete expressions

As an example, consider the model:

```
model Test
  Real x;if time>=0.5 and time<0.9 then 1 else 0.1;
  Real y;
equation
  der(x)=der(y);
end Test;
```

If you try to simulate this model, it will fail, and you will get the message:

```
Model error - differentiated if-then-else was not continuous:
(if time >= 0.5 and time < 0.9 then 1 else 0.1)
Value jumped from 0.1 to 1.
There is support for using impulses to handle this; please
refer to the manual.
```

If you set the flag `Advanced.Translation.ImpulseWhenDifferentiatingDiscontinuity = true` and then simulate, you will succeed, with the messages:

```
Introducing impulse to handle differentiated expression that
was not continuous:
(if time >= 0.5 and time < 0.9 then 1 else 0.1)
Value jumped from 0.1 to 1.
```

```
Introducing impulse to handle differentiated expression that
was not continuous:
(if time >= 0.5 and time < 0.9 then 1 else 0.1)
Value jumped from 1 to 0.1.
```

The problem is solved by generating an impulse so that y also changes discontinuously.

The reasons that this feature is not activated by default (the flag is by default `false`) are:

- Impulses currently only work when simulating normally, and not in Matlab/Simulink, FMU etc.
- The "integration" of the discontinuity is currently not fully implemented.
- The feature only handles first derivatives.
- If there are multiple relevant things changing at an event it might not be currently possible to handle.

(This feature was a beta feature in the previous Dymola version. The flag has changed name in Dymola 2024x Refresh 1.)

Inheriting a partial experiment-annotation

By default, the **General** tab of the simulation setup (reached by the command **Simulation > Setup**) is stored in the model.

The framed settings in the figure below are since long saved as an annotation when saving the model, `annotation(experiment(...))`.

Simulation Setup

×

General

Translation

Output

Debug

Compiler

Realtime

FMI Export

FMI Import

Experiment

Model

Unnamed

Result

Unnamed

Max simulation run time

For batch

Not set

s

Per simulation

Not set

s

Simulation interval

Start time

0

s

Stop time

1

s

Steady State

Output interval

Interval length

0

s

Number of intervals

500

Integration

Algorithm

Dassl

Tolerance

0.0001

Fixed Integrator Step

0

s

Store in Model

☒ Automatically store General and Inline integration settings

OK

Cancel

In Dymola 2024x Refresh 1, if you extend a model having some of the framed settings changed, you inherit these changes (as an experiment annotation).

If you change any of these settings in the model extending from the first model, you will have the resulting settings when storing the model as the settings from the first model + the settings

from the model extended from the first one. The *specified* settings in the model extended from the first one have precedence; they override the settings in the first model.

As an example, consider:

```
package P
  model M

    annotation (experiment(StopTime=2, __Dymola_Algorithm="Dassl"));
  end M;

  model M2
    extends M;
    annotation (experiment(__Dymola_Algorithm="Lsodar"));
  end M2;
end P;
```

The result of simulating the model M2 and looking at the simulation setup is:

Simulation Setup

×

General

Translation

Output

Debug

Compiler

Realtime

FMI Export

FMI Import

Experiment

Model

P.M2

Result

M2

Max simulation run time

For batch

Not set

s

Per simulation

Not set

s

Simulation interval

Start time

0

s

Stop time

2

s

Steady State

Output interval

Interval length

0

s

Number of intervals

500

Integration

Algorithm

Lsodar

Tolerance

0.0001

Fixed Integrator Step

0

s

Store in Model

☒ Automatically store General and Inline integration settings

OK

Cancel

Improved handling of minor time events

The solvers Dassl, Lsodar, Cvode, and Ida now handle minor time events more efficiently, allowing them to take larger integration steps. The solver behavior is also less affected by the choice of using equidistant grid or not.

Upgrade to SUNDIALS 6.4.1 and SuperLU_MT 3.1

(This feature was a beta feature in the previous Dymola release. Now being unconditionally implemented (the old versions have been removed), the corresponding Beta flag has been removed.)

The SUNDIALS version 6.4.1 and SuperLU_MT version 3.1 are now used in Dymola.

For SUNDIALS this applies to Ccode and Ida used in Dymola, for FMU export, and for FMU source code export. For SuperLU_MT this applies to all implicit solvers in Dymola, to FMU export, to FMU source code export, and to sparse solution of systems of equations.

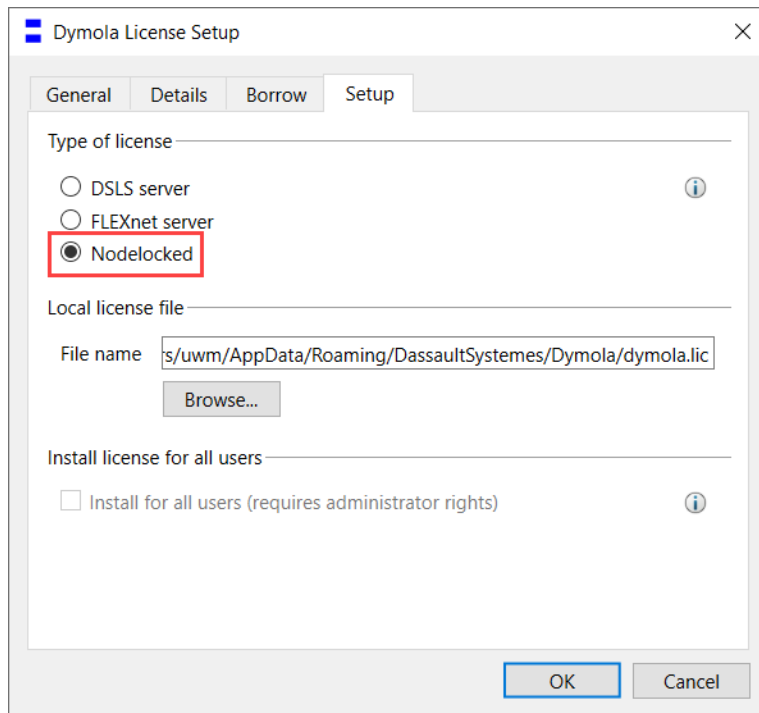
3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 74.

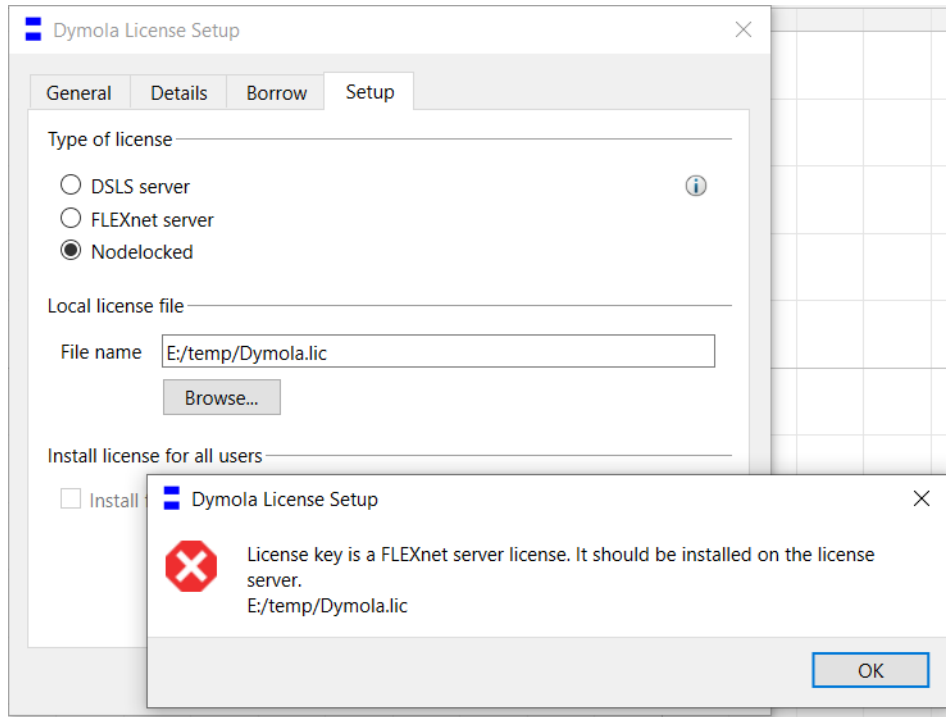
3.4.1 Improved license handling

Simplified setup of nodelocked license

In Dymola 2024x Refresh 1, Dymola will automatically detect what type of nodelocked license file you try to install. This means that the license setup (reached by the command **Tools > License Setup**, the **Setup** tab) has been simplified to have only one selection for a nodelocked license:



Dymola also detects if you by mistake try to install a license file intended for the license server as your local nodelocked license. The message given suggests a suitable action. An example:



Teleworking (remote login) supported in Dymola

Teleworking (to work from home by making a remote login to work) is now allowed in the Dymola license conditions. In earlier versions of Dymola, this has required a shareable (server) license. From Dymola 2024x, new FLEXnet license keys support teleworking also for nodelocked licenses. The following restriction apply:

- Only one user can make a remote login and use Dymola with the nodelocked license.
- To enable teleworking with nodelocked licenses, a new FLEXnet license key must be ordered and installed.
- With the new license key, teleworking is supported from Dymola 2022x forward. However, error messages when two users try to login are improved in Dymola 2024x Refresh 1.
- Teleworking is not supported for nodelocked DSLS license keys.

See also the document [Dymola 2024x Updates for FLEXnet Licenses](#).

More information in About Dymola, in particular TOS (Termination of Support)

The command **Tools > About Dymola** now displays more information when it comes to important dates. The dates now presented are:

- License expires
- Support period expires [*new, also referred to as TOS (Termination of Support)*]

- Dymola WWGA date [*new, the release date of this version (World Wide General Availability)*]

Notes:

- Support entitles you to run new versions of Dymola. When you have reached TOS, you are no longer entitled to versions that are more recent. To use new versions, you must update your support period.
- TOS becomes effective when you install a newly generated license key. Because most of license keys are valid for up to two years, the effect of the TOS may become delayed.
- All recent versions of Dymola will be subject to license checking retroactively (with a new license key).
- Even if you have reached TOS, you can still run older versions of Dymola, those where the WWGA date is before the TOS date. This is different from a temporary license, which does not allow you to run any version of Dymola at all after expiration, except Dymola in trial mode.
- TOS dates can also be applied to libraries (with server licenses).
- If any licensing problem, Dymola will start in trial mode with limited capability.
- Any issues with TOS should be addressed to your local Dymola reseller, it is most likely not a technical problem.

