

Dymola

Dynamic Modeling Laboratory

Dymola Release Notes

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

Document version: 1

© Copyright 1992-2022 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 2023	7
3.1	Introduction	7
3.1.1	Additions and improvements in Dymola	7
3.1.2	New and updated libraries	7
3.2	Developing a model	9
3.2.1	Improved search of records	9
3.2.2	Model metadata support	13
3.2.3	Support of Markdown for documentation	14
3.2.4	Minor improvements	16
3.3	Simulating a model	18
3.3.1	Better presentation of timers	18
3.3.2	Steady state solver interface	21
3.3.3	Max run time per simulation	24
3.3.4	Scripting	25
3.3.5	Minor improvements	26
3.4	Installation	39
3.4.1	Installation on Windows	39
3.4.2	Installation on Linux	40
3.4.3	Dymola license server on Windows and Linux	40
3.5	Model Experimentation	41
3.5.1	Improvements for sweeping parameters	41
3.6	Other Simulation Environments	44
3.6.1	Dymola – Matlab interface	44
3.6.2	Real-time simulation	44
3.6.3	Dymosim DLL	45
3.6.4	SSP support	45
3.6.5	FMI Support in Dymola	47
3.6.6	eFMI Support in Dymola	48
3.7	Visualize 3D	49
3.7.1	Surface plots	49
3.8	Modelica Standard Library and Modelica Language Specification	49
3.9	Documentation	49

3.10	Appendix – Installation: Hardware and Software Requirements	50
3.10.1	Hardware requirements/recommendations	50
3.10.2	Software requirements	50

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 2023:

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 50 for more information. **Notes:**

- From Dymola 2020x, Visual Studio C++ compilers older than version 2012 are no longer supported.
- From Dymola 2022x, Visual Studio 2013 is not supported anymore, due to the logistics of supporting multiple old versions for all solvers. (Visual Studio 2012 is however still supported, 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 2023 has limited support for the MinGW GCC compiler, 32-bit and 64-bit. For more information about MinGW GCC, see section “Compilers” on page 50, the section about MinGW GCC compiler.

WSL GCC (Linux cross-compiler)

Dymola 2023 has limited support for the WSL (Windows Subsystem for Linux) GCC compiler, 64-bit. For more information about WSL GCC, see section “Compilers” on page 50, 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 53 for more information.

2 About this booklet

This booklet covers Dymola 2023. 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 2023

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2023. In particular, Dymola 2023 provides:

- Better presentation of timers (page 18):
 - Timer information in a new tab in Simulation Analysis dialog
 - Printing timer values to a text file
 - Simulation setup options for timers
- Improved search of records (page 9)
- Model metadata support (page 13)
- Support of Markdown for documentation (page 14).
- Steady state solver interface:
 - Specifying a start time for the start of dynamic steady-state finding (page 21)
 - Strict steady-state detection tolerance for dynamic steady-state finding (page 21)
- Max run time per simulation (page 24)
- Nonlinear solver statistics available for plotting (page 26)
- eFMI support in Dymola (page 48)
- Support of System Structure and Parameterization (SSP) export (page 45)
- Improved display unit handling when sweeping parameters (page 41)
- License server improvements (page 40):
 - FLEXnet Publisher version update (Windows and Linux)
 - 64-bit license server now also on Windows
 - `lmadmin` removed
- New white paper “Using Windows Subsystem for Linux with Dymola” (page 49)

3.1.2 New and updated libraries

New libraries

There are no new libraries in this Dymola version.

Updated libraries

The following libraries have been updated:

- Aviation Systems Library, version 1.3.0
- Battery Library, version 2.4.0
- Brushless DC Drives Library, version 1.2.0
- ClaRa DCS Library, version 1.6.0
- ClaRa Grid Library, version 1.6.0
- ClaRa Plus Library, version 1.6.0
- Claytex Library, version 2022.1
- Claytex Fluid Library, version 2022.1
- Cooling Library, version 1.4.4
- Dassault Systemes Library, version 1.8.0
- Design, version 1.1.3
- Dymola Commands Library, version 1.13
- Dymola Models Library, version 1.5.0
- Electric Power Systems Library, version 1.6.0
- Electrified Powertrains Library (ETPL), version 1.5.0
- Fluid Dynamics Library, version 2.13.0
- Fluid Power Library, version 2022.1
- FTire Interface Library, version 1.1.3
- Human Comfort Library, version 2.13.0
- HVAC (Heating, Ventilation, and Air Conditioning) Library, version 2.13.0
- Hydrogen Library, version 1.3.6
- Model Management Library, version 1.3.1
- Multiflash Media Library, version 1.1.0
- Plot 3D Library, version 1.1.2
- Pneumatic Systems Library, version 1.5.0
- PowerTrain Library, version 2.7.0
- Testing Library, version 1.5.0
- Thermal Systems Library, version 1.9.0
- Thermal Systems Mobile AC Library, version 1.9.0
- Vehicle Interfaces Library, version 2.0.1
- VeSyMA (Vehicle Systems Modeling and Analysis) Library, version 2022.1
- VeSyMA - Engines Library, version 2022.1
- VeSyMA - Powertrain Library, version 2022.1
- VeSyMA - Suspensions Library, version 2022.1
- VeSyMA2ETPL Library, version 2022.1
- Visa2Base, version 1.12