New built-in function `DymolaLicenseInfo()` that returns license information

The new built-in function `DymolaLicenseInfo()` provides a lot of information that is valuable debugging license issues. Any support request related to licenses should be accompanied by the output from `DymolaLicenseInfo()`. An example from a license without any issue:

```
X DymolaLicenseInfo()
  = "*" Version:
  Dymola Version 2024x Refresh 1 Beta 3, 2024-03-27
  * User:
  Lund_DS AB
  * Site:
  DS Internal
  Dymola license server
  * Checked out license features:
  Standard
  * License number:
  14388
  * Expiration date:
  2025-02-09
  * Support end date (TOS):

  * Dymola release date (WWGA):
  2024-04-19
  * License server:
  dell697cem
  * License status:
  License file is correct
  "
DymolaLicenseInfo()
```

Commands

Commands Logs

Do not forget the brackets when entering `DymolaLicenseInfo()`. Without the brackets, the call is not valid.

3.4.2 Installation on Windows

Discontinued support of Microsoft Visual Studio 2012 compiler

From this Dymola version, Dymola 2024x Refresh 1, the compiler Microsoft Visual Studio 2012 is no more supported. This is the case for both the commercial Visual Studio 2012 and the Visual Studio 2012 Express edition.

The reason for Visual Studio 2012 not being supported anymore is the logistics of supporting multiple old versions for all solvers.

The oldest Visual Studio compiler now supported is Visual Studio 2015.

For a full list of supported compilers, see section “Compilers” starting on page 75.

If you try to translate/simulate a model that uses libraries that are built with older Visual Studio versions than 2015, you can get link errors. The message that appears indicates that you can set the flag

```
Advanced.Translation.LinkWithIobFuncStub = true
```

as a temporary workaround. However, you are strongly recommended to rebuild these libraries using a supported compiler.

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

Support of the MinGW GCC compiler will be discontinued in a future release

The support of the MinGW GCC compiler will be discontinued in a future release of Dymola. We recommend that you already now switch to the Windows Subsystem for Linux (WSL) GCC compiler. For more information on that compiler, see section “Compilers” starting on page 75.

Updated Qt version

Dymola 2024x Refresh 1 is built with Qt 6.6.0.

3.4.3 Installation on Linux

Discontinued support for 32-bit simulation in Dymola 2025x

This version of Dymola, Dymola 2024x Refresh 1, is the last version that supports 32-bit simulation on Linux. The future discontinued support will mean:

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

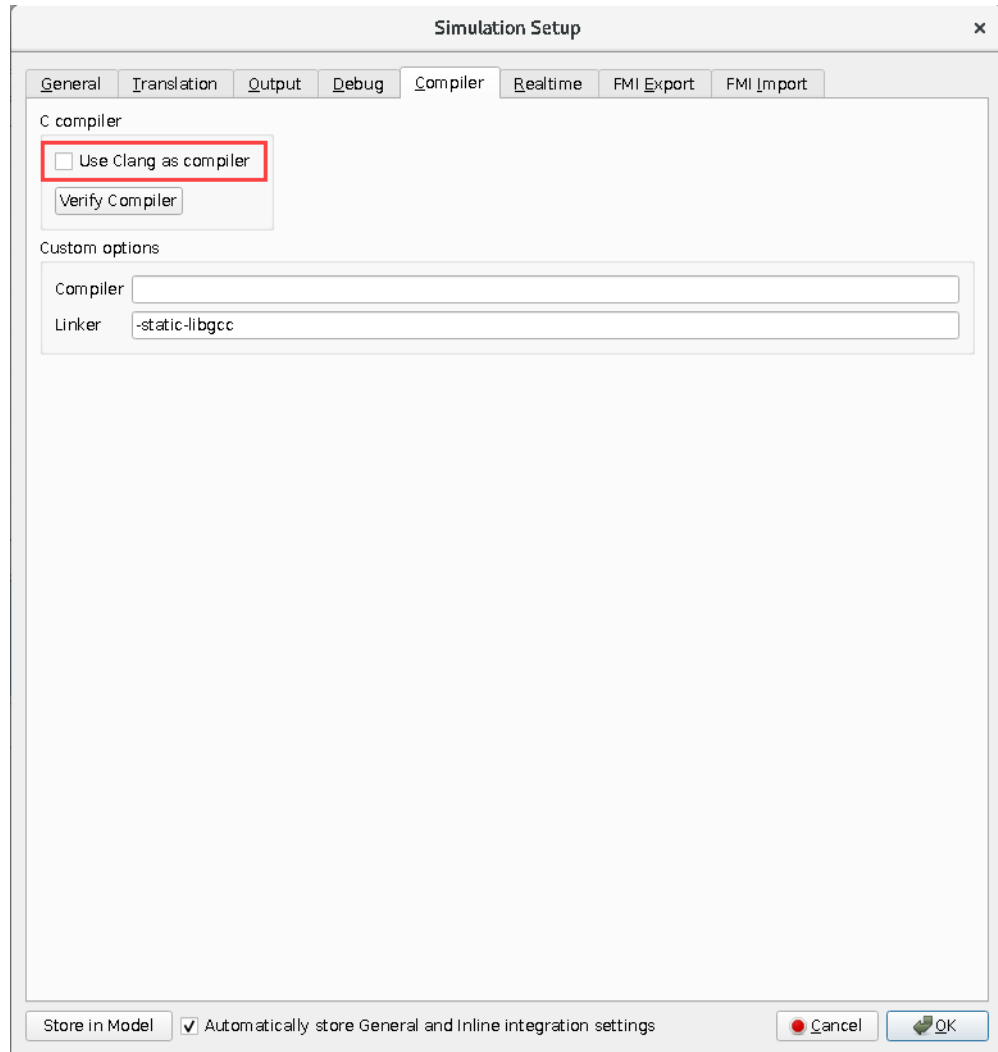
(Note that the support for 32-bit simulation on Windows is continued in Dymola 2025x.)

Updated Qt version

Dymola 2024x Refresh 1 is built with Qt 6.6.0.

GUI for selecting Clang as compiler

You can now set Clang as compiler in the simulation setup, the **Compiler** tab, by activating **Use Clang as compiler**:



By default, the Clang compiler is not selected (as in image above).

(The simulation setup for this feature is reached by the command **Simulate > Setup**, the **Compiler** tab.)

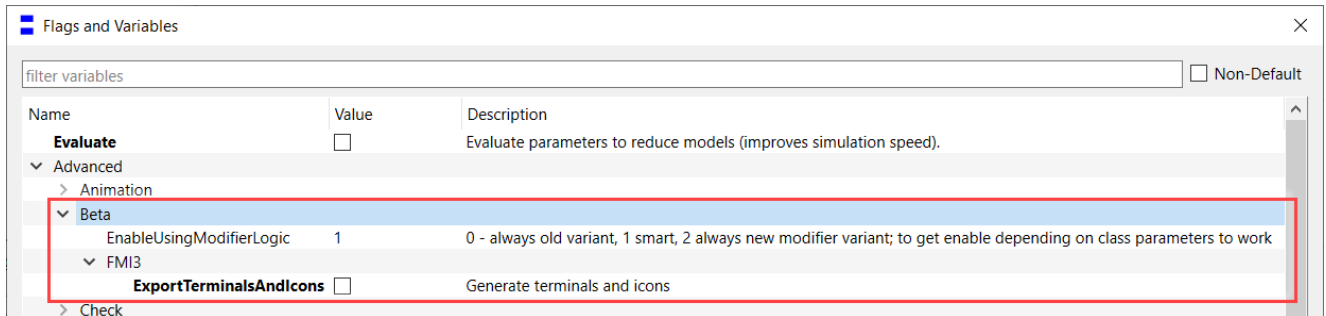
3.5 Features under Development

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 are grouped by `Advanced.Beta` flags in **Tools > Options > Variables...**:
An example from Dymola 2024x Refresh 1:



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 2024x Refresh 1, the following “under development” features are available:

Better handling of enabling the edition of 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

For this feature, see section “Terminals and icons” on page 60.

3.6 Model Management

3.6.1 Encryption in Dymola

Standardized Modelica library encryption

Dymola support the current Modelica Change Proposal MCP-0039 Licensing and encryption, which is planned for a future version of the Modelica language specification. The intention is to provide a standardized library format for multiple Modelica tools, thereby reducing the overhead of distributing libraries.

The current status of the project can be found here: <https://github.com/modelica/Encryption-and-Licensing> and a short overview follows below. End-users of the library do not need to worry about the details; they only need to know the file ending to look for when opening the packaged library: `.mol` files.

In contrast, library vendors need to master the internal details.

In short, this proposal describes a container for distributing Modelica libraries and a protocol for how a Modelica tool should communicate with an executable for licensing and decryption of the library. The use of such a mechanism is as follows:

- The library vendor encrypts the library and provides a licensing mechanism.
- The library vendor also provides a “library vendor executable” (LVE) that is supplied with the library. This LVE handles license checking and decryption.
- Known Modelica tools, for example Dymola, can communicate with the LVE to read the library contents. A tool vendor becomes known to a library vendor by providing a tool-specific public key.

Minor improvements

Library encryption improvements

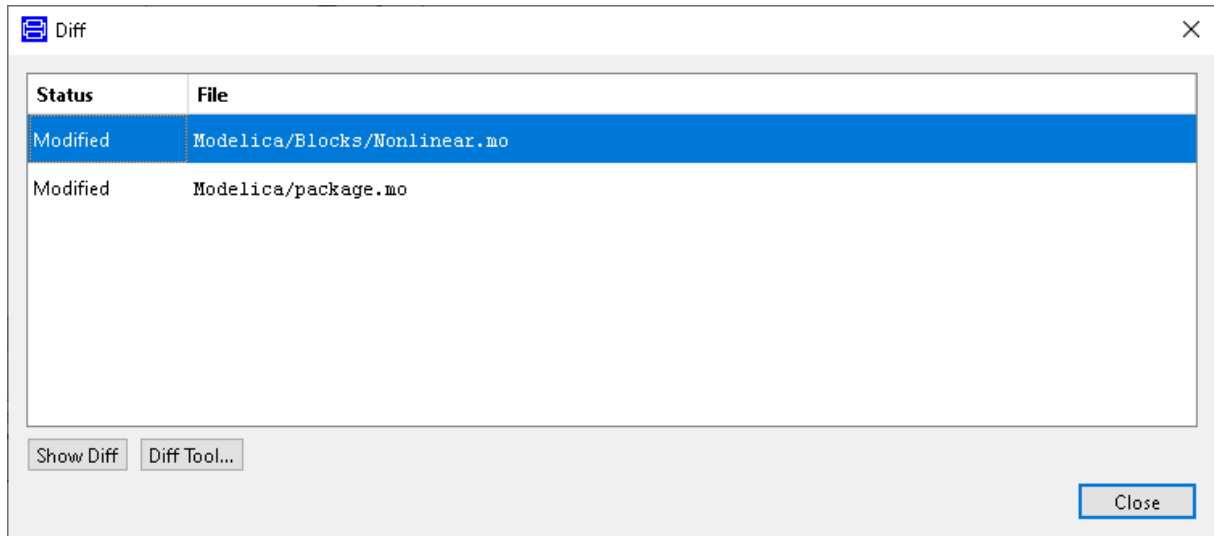
In previous Dymola versions, if the top-level protection annotation `showVariables` was set to `false` in any encrypted library used by a model, components from all other encrypted libraries in this model were treated the same, that is, all protected variables of components from encrypted libraries were concealed.

In Dymola 2024x Refresh 1, this is not the case anymore, only the protected variables of components from encrypted libraries with the protection annotation `showVariables=false` are concealed, the protected variables of components from other encrypted libraries can be stored and plotted.

3.6.2 Extended Git support

Git diff using GUI

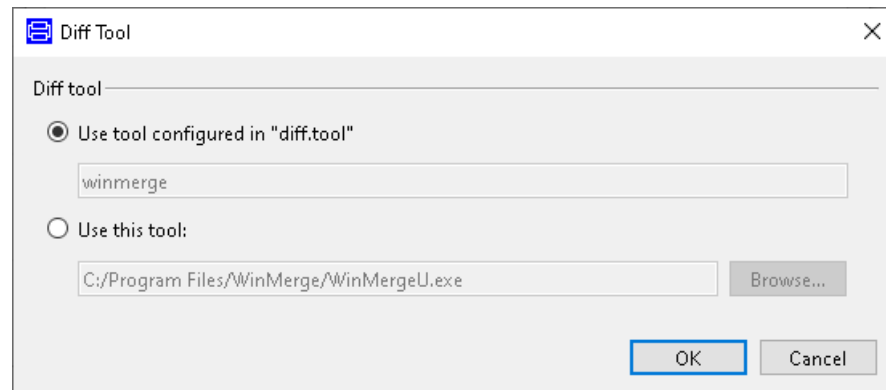
When selecting the command **Version > Diff**, a dialog opens that shows the modified files.



Double-click on a row, press **Show Diff**, or press **Enter**, to show the changes in the selected file. The changes are shown using an external diff tool.

When running **Version > Diff** from a top-level class, the dialog will be shown and all changes in the package are shown. When running it from a subclass, the diff tool is opened directly, since there can only be one modified file.

Which diff tool to use can be configured by pressing the button **Diff Tool....** This button is also available in the **Options** dialog.



There are two choices. You can use the diff tool configured by the option `diff.tool` in git config, or you can use the diff tool specified in the global string `Advanced.File.VersionDiffPath`. The latter should contain the path to the diff tool executable, for example, "C:/Program Files/WinMerge/WinMergeU.exe".

Only two diff tools are currently supported: **WinMerge** and **KDiff3**.

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

The following improvements are available when using “Design with Dymola” in 3DEXPERIENCE 2024x FD02 together with Dymola 2024x Refresh 1.

Option to customize the working directory for libraries opened from 3DEXPERIENCE

To select the working directory, you can use the flag `Advanced.File.PowerBy.Directory`. The default value is an empty string `""`. If the flag has the default value, the folder used is:

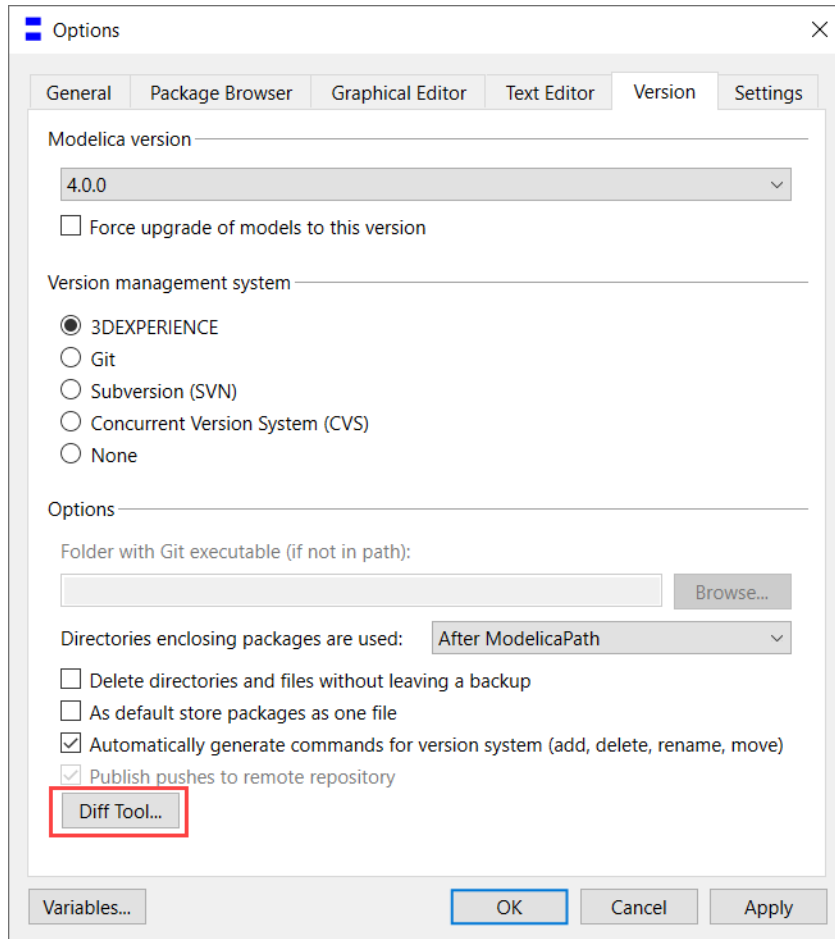
```
%userprofile%\Documents\Dymola\3DEXPERIENCE
```

Using an external tool for comparison and merge of Modelica files

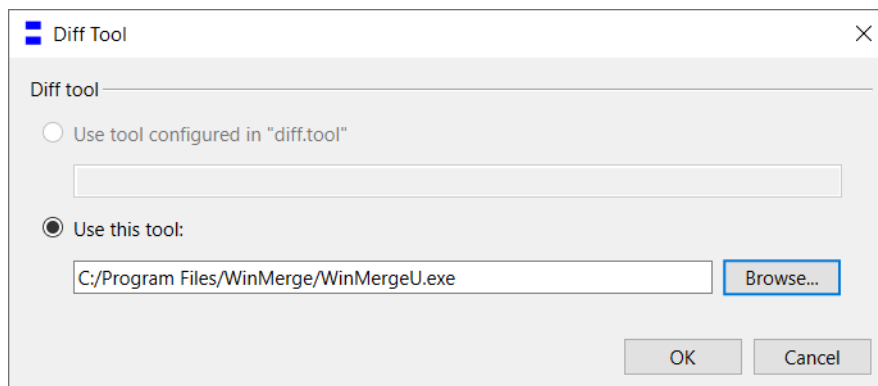
You can use an external tool for comparison and merge of Modelica files. The path to the tool is set by the flag `Advanced.File.VersionDiffPath`. The default path is an empty string `""`. If the flag has the default value, comparison/merge is not available in Dymola.

If you use any of the tools **WinMerge** or **KDiff3**, the flag should contain the path to the diff tool executable, for example, `"C:/Program Files/WinMerge/WinMergeU.exe"`.

To set the flag, you can use the new button **Diff Tool...** in the **Options** dialog, reached by the command **Tools > Options**, the **Version** tab:



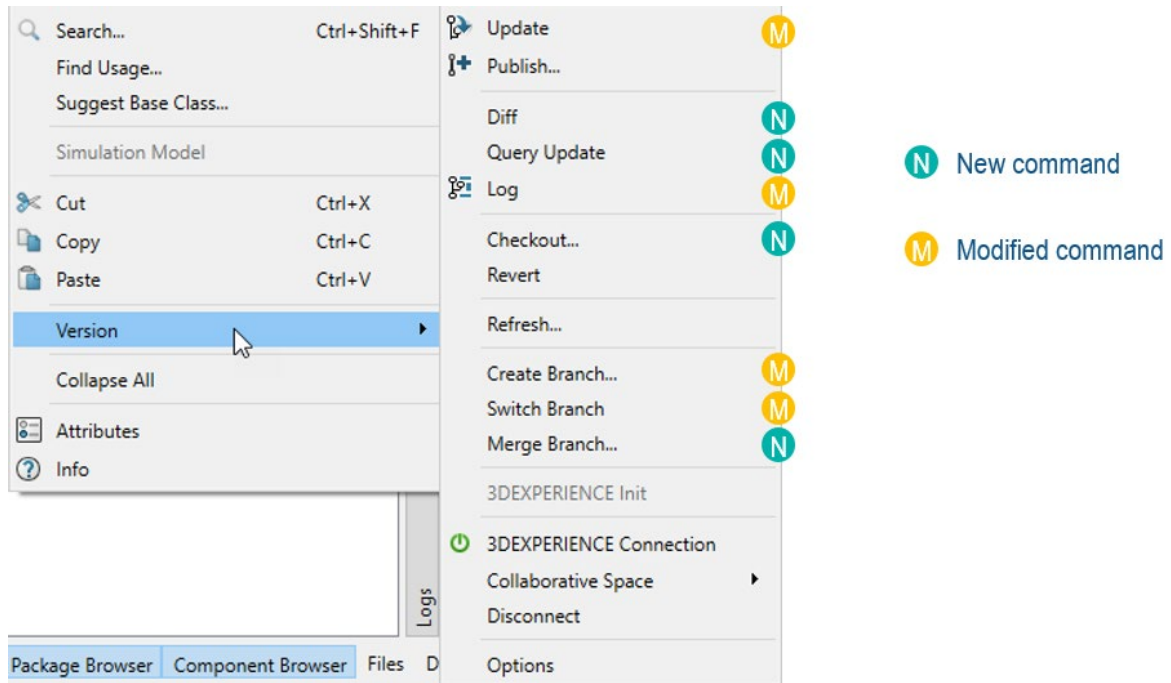
An example of using this button:



For other tools than **WinMerge** and **KDiff3**, you must specify the program full path with specific arguments `%local` `%remote` `%target` respectively, to specify the local, remote and target files.

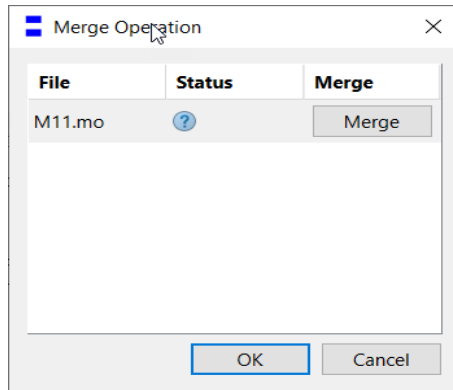
New and modified versioning commands

Right-clicking a model in the package browser and pausing over the **Version** command now gives:



Improved command: Update

The context command updates the local model with a more recent one in the current branch. The enhancement is that in case of a conflict, for example, if a file has been modified both locally and remotely in the database, a merge tool can be used to solve the conflict. An example of the result of clicking the **Update** command:

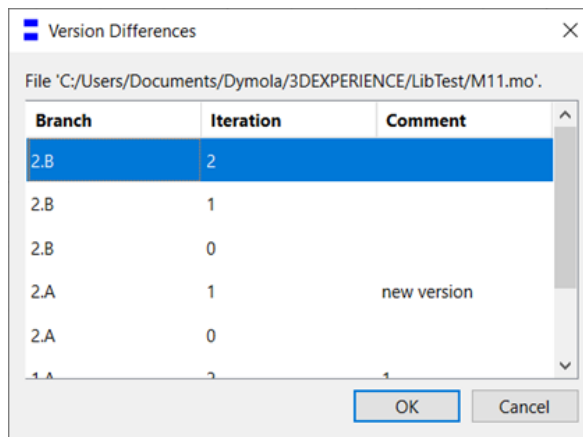


Clicking **OK** opens a merge tool, if specified. See “Using an external tool for comparison and merge of Modelica files” on page 46.