- Visa2Paper, version 1.12
- Visa2Steam, version 1.12

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 Improved search of records

In Dymola 2023, you can search records using a new GUI. As an example do the following:

- Open `Modelica.Thermal.FluidHeatFlow.Examples.SimpleCooling`, and create an editable copy of it (by right-clicking it in the package browser and selecting **New > Duplicate Class...**, and finally clicking **OK** to confirm).
- Right-click the background and select **Parameters**.
- Right-click the **Medium** parameter value and select **Select Record...**

The following dialog appears:

To enter your search criteria's, double-click the criteria boxes and enter values, using the specified operators (“!=” means “not equal to”). If texts are searched for, a partial text match is done, that is, as an example, the string “dym” matches “Dymola”. All specified criteria must match (logical and).

A search example, with **Search** clicked afterwards:

Select Record

Select record for medium
Cooling medium

Search available records for variables matching a value.
- Operators <, <=, =, !=, >= or > followed by numeric value.
- Otherwise a partial text match is done.

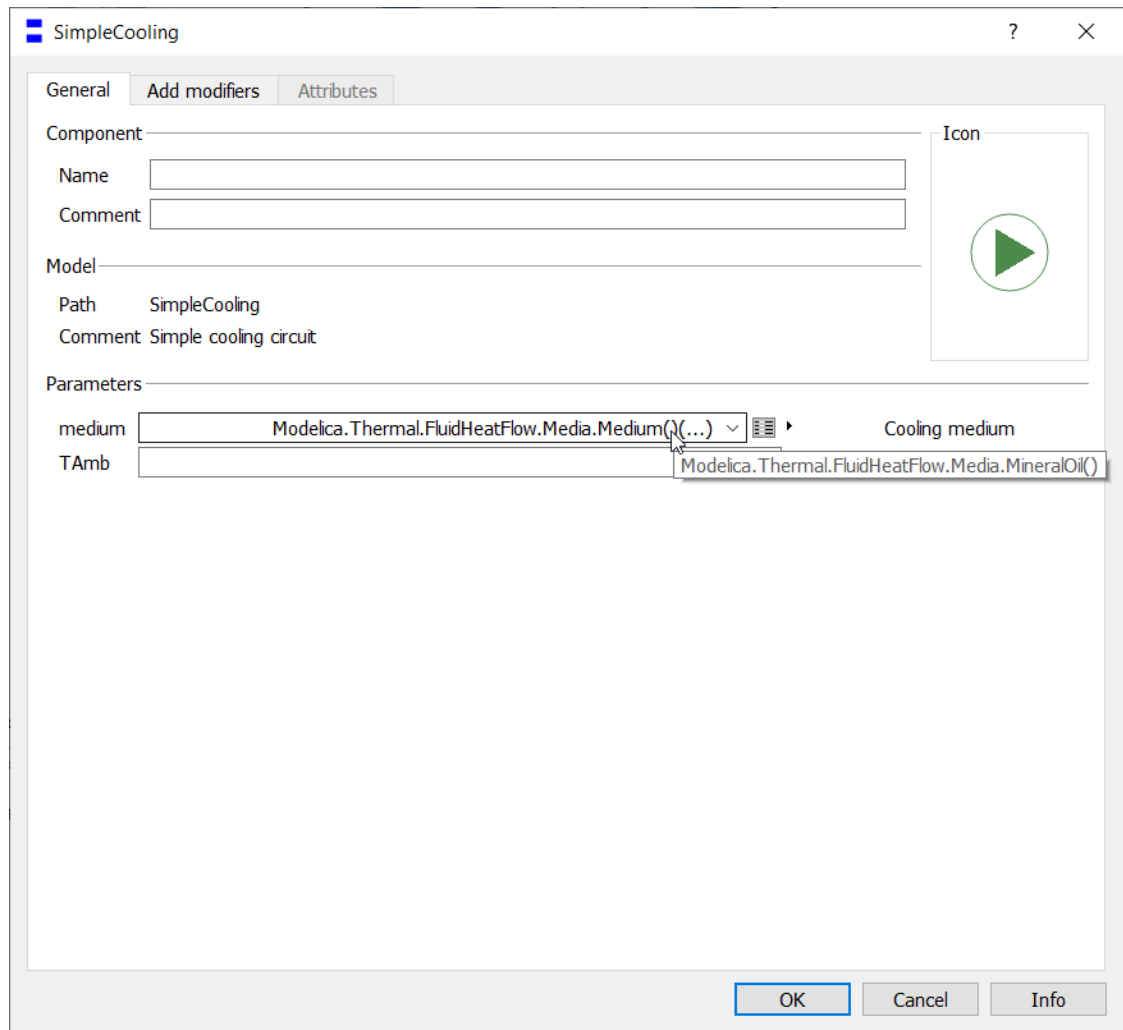
rho	cp	cv	lambda	nu
<1000		>=800		

Search

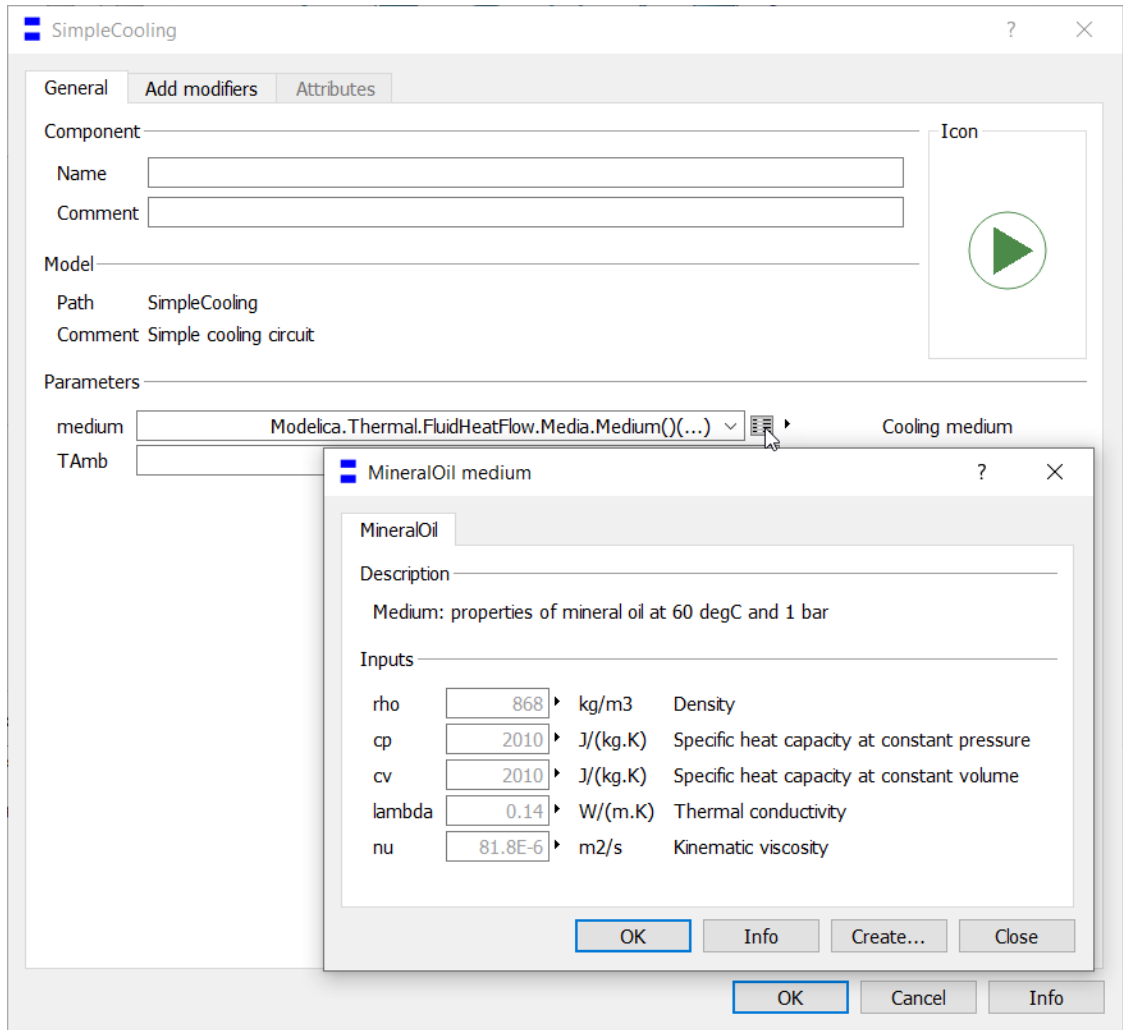
Name of record	rho	cp	cv	lambda	nu
Medium: properties of water at 10 degC and 1 bar	999.7	4192	4192	0.588	1.307E-6
Medium: properties of water at 30 degC and 1 bar	995.6	4177	4177	0.615	0.8E-6
Medium: properties of water at 90 degC and 1 bar	965.3	4205	4205	0.676	0.347E-6
Medium: properties of mineral oil at 60 degC and 1 bar	868	2010	2010	0.14	81.8E-6