New command: Diff

The new context command **Diff** launches an external diff tool, to compare differences between local files and with remote iteration.

If the command is launched on a package (or root package), it compares all subfiles. An example of the result of clicking the **Diff** command:



For more on the diff tool, see “Using an external tool for comparison and merge of Modelica files” on page 46.

Note. If no external tool is available, then the **Diff** command is hidden.

New command: Query Update

The new **Query Update** context command displays logged information about updates. An example can be:

```
No update. #2 is the latest iteration.
```

or

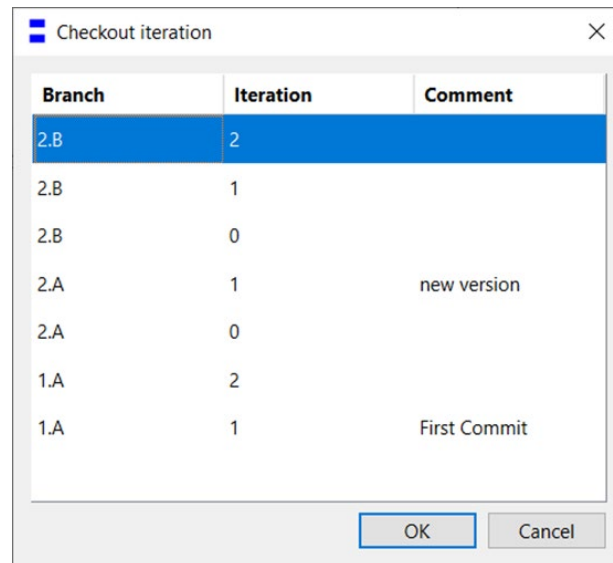
```
-Latest iteration is #3.  
* Files in conflict:  
M11.mo
```

Improved command: Log

The context command **Log** now displays the whole history, not just the local data on the computer.

New command: Checkout

With the new context command **Checkout...** you can open any iteration of a model from the database. An example of the result of clicking **Checkout...**:



Improved command: Create Branch

The context command **Create Branch...** now displays a dialog with new options:

- Creation of a new branch, or create new revision in the current branch.
- The Modelica version can be modified directly from this dialog.

An example of the dialog:

Create Branch

×

Branch or Revision origin

Created from the current revision: 1.A #4

Branch or Revision choice

☒ Branch
☐ Revision

Modelica 'version' annotation

Current library version 1.0.5

New library version 1.0.6 Intermediate

Description

Revision description:

OK

Cancel

Improved command: Switch Branch

The context command **Switch Branch...** displays a dialog where now a new column with the Modelica version has been added.

Example:

Switch Branch

×

Fetch remote branches...

Select the new branch:

Branch	Title
1.A	LibTest
2.B	LibTest 1.0.1
2.A	LibTest 1.0.1

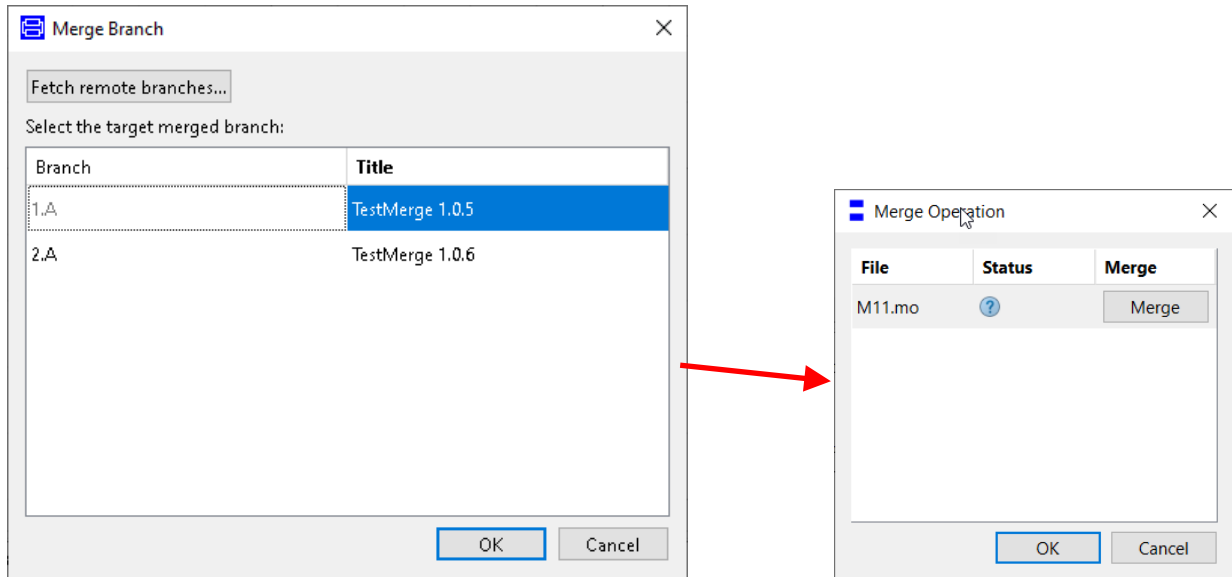
OK

Cancel

New command: Merge Branch

The new context command **Merge Branch...** allows you to merge a branch(/revision) with another one. Modification of current iteration will be included to the other branch; the merge tool is launched if needed (and if it is available).

An example where the merge tool is needed:



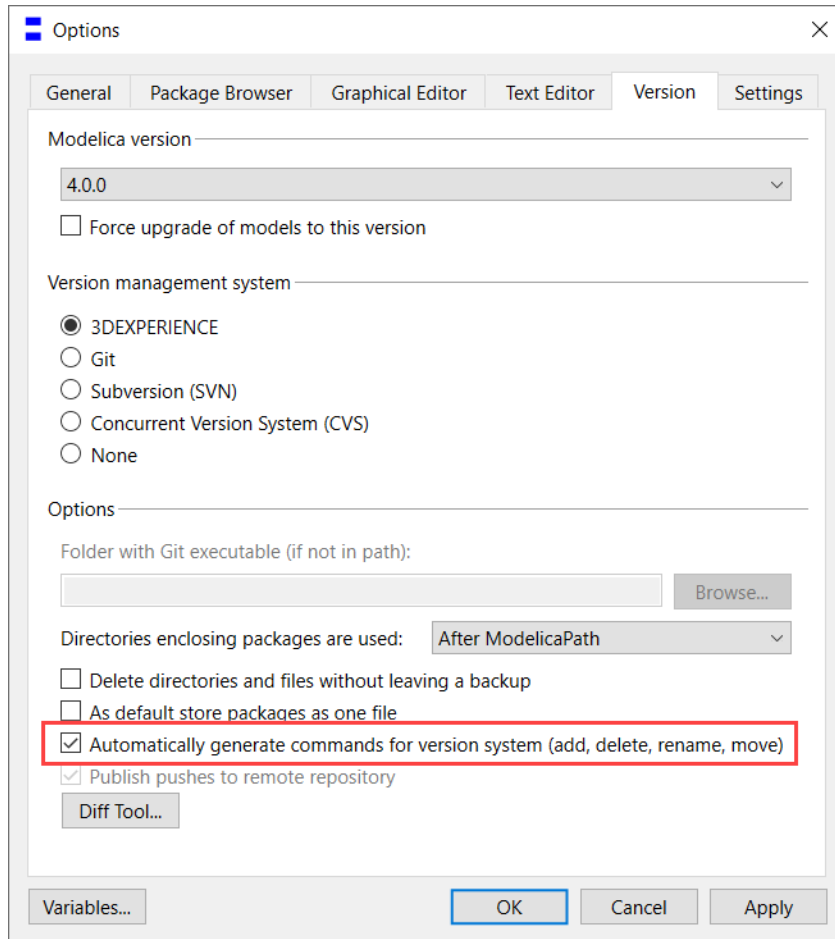
For more about the merge tool, see “Using an external tool for comparison and merge of Modelica files” on page 46.

Improved command: Rename

In previous version, when you renamed a model stored in the database, using, for example, the context command **Rename**, the original file was deleted, and a new file with the new name was created. This meant that there was no traceability for renaming.

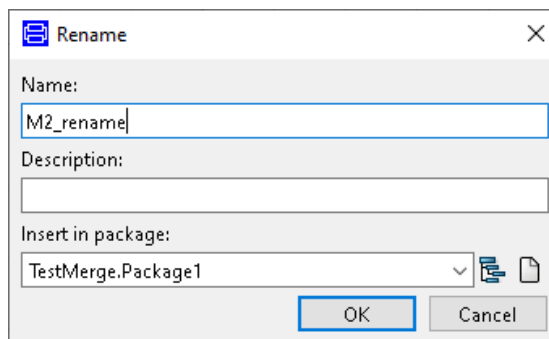
In Dymola 2024x Refresh 1, when you rename a model stored in the database, it is instead seen as moved to a new file with the new name. This means that there are now traceability for renaming.

Note. For this to work, you must activate the setting **Automatically generate commands for version system (add, delete, rename, move)** in the **Options** dialog:

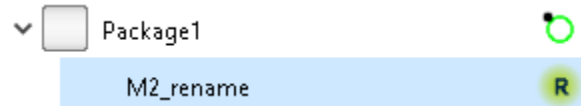


If you rename a package with submodels, all submodels are seen as renamed/moved.

As an example, if you rename a model M2 to M2_rename:

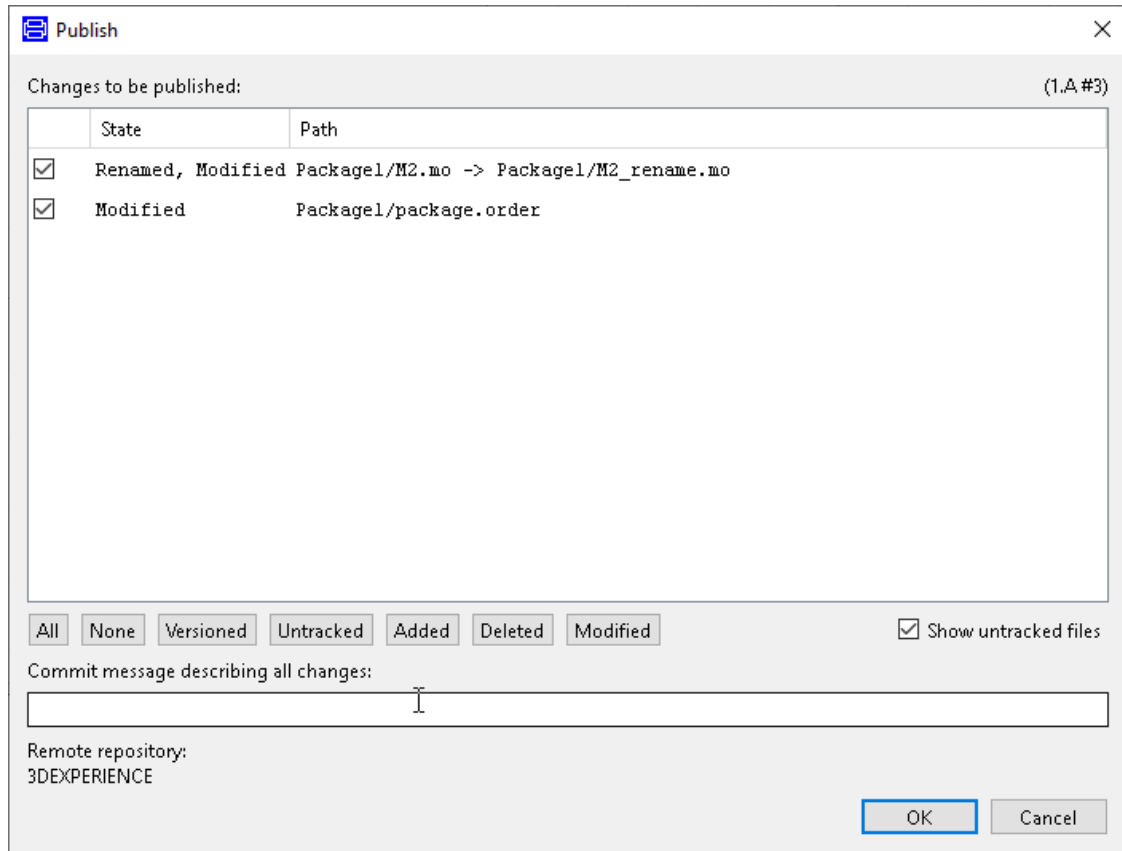


You will get the following in the package browser:



(The green R indicates rename.)

When you publish the renamed model, you will get:



3.7 Other Simulation Environments

3.7.1 Dymola – Scilab interface

Dymola 2024x Refresh 1 introduces support for a Scilab interface.

Result MAT-files issued by Dymola simulation (`dsres.mat`) can now be imported and post-processed in Scilab using similar functions as “Matlab Dymtools”.

“Scilab Dymtools” can typically be found in:

```
C:\Program Files\Dymola 2024x Refresh 1\scilab\dymtools\
```

To use “Scilab Dymtools”, please first download Scilab 2024.0.0 (or newer) from <https://www.scilab.org/> and install it.

Then load Dymtools using the following command:

```
exec("C:\Program Files\Dymola 2024x Refresh 1\scilab\dymtools\loader.sce", -1);
```

The following message will be displayed:

```
Loading Dymola simulation results post-processing tools
Run 'dymbrowse()' to launch GUI.
```

Then you can use the following Dymtools functions:

- dymbrowse: Interactively browse, plot and compare Dymola simulation results,
- dymget: Loads trajectory for specific variable into Scilab workspace,
- dymload: Loads trajectory into Scilab workspace.

If you want additional information, enter **help dymtools** in Scilab console.

3.7.2 Dymola – Matlab interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2019a (ver. 9.6) up to R2023b (ver. 23.2). On Windows, only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. On Linux, the gcc compiler is supported. The LCC compiler is not supported, neither on Windows nor on Linux.

3.7.3 Real-time simulation

Compatibility – dSPACE

Dymola 2024x Refresh 1 officially supports the DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2024x Refresh 1 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 R2022b and R2023a
- dSPACE Release 2023-B with Matlab R2022b, R2023a, and R2023b

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

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

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

Compatibility – Simulink Real-Time

Compatibility with Simulink Real-Time has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R2019a (Simulink Real-Time ver. 6.10) to R2023b (Simulink Real-Time ver. 23.2). Only Microsoft Visual C compilers have been tested.

3.7.4 Java, Python, and JavaScript Interface for Dymola

Java interface updated

The Java interface is updated:

Functionality

- The minimum supported version is Java 8.
- The function `readTrajectory()` can now handle large results.
- It is now possible to select log level. The default log level is that warnings and errors are shown.

Closing Dymola

The Java interface automatically starts a hidden instance of Dymola. It needs to be closed. Either call `close()` explicitly, or use try-with-resources.

In earlier versions of the Java interface, `finalize()` was used. That method is now deprecated and has been removed from `DymolaWrapperInternal`. The Java interface now implements `AutoCloseable`. The example `DymolaExample2.java` contains an example of try-with-resources. The method `close()` is automatically called when the try-block goes out of scope.

Documentation

The Javadoc API documentation has been updated. Private classes and variables are hidden. Deprecated HTML tags have been replaced.

Python interface updated

The Python interface is updated:

- Python 3.7 and higher is now supported.
- Python 2 is not supported anymore.
- The Python interface function `list()` has been renamed to `listvariables()` due to name clash.
- The function `SetVariable()` has been added to the Python interface. It sets the value of a variable to a bool, int, float, str, list, or Expression.
- The old Python interface was distributed in **egg** format. It is a deprecated package format. The updated Python interface uses the newer **wheel** format. The **wheel** for the Python interface is available in the same location as previously **egg**. It can be installed using `pip`, or added to the `PYTHONPATH`. (It should not be installed automatically by the installer, since you can only have one version per library installed.)

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 25.

3.7.5 SSP Support

Support for multiple System Structure Descriptions (SSDs) in an SSP file

Dymola 2024x Refresh 1 supports SSP files that contain multiple System Structure Descriptions (SSDs). One Modelica model is created for each SSD. In earlier versions of Dymola, only the first SSD in an SSP file was read.

SSP parameter and variable representation

A recent discussion in the SSP design group clarifies that:

- Parameters shall be represented by connectors with `kind="parameter"` in SSP.

- Public local variable declarations are represented by connectors with `kind="local"`.

Consequently, parameter propagation, e.g. from system to a component element, is described by a connection. Example:

```
<ssd:Connection startConnector="T2" endElement="clutchSystem"
endConnector="T2"/>
```

This is now handled in Dymola.

Storing the start value of a parameter

SSP has parameter values, but the more advanced notion that a parameter can have a start value but no value is not handled. For that reason, there is a flag in Dymola `Advanced.SSP.StoreStartAsValue` (default `false`) that stores the start value if a "real" value is missing.

If the SSP contains a component that is a Modelica library type, then the component is exported as a reference to a Modelica component, which of course is only useful if imported into a Modelica environment. In such a case, start values are preserved instead of being converted to value.

Warnings given when trying to set input or output connectors

Sometimes SSP models contain settings of component inputs and outputs. This is meaningless in a Dymola context and a warning is given. It should be noted that unconnected outputs are well defined in Modelica, but every input connector **must** be connected to.

Example of warnings:

```
Warning: Output connector level1.tankDemo.TankPhysicalPart.tank2Level is given
value 0.0 (ignored)
Warning: Input connector level1.tankDemo.TankPhysicalPart.openValve1 is given value
false (ignored)
If it is unconnected, use a Modelica.Blocks.Sources.BooleanExpression e(y=false)
```

CSV file format supported for SSP

System Structure and Parameterization (SSP) defines parameter sets in SSV format (an XML encoding) as described in the SSP specification. In addition, Dymola support storage of parameter values in CSV (Comma Separated Value), in addition to SSV.

The SSP parameter binding should contain the appropriate **source** and **type** attributes. In addition, **prefix** and **sourceBase** attributes are supported with the usual meaning according to SSP.

```
<ssd:ParameterBinding source="parset.csv" type="text/csv">
```

The designated CSV shall specify the parameter name (column 1) and parameter value (column 2). Optionally each row can specify the unit of the value (column 3) and a description (column 4). The file shall not contain a header row. Example:

```
J1.J, 1.0
J2.J, 0.8, , "Medium light (default unit)"
T2, 600, "ms", "Use of a scaled SI unit"
```

Note that parameter names are given as the full path, using dot-notation to designate nested parameter.

Unit handling for parameters in SSP

Parameters can specify a unit of the value of a parameter, expressed in XML,

```
<ssv:Parameter name="T2">
  <ssv:Real value="400" unit="ms"/>
</ssv:Parameter>
```

This is handled in Dymola:

- The unit attribute of the parameter is used as the displayUnit attribute in Modelica; the assumption is that proper variable/parameter declarations use types that have the appropriate units.
- The value is scaled accordingly, because a Modelica modifier requires the value in the base unit.
- It should be noted that the unit handling uses the unit definitions in Dymola, and does not depend on unit attributes defined in the SSP file.

The Modelica code generated from the above example is the modifier:

```
T2(displayUnit="ms")=0.4
```

Handling value attribute in SSP

Parameter sets in SSP are sometimes using the attribute `value` to represent the value of a parameter, especially if there are other attributes. For example:

```
x.value = 5.6 kV
x.min = 0
```

The correct representation in Modelica after importing an SSP would be:

```
x(displayUnit="kV", min=0)=5600
```

If you set the flag

```
Advanced.SSP.SupportValueParameter = true
```

You will get the correct Modelica representation of the attribute `value` when you import the SSP. (The default value of the flag is `false`.)

FMUs in an imported SSP collected in a subpackage

Previously, any FMUs in an imported SSP were collected directly under the created SSP package. In Dymola 2024x Refresh 1, they are by default collected in a subpackage `FMU` under the SSP package.

This behavior is controlled by the variable `Advanced.SSP.SubpackageFMU`. The variable is of type `String`, and the default value is `"FMU"`. If you want the folder name `Components` instead, you can set the variable as:

```
Advanced.SSP.SubpackageFMU = "Components"
```

If you set the variable to an empty string "", the FMUs are handled like in previous versions of Dymola.

The variable is saved between sessions.

3.7.6 FMI Support in Dymola

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

Support for FMI 3.0

General

Dymola 2024x Refresh 1 includes limited support for FMI version 3.0, with support for the new features described in the below sections. Note that all existing import and export features in Dymola are implemented in FMI 3.0.

Note!

The declaration order of alias variables in FMI 3.0 will not match the order of the original variables, due to a restriction in the specification. Thus, multibody animations of imported FMUs will not work.

For general information about FMI 3.0, see:

- [The FMI 3.0 specification](#)
- [A paper describing the new features of FMI 3.0](#)

These links are also available using **Tools > Help Documents**.

Terminals and icons

***Note!** This feature is in the beta-stage and performance and reliability will later be improved. As beta feature, it is not documented in the manual.*

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.)

Event mode co-simulation

An *exported* FMI 3.0 FMU supports early return for events.

An *imported* FMI 3.0 FMU supports event mode co-simulation if the following is satisfied:

- The flag `Advanced.FMI3.EventModeCoSim` is set to `true`. (The default value of the flag is `false`.)
- The imported FMU indicates that it supports:
 - Returning early for events
 - Getting and setting the states
 - Variable size communication step size.