OK Cancel

You can now select the wanted record and click **OK** (or double-click the wanted record) to select the record in the Parameter dialog. Assume that you select the last record; the result will be:

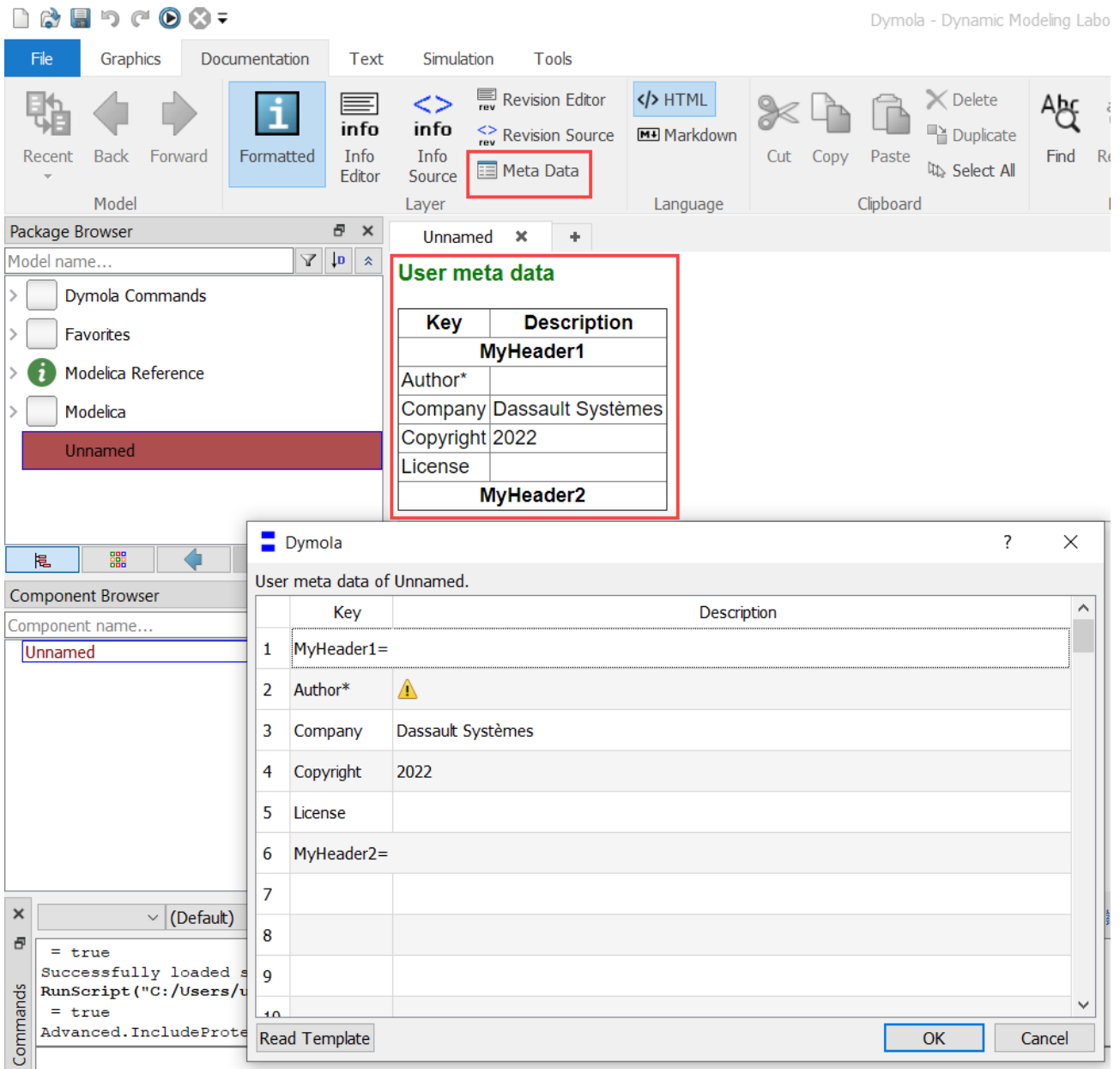


If you click the button after the medium value field, you can see the medium properties:



3.2.2 Model metadata support

Model metadata, in the form of key/value pairs, can be stored in models. The metadata is stored as an annotation in the model, and is automatically displayed in the documentation layer, if present.