The imported FMU has some additional hooks, since it is necessary to propagate the potential early return to other parts of the model (including other co-simulation FMUs).

(This feature was a beta feature in the previous Dymola version. The flag has changed name in Dymola 2024x Refresh 1.)

Co-simulation: Intermediate update

Dymola supports intermediate update for co-simulation FMUs. Setting the flag `Advanced.FMI.SetInputDerivatives=true` interpolate inputs for FMI 3 co-simulation, similarly as input derivatives for FMI 2. (The default value of the flag is `false`.)

Input interpolation is supported by FMI 3.0 FMUs generated by Dymola.

This feature was already available in Dymola 2023x Refresh 1.

Co-simulation: Early return

Dymola 2024x supports early return for an *exported* FMU, that is, to stop the execution of `fmi3DoStep` and return without reaching the predefined communication time.

Co-simulation: Event handling

Dymola 2024x supports event handling for an *exported* FMU.

Co-simulation: Variable step communication interval

See “Variable step co-simulation” below.

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.

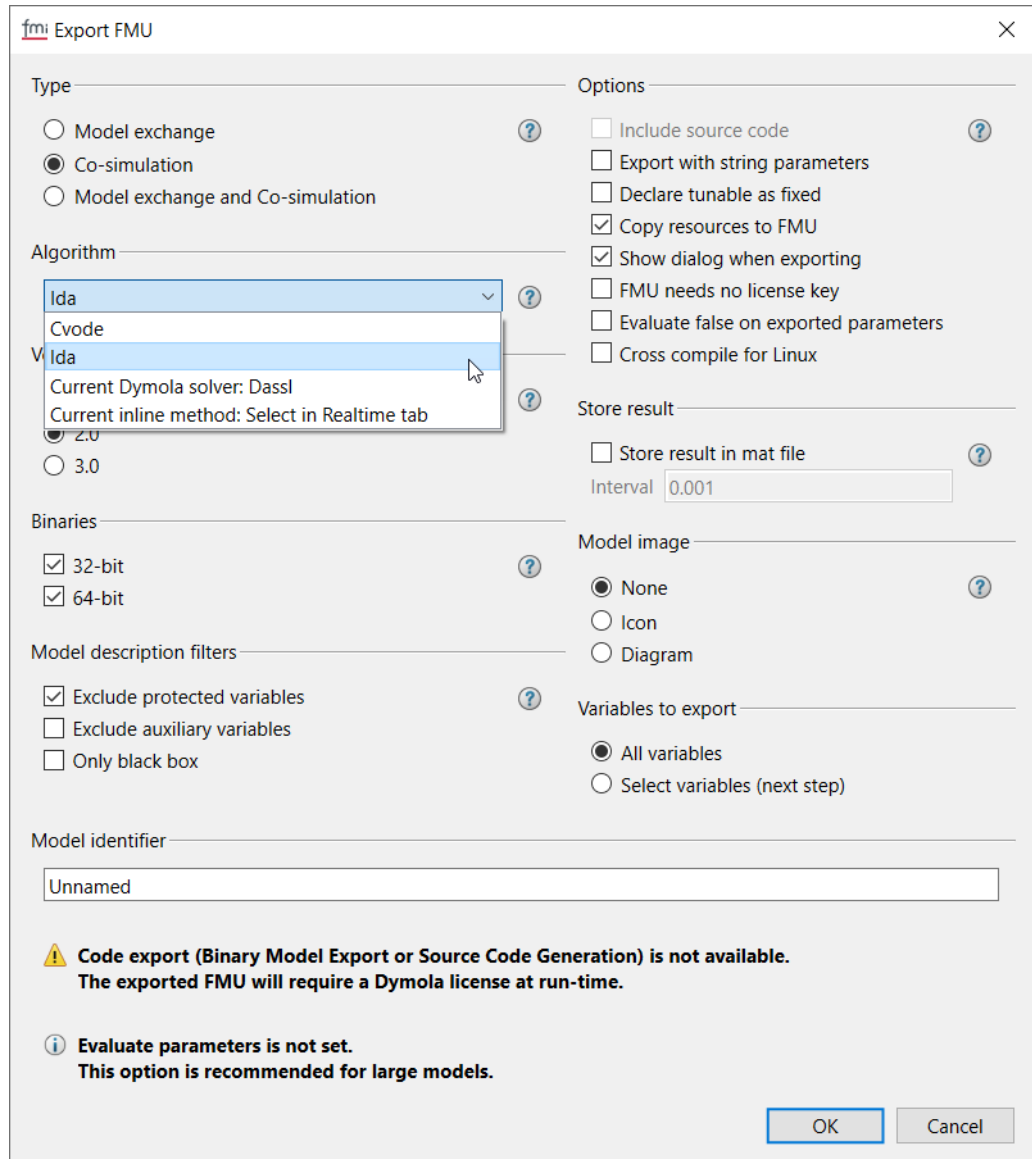
FMI Export

SUNDIALS solver Ida available for FMU export

The SUNDIALS solver Ida is now available alongside Ccode as co-simulation algorithm in our most powerful FMUs. (Previously, when only Ccode was available, this selection was referred to as **Co-simulation using Ccode**.)

Ida can be used both in ODE mode and in DAE mode.

The Export FMU dialog that appears when you export an FMU has been clarified and the Ida option has been added. An example of selecting Ida when exporting an FMU (for all options, see section “Improved GUI for exporting an FMU” on page 65):



First, you have to select the type of FMU. You can select **Model exchange**, **Co-simulation**, or **Model exchange and Co-simulation**. In this example, we select **Co-simulation**.

Then, you can select the algorithm. Having selected **Co-simulation** as first step, you can now select any of **Cvode**, **Ida**, **Current Dymola solver**, or **Current inline method**.

When selecting **Cvode** or **Ida**, you can export FMUs with both a model exchange interface and co-simulation interface.

The **Current Dymola solver** option picks the Dymola solver as selected in the **General** tab of the "Simulation Setup". Note that Dymola solvers cannot be combined with model exchange or source code export.

The final option **Current inline method** picks the current inline method as defined in the **Realtime** tab of the "Simulation Setup".

For how to handle Ida in scripting, see section “The built-in function translateModelFMU improved” on page 26.

Option to prevent evaluation of exported parameters when variable filtering is applied

The screenshot shows the 'fmi Export FMU' dialog box with the following settings:

- Type:** ☒ Model exchange and Co-simulation
- Algorithm:** Ccode
- Version:** ☒ 2.0
- Binaries:** ☒ 32-bit, ☒ 64-bit
- Model description filters:** ☒ Exclude protected variables, ☐ Exclude auxiliary variables, ☐ Only black box
- Options:** ☐ Include source code, ☐ Export with string parameters, ☐ Declare tunable as fixed, ☒ Copy resources to FMU, ☒ Show dialog when exporting, ☐ FMU needs no license key, ☐ Evaluate false on exported parameters (highlighted with a red rectangle), ☐ Cross compile for Linux
- Store result:** ☐ Store result in mat file, Interval: 0.001
- Model image:** ☒ None, ☐ Icon, ☐ Diagram
- Variables to export:** ☒ All variables, ☐ Select variables (next step)
- Model identifier:** Modelica_Mechanics_Rotational_Examples_CoupledClutches

Warnings:

- Code export (Binary Model Export or Source Code Generation) is not available.** The exported FMU will require a Dymola license at run-time.
- Evaluate parameters is not set.** This option is recommended for large models.

Buttons: OK, Cancel

When exporting an FMU, activating this option prevents evaluation of parameters that are included in any filtering of variables. You perform such filtering when you activate **Select variables (next step)** in the dialog above. The idea with activating this option is to be able to change these exposed parameters in the exported FMU. The setting is by default not activated, the reason being that preventing evaluation may cause some models to fail when being translated. The option not activated corresponds to the flag

`Advanced.FMI.EvaluateFalseExposedParameters = false`. The option and the flag is reset when starting a new session.

Meta data included in an exported FMU

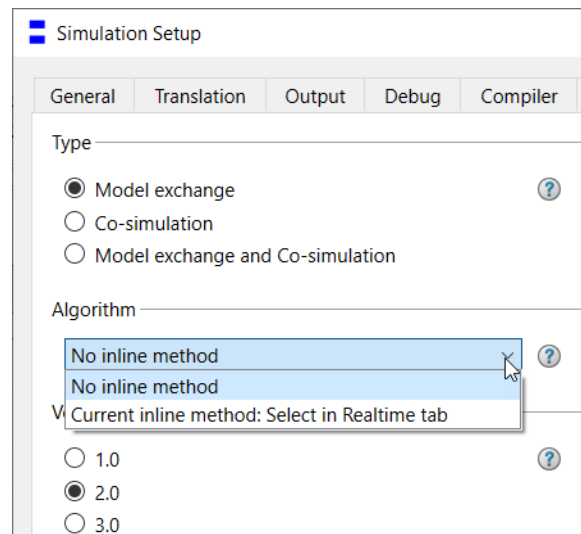
If meta data is available in a Modelica model, and an FMU is generated from this model, the meta data is stored as an SMRD (Simulation Resource Meta Data) file in the FMU archive, in a folder `extra`.

Improved GUI for exporting an FMU

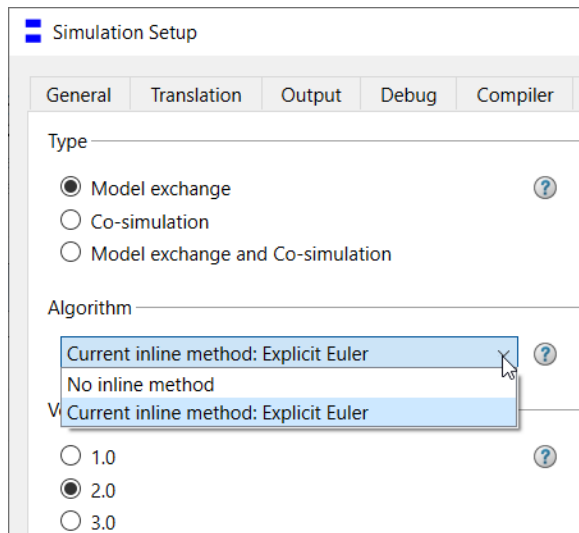
The GUI for exporting an FMU has been improved. This GUI is available both when exporting an FMU and in the general menu of FMU export in the simulation setup (reached by the command **Simulation > Setup**, the **FMI Export** tab). The main change is that it is now possible to select **Algorithm** for the different FMI types. Looking at the possible selections for the FMI type and algorithm in the simulation setup, the general possible selections are:

Model exchange.

For the FMI type **Model exchange**, the default selection of the algorithm is:

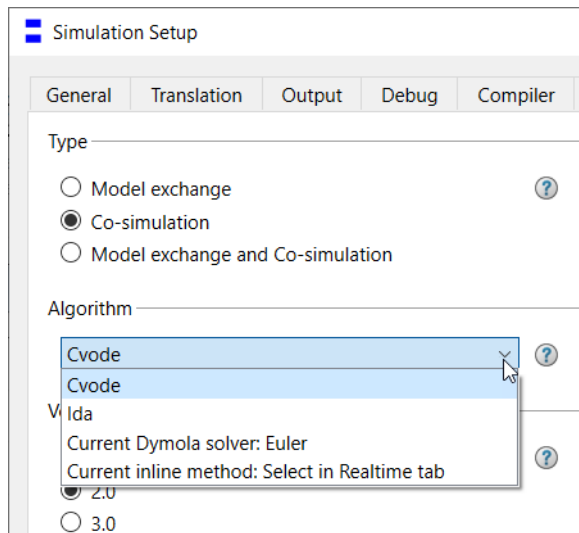


If you select an inline method in the **Realtime** tab, the selection can look like (that selection then being used also here):



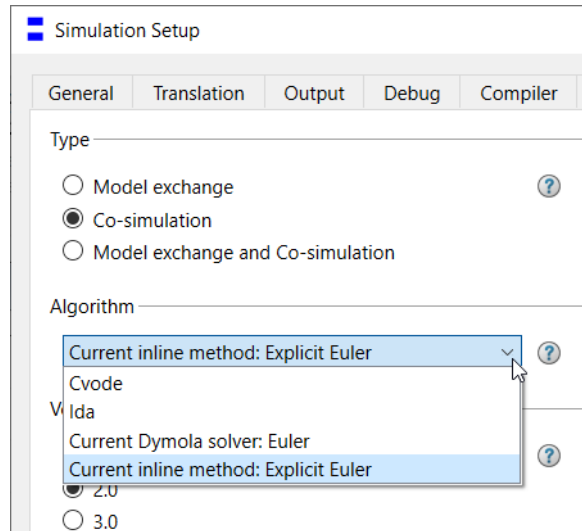
Co-simulation.

If you instead select **Co-simulation** as the FMI type, the default algorithm selection is:



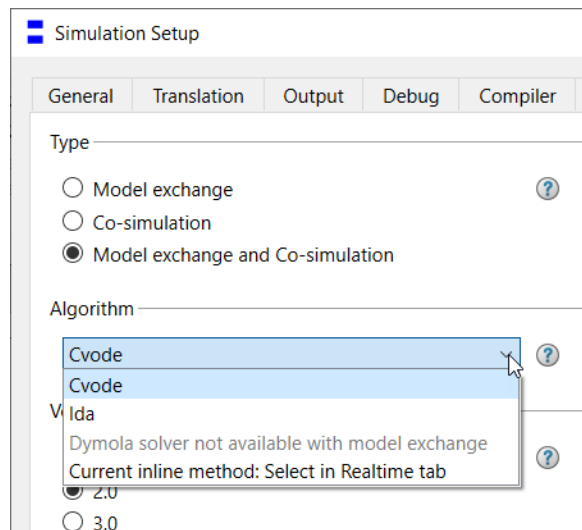
Note that if you select **Co-simulation**, and then any algorithm *except* **Current Dymola solver**, such a selection corresponds to the selection **Co-simulation using Cvode** in previous Dymola versions (except that Ida was not selectable this way in previous versions).

If you select an inline method in the **Realtime** tab, the selection can look like (that selection then being used also here):



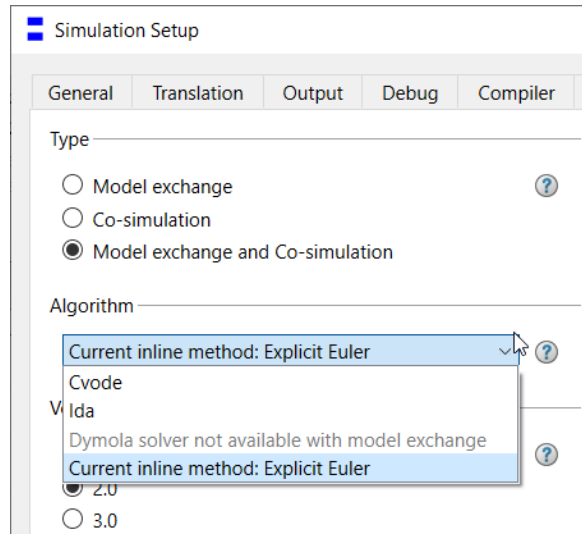
Model exchange and Co-simulation.

If you select **Model exchange and Co-simulation** as FMI type (that is the default selection), you get the default:



Note that this selection was previously named **Model exchange and Co-simulation using Cvode**, but since now also Ida is available, the name has been changed to just **Model exchange and Co-simulation**.

(Selecting inline method in the **Realtime** tab gives, as example:



Note that the former option to select, as FMI type, **Co-simulation using Dymola solvers**, has disappeared, this selection is covered instead by the **Algorithm** options for the FMI type **Co-simulation**.

The GUI that appears when you export an FMU has been changed in the same way as the simulation setup.

FMI import

Variable step 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. In Dymola 2024x, you can change `dostepTrigger` to a variable that can be externally triggered. You do this by setting the flag:

```
Advanced.FMI.ExternalDoStepTrigger = true
```

(The flag is by default `false`.)

Note that 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.

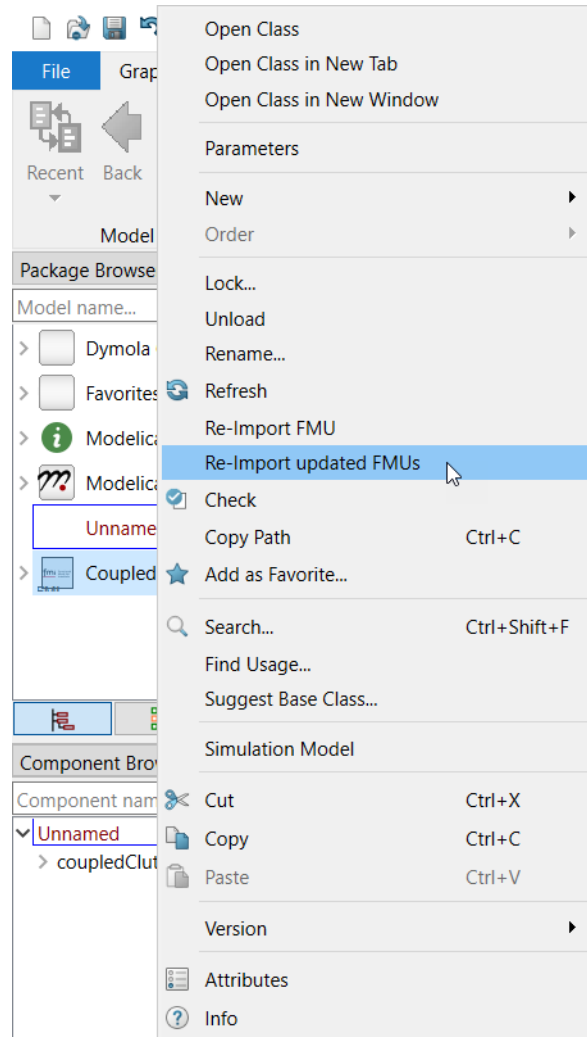
(This feature was a beta feature in the previous Dymola version. The flag has changed name in Dymola 2024x Refresh 1.)

Meta data read from an imported FMU

If an imported FMU contains meta data, that data is read, and added (as annotations) in the wrapper class of the generated Modelica model.

Option to re-import FMUs if needed

You can now right-click any imported FMU in the package browser and apply a new command **Re-import updated FMUs**. The command re-imports all FMUs that have newer time stamps than the imported ones.



The result of the re-import is logged in the command window.

Notes:

- If the current Dymola version differs from the one used when importing the FMUs the first time, the FMUs are also re-imported. This functionality can be disabled by setting the flag `Advanced.FMI.ReImportFMUIfDymolaVersionDiffer=false`. (The flag is by default `true`.)

- Older FMUs, that is, FMUs that were imported before this feature was implemented, are seen as outdated and will be reimported. A new time stamp will be added to the FMUs in the operation.

3.7.7 eFMI Support in Dymola

The main improvement for Dymola 2024x Refresh 1 is the seamless integration of [Software Production Engineering](#) on [3DEXPERIENCE](#) for eFMI Production Code container generation, as described in detail in the next section. Further improvements are listed in the next but one section.

Support for Software Production Engineering on 3DEXPERIENCE

With this Dymola release, we finally officially support [Software Production Engineering](#) on [3DEXPERIENCE](#). “Software Production Engineering” – which is available in [3DEXPERIENCE](#) by role “Systems Software Production Engineer” (SOP-OC) – is the final name for what we called “CATIA ESP” in the past; it comprises all functionality of the former “CATIA ESP prototype”, which now is deprecated.

Software Production Engineering is a separate Dassault Systèmes product and requires a separate license. It is not covered by any [Dymola](#) license, including the Dymola Source Code Generation License. If you like to continue using the eFMI Production Code generation capabilities of the former CATIA ESP prototype – as needed for convenient software-in-the-loop testing of eFMUs in Dymola – you need to purchase a Systems Software Production Engineer license and use the [3DEXPERIENCE](#) platform. The Software Production Engineering service on [3DEXPERIENCE](#) will be available from April 29, 2024.

Important!

Like in previous releases, Dymola’s eFMI Algorithm Code and Behavioral Model generation facilities only require a Dymola Source Code Generation License; an additional Systems Software Production Engineer license is not required.

Important!

According to the disclaimer of previous CATIA ESP prototype distributions (distributed as “Dymola eFMI kit supplementary”), CATIA ESP prototypes must not be used anymore starting from April 29, 2024, when Dymola provides a seamless integration of Software Production Engineering on [3DEXPERIENCE](#).

Software Production Engineering is licensed as a credit-based product. With purchasing a license, you acquire a number of credits; every time some eFMI Production or Binary Code container is successfully generated, an amount of credits are consumed. If you run out of credits, you can conveniently purchase more. The credit-based approach provides you high flexibility and scales cost for your eFMI-based embedded code generation activities.

Please contact Michael Seibt (Michael.Seibt@3ds.com) for further licensing and sales information about Systems Software Production Engineer.