There is a new dialog for viewing and editing model metadata. You can access the dialog by the command **Documentation > Meta Data** (highlighted button in previous image). The dialog is also seen in the image above.

Keys and descriptions are arbitrary text strings. Key that ends with * is regarded as mandatory (in the example above, *Author**). A warning is shown if the corresponding description field is empty. Keys that ends with = is creating a header (for such key you cannot add any description).

If the model is read-only, the metadata is displayed, but the **OK** and **Cancel** buttons are replaced by a **Close** button.

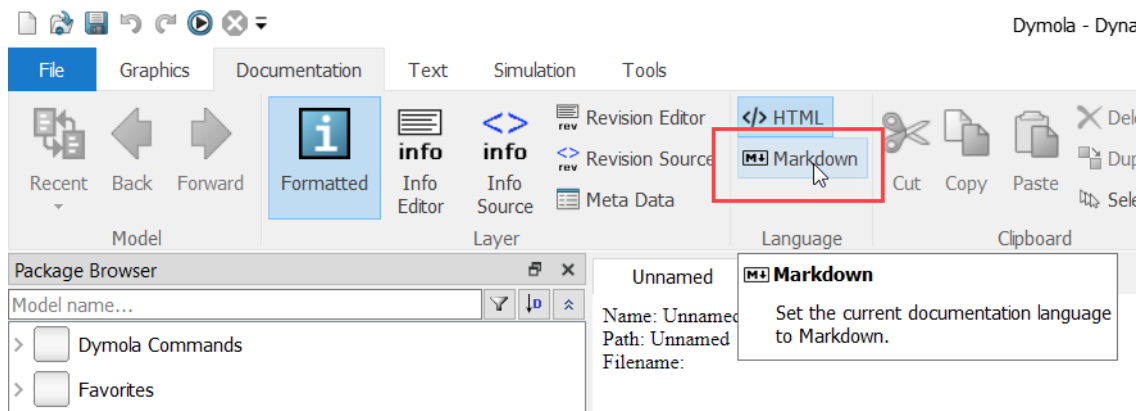
The **Read Template** button opens a text file with metadata keys, stored as one key and its description (separated by one or more TAB characters) per text line. A template for the above metadata would be:

```
MyHeader1=
Author*
Company           Dassault Systèmes
Copyright          2022
License
MyHeader2=
```

Note. The existing keys and descriptions are erased when reading a template file.

3.2.3 Support of Markdown for documentation

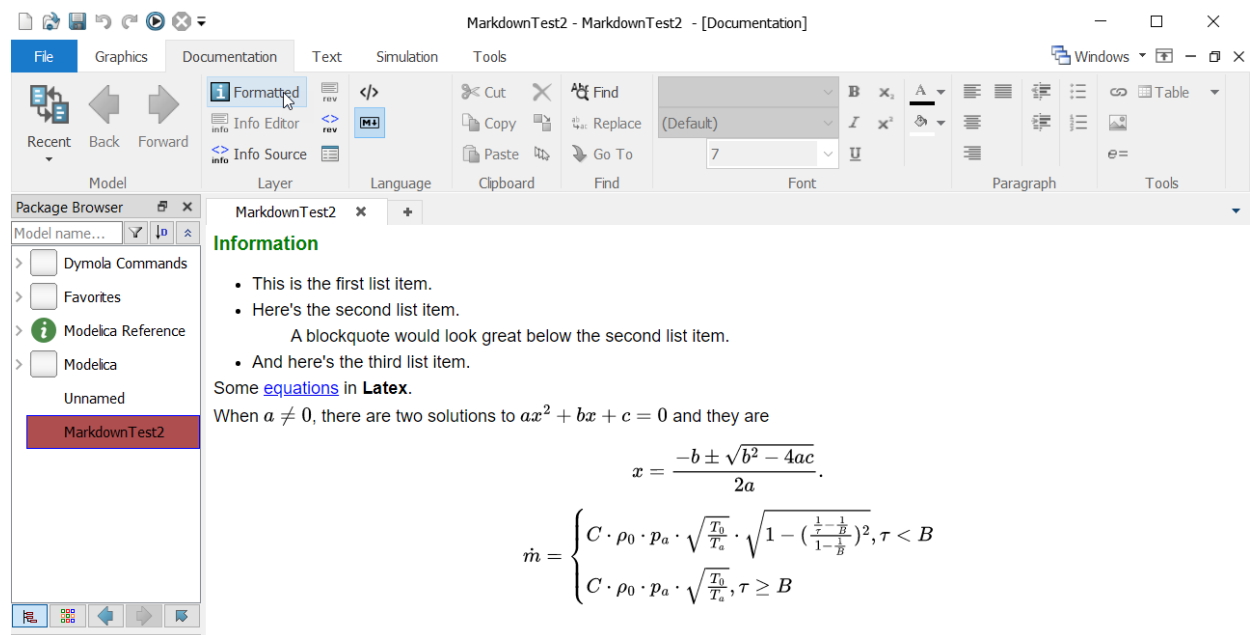
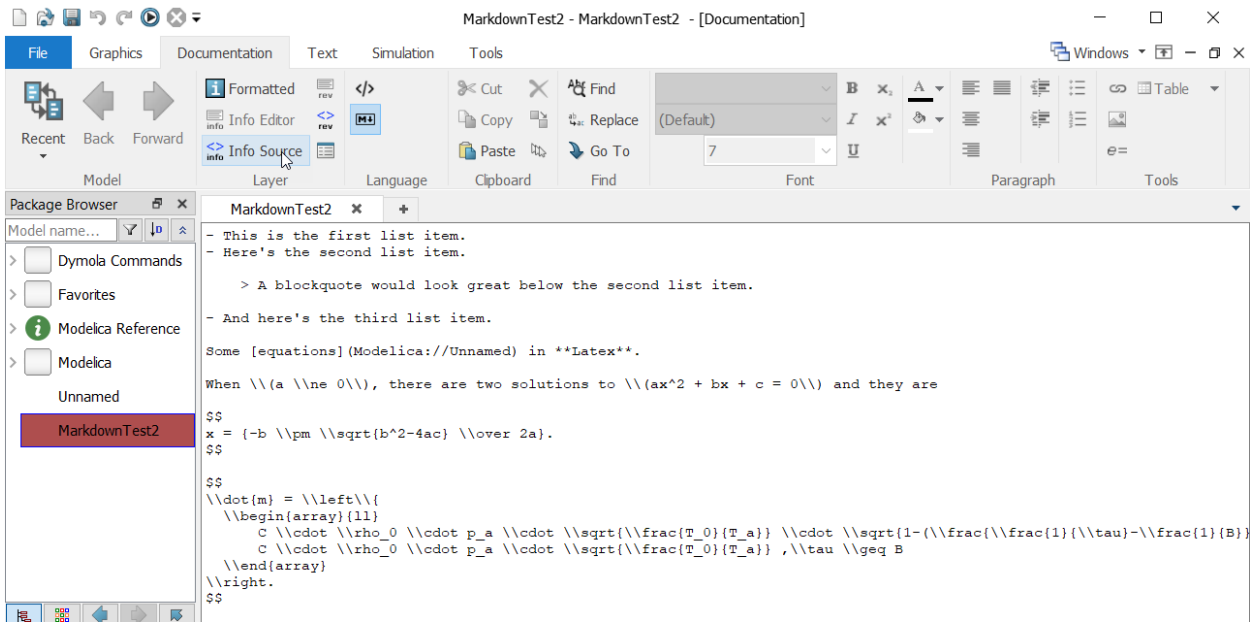
Markdown is supported for documentation, you can activate it by the command **Documentation > Markdown:**



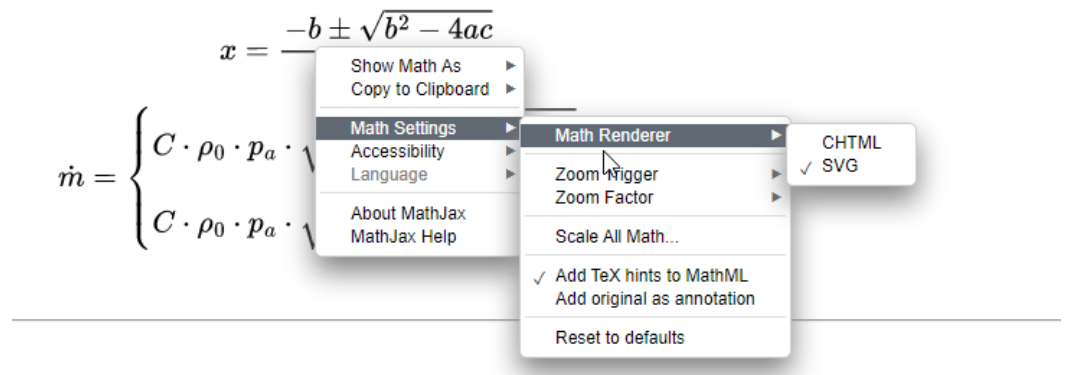
For more information about basic Markdown, see <https://www.markdownguide.org>.

Note that equations in Latex and MathML are supported. MathJax is used for rendering. An example of Latex text with equations:

The editor is shown in the first image below, the formatted text in the second image:



Note that there is an option to change the rendering by right-clicking any equation:



MathJax is by default activated by the flag `Advanced.Editor.MathJax=true`. You can set the flag to `false` to deactivate MathJax.

3.2.4 Minor improvements

Privacy of UML generation

The **Tools > UML** command uses an external service (yuml.me) to do the formatting and presentation of the UML diagram. This means that certain information from the user's model, such as model names, is passed to some external server on the Internet. While the use of this command is voluntary, there is a risk that user might select the command by mistake (especially so from the model's context menu).

To prevent this from happening, the user is now, by default, asked for confirmation when applying the UML command. This corresponds to the flag

```
Advanced.UI.AllowExternalServer = 1
```

By setting the flag to 0, the UML command is disabled, and removed from the menu. By setting the flag to 2, the command is allowed without asking for confirmation.

Improved handling of the amount of information in the error log

In previous versions of Dymola you could not decide the level of information displayed, an example is the very simple model:

```
model Unnamed
  Boolean test;
equation
  test = 1;
end Unnamed;
```


The error message was:

- ⓘ Check of [Unnamed](#):
- ✖ For equation
 - > test = 1;
found in class [Unnamed](#), declaration window at line 4.
- ⓘ Basic type inconsistencies detected.
- ⓘ Check aborted.
- ✖ ERRORS have been issued.

To understand the problem just looking at that text was not all clear, you had to expand it:

- ⓘ Check of [Unnamed](#):
- ✖ For equation
 - ∨ test = 1;
found in class [Unnamed](#), declaration window at line 4.
 - ✖ The types of the operands in (test) = (1)
are Boolean and Integer, but they must be compatible.
 - ⓘ Variable test was declared in class [Unnamed](#), declaration window at line 2.
 - ⓘ Basic type inconsistencies detected.
 - ⓘ Check aborted.
 - ✖ ERRORS have been issued.

For Dymola 2023, you can use the flag `Advanced.UI.AutoOpenLog` to decide to what level the messages in the error log are expanded:

- 0 (default) – No expansion, like in previous Dymola versions.
- 1 – First level of warnings and errors are expanded.
- 2 – All levels of warnings and errors are expanded.

Note that the last alternative corresponds to right-clicking the log text and selecting **Expand All**. However, this expansion is only kept until the next check, the flag value is kept during the session.

The previous header “For equation” has also been changed to “Type error for equation” to make the message more clear. So, setting the flag `Advanced.UI.AutoOpenLog = 1` and checking the above model in Dymola 2023 automatically gives:

- ⓘ Check of [Unnamed](#):
- ✖ Type error for equation
 - ∨ test = 1;
found in class [Unnamed](#), declaration window at line 4.
 - ✖ The types of the operands in (test) = (1)
are Boolean and Integer, but they must be compatible.
 - ⓘ Variable test was declared in class [Unnamed](#), declaration window at line 2.
 - ⓘ Basic type inconsistencies detected.
 - ⓘ Check aborted.
 - ✖ ERRORS have been issued.

Improved checking for overriding final modifiers

The check for overriding final modifiers has been improved to detect if a parameter declared as final in a record is overridden by setting the entire record (unless this gives the same value).

However, to be compatible with current libraries, the check is currently skipped for replaceable parameter records (that contain final parameters).

If you are developing libraries using such parameters, please contact Dassault Systèmes support at <https://www.3ds.com/support> for instructions.

3.3 Simulating a model

3.3.1 Better presentation of timers

General

In Dymola 2023, most block timers are now printed to `dsmodel.mof` if the following flags are set:

```
Advanced.GenerateBlockTimers = true;  
Advanced.OutputModelicaCode = true;
```

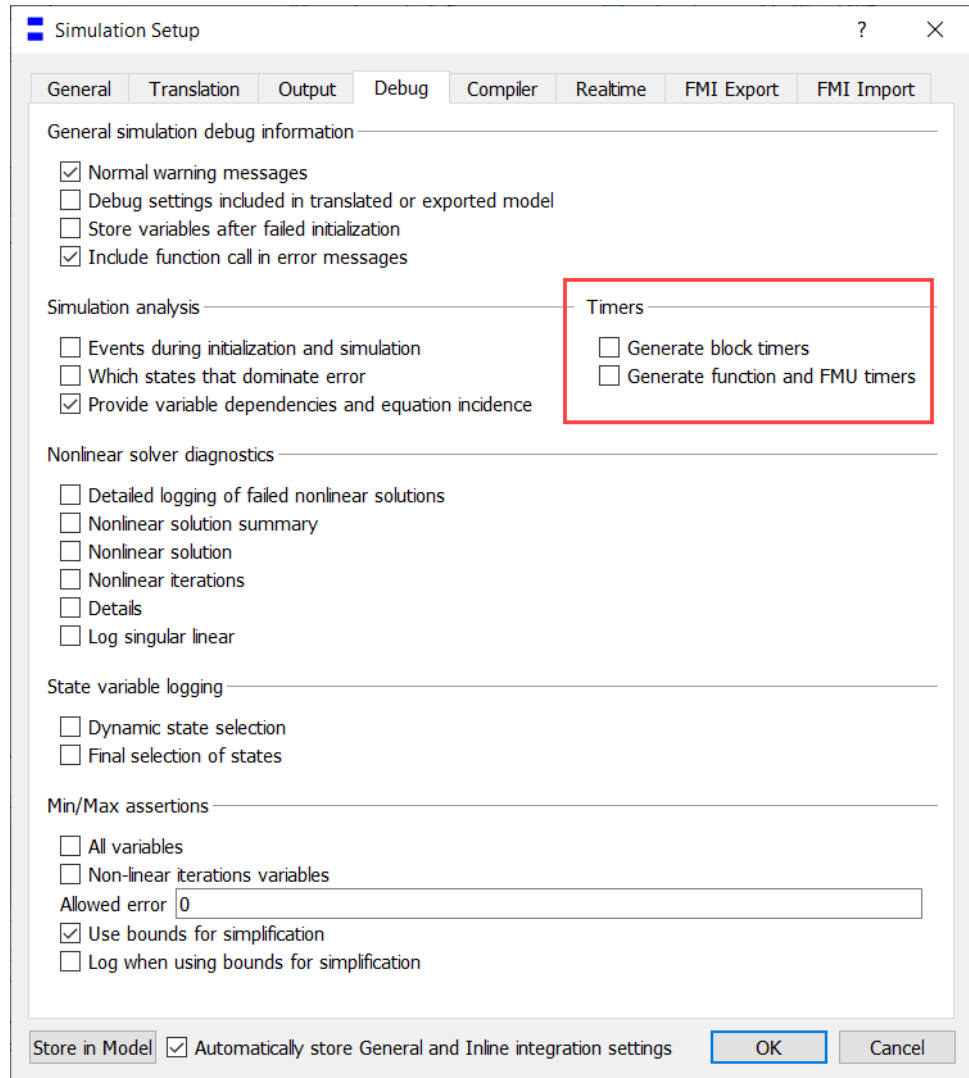
(By default, these flags are `false`.)

The block numbers match those in the simulation log.

When the new flag `Advanced.Debug.WriteTimerResultsToFile` is set to `true`, the results of any timers will be printed, as a semi-colon separated list, to the file `timers.txt`, in addition to the printout in the simulation log. The flag is by default `true`. Currently, the text file is updated once per second when simulating, if timers are generated.

The handling of precision timing is improved, precision timers are only used when any timers are generated (that is, when any of the flags `Advanced.GenerateBlockTimers`, `Advanced.GenerateFunctionTimers`, or `Advanced.GenerateTimers` are set to `true`). This means that if no timers are generated, the time reported in the normal simulation statistics in the simulation log does not use precision timing. Given this, the default value of the flag `Advanced.Define.PrecisionTiming` is now changed to `true`, since if no timers are generated, the precision timing is not used anyway. The advantage is that you don't have to care about the value of this flag anymore.

Simulation setup options for timers



The simulation setup dialog (reached by the command **Simulation > Setup**) now contains a new group **Timers**, with options to activate block timers, and function and FMU timers, respectively. The settings are by default not activated.

The options correspond to the flags `Advanced.GenerateBlockTimers` and `Advanced.GenerateFunctionTimers`.

Presentation of timers in the simulation analysis dialog

If timers are active (by any of the previously mentioned flags and settings), and the flag `Advanced.Debug.WriteTimerResultsToFile` is set to true, the timer values are

presented in the Simulation Analysis dialog, in the new tab **Timers**. The tab has two subtabs, **Block timers**, and **Function and FMU timers**. If you open the demo Coupled Clutches and simulate it, the dialog can look like:

Simulation timers read after the end of the simulation at time = 1.5 s.

Block timers Function and FMU timers

Name of block	Block number	Total CPU [s]	Mean [µs]	Min [µs]	Max [µs]	Number of calls
Entire model	0	0.007	1.52	0.15	284.12	4570
Event handling	1	0.000	16.72	2.58	283.92	29
Empty timer	2	0.000	0.01	0.00	0.16	4570
Outside of model	3	0.072	15.76	0.10	2368.38	4569
> Parameters	4	0.000	41.02	41.02	41.02	1
> InitialSection	10	0.000	173.45	173.45	173.45	1
> DynamicsSection	42	0.005	1.66	0.80	108.32	3048
> AcceptedSection1	71	0.000	0.09	0.04	16.58	1969
> AcceptedSection2	73	0.000	0.08	0.04	18.79	1969

Expand Highlighted Expand All Collapse All Refresh Close

The timer values are automatically