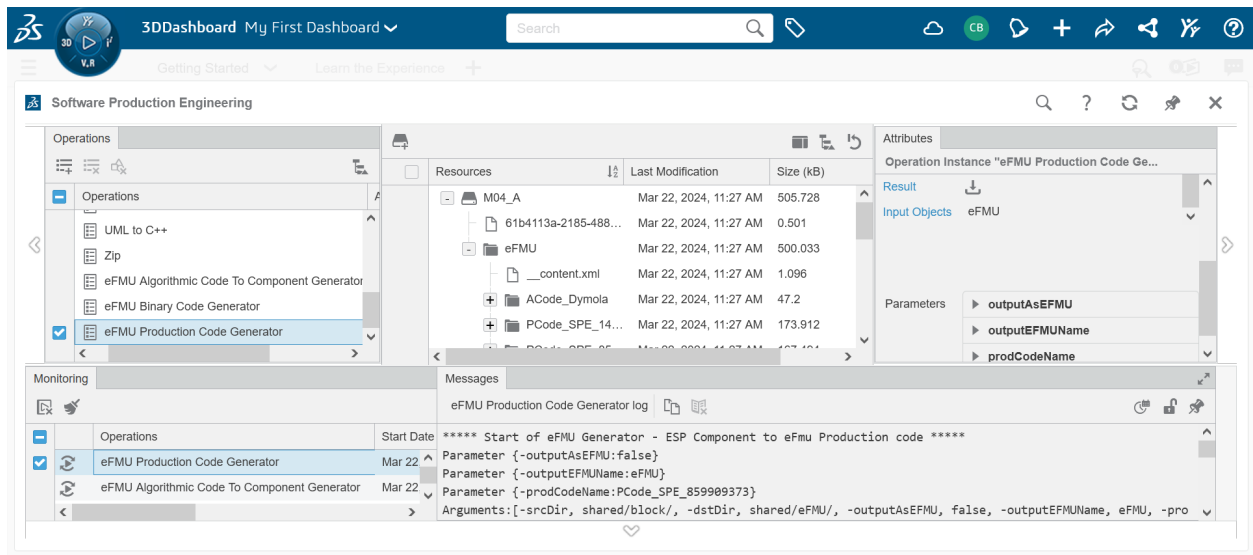
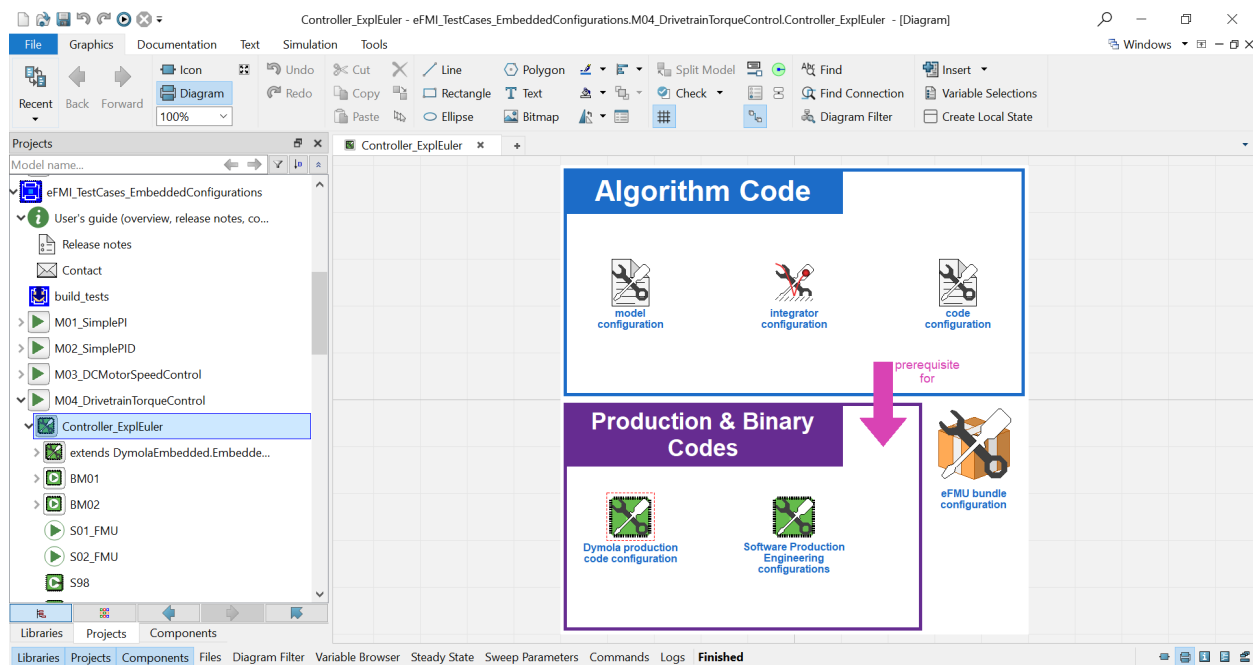


Figure 1: *Software Production Engineering* widget on **3DEXPERIENCE** for an eFMU generated from Dymola for M04 of eFMI_TestCases (upper image: Dymola eFMU generation configuration; lower image: Software Production Engineering widget).

Please consult the requirements documentation of the `DymolaEmbedded` library (`DymolaEmbedded.UsersGuide.Requirements`) for details about the requirements for

using Software Production Engineering from Dymola. In particular, note that its Dymola integration requires a [Java](#) runtime environment version 17 or newer – e.g., an [OpenJDK](#) distribution – in its default installation location – e.g., C:\Program Files\Java\jdk-17.0.2.

For this release, Software Production Engineering on **3DEXPERIENCE** is only used to generate eFMI Production Code containers from Algorithm Code containers; for Binary Code container generation still the CATIA ESP prototype is used, whose Binary Code container generator now ships with Dymola. The bundled Binary Code container generator requires:

- [OpenJDK 1.8.0.265](#) in its default installation location C:\Program Files\Java\jdk-1.8.0.265. A binary version for Windows can be obtained on request from support (see technical support in `DymolaEmbedded.UsersGuide.Contact`).
- [Microsoft Visual Studio](#) installation in its default installation location; any of version 11.0-17.0, i.e., Microsoft Visual Studio 2012-2022.

IMPORTANT: *The bundled Binary Code container generator is subject to, and requires, a Systems Software Production Engineer license (it uses the license that is used for eFMI Production Code container generation via Software Production Engineering on 3DEXPERIENCE). No credits are consumed yet for using the bundled Binary Code container generator, but will be in the future updates/releases; the switch to Software Production Engineering on 3DEXPERIENCE for eFMI Binary Code container generation is scheduled for the next Dymola release.*

Moving to the **3DEXPERIENCE** platform enables us to provide you with future advanced Cloud-based eFMI tooling – including further options to use Systems Software Production Engineer credits in the embedded code generation domain. **3DEXPERIENCE** eases distribution of generated artefacts and workflows and is part of our joint journey to leverage, and early experience, the new disruptive eFMI technology. We highly appreciate your feedback on the Dassault Systèmes eFMI tooling and we will continue to support you with state-of-the-art tooling while defining an eFMI ecosystem and portfolio.

Further eFMI improvements

The following improvements have been implemented for Dymola 2024x Refresh 1:

- eFMU co-simulation stubs can now be generated even if no Production Code and respective Binary Code containers are available; hence, Software Production Engineering (former CATIA ESP) is not required anymore to generate eFMU co-simulation stubs. Stubs without backing production code (i.e., implementation) are non-functional mockups; they compute nothing and just set eFMI GALEC error signals that are by default asserted (cf. `DymolaEmbedded.BuildUtilities.BinaryCode.BinaryStub.__assert_error_signals`).
- Behavioral Model containers can now be generated without Software Production Engineering (former CATIA ESP) generated production and binary code. The eFMU co-simulation stub of the SiL tests of respective eFMU experiment packages are just non-functional mockups; they cannot be simulated (you cannot do a software-in-the-

loop test if there is no software), but the reference experiment can still be used to define and generate a Behavioral Model container.

- Generated eFMU experiment packages now come with preset SiL tests for each available production code (`Test_SiL_Scenario_X_ProductionCodeContainerName`). The original generic SiL tests `Test_SiL_Scenario_X` are still available and can be used for user experiments like recalibration and re-initialization; their checks are by default disabled to encourage such user experiments that would likely cause them to fail.
- Generated eFMU experiment packages now contain documentation about how to use them, which configuration information – like tolerances – users have to provide and how via manual edits and which kind of changes likely cause the Behavioral Model container generation to break.
- Added support for extracting test scenarios from redeclared eBlocks when generating eFMU experiment packages.
- eFMUs exported as FMUs now have their `fmi_CommunicationStepSize` properly preconfigured.
- `eFMI_TestCases_EmbeddedConfigurations` library: Replaced the old hand-written SiL tests with eFMU experiment packages derived from the reference test scenarios provided by `eFMI_TestCases`. The new experiment packages use the 32-Bit and 64-Bit floating-point tolerances as described by the application scenarios of `eFMI_TestCases` and can be used to generate respective eFMI Behavioral Model containers.
- `DymolaEmbedded` and `eFMI_TestCases_EmbeddedConfigurations` libraries updated to version 1.0.3.

3.7.8 Code and model export

The template project *StandAloneDymosim* migrated

The template project *StandAloneDymosim* that describes how to interface models exported by any of the license options “Binary Model Export” or “Source Code Generation” to standard integration routines is now migrated to use the Visual Studio 2015 compiler. Note that newer versions of the Visual Studio compiler are normally supported as well, when opening the project using a newer version of the compiler; the project will be automatically converted.

3.8 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.

3.9 Documentation

General

In the software, distribution of Dymola 2024x Refresh 1 Dymola User Manuals of version “March 2024” will be present; these manuals include all relevant features/improvements of Dymola 2024 Refresh 1 presented in the Release Notes, except the “under development” ones (if present).

Restructuring of Dymola web pages – related compiler and Linux information now in the manuals

The Dymola web pages have been restructured. One consequence is that the specific compiler page (www.Dymola.com/compiler) and the Linux page are now gone. (Clicking any such links, like the compiler one just given, takes you to the Dymola general homepage.)

Most of the information in these pages were already in the manuals, except for some details about installation. That information is now also included in the manual “*Dymola User Manual 1: Introduction, Getting Starting, and Installation*”, the chapter “Appendix – Installation”.

3.10 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2024x Refresh 1 are listed.

3.10.1 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.

3.10.2 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions

Dymola 2024x Refresh 1 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 2024x Refresh 1 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 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 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 code generator for this compiler.
- Visual Studio Enterprise 2019 (16). Note that you can select to use Clang code generator for this compiler.
- Visual Studio Enterprise 2022 (17). Note that you can select to use Clang code generator for this compiler.
- Visual Studio Professional 2022 (17) Note that you can select to use Clang 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 2024x Refresh 1 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.

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

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 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 2024x Refresh 1 supports DSLS R2024x. Earlier DSLS versions cannot be used.

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

Linux

Supported Linux versions and compilers

Dymola 2024x Refresh 1 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 2024x Refresh 1 it is Qt 6.6.0.
- 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 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 2024x Refresh 1 is supported as a 64-bit application on Linux.

Notes:

- 32-bit compilation for simulation might require explicit installation of 32-bit libc. E.g.:
on Red Hat Linux (RHEL): `sudo yum install g++-multilib libc6-dev-i686`
on Ubuntu: `sudo apt install g++-multilib libc6-dev-i686`
- Dymola is built with Qt 6.6.0 and inherits the system requirements from Qt. However, since Qt 6.6.0 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.

Note on libraries

- The library UserInteraction is not supported on Linux.

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 2024x Refresh 1 supports DSLS R2024x. 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.

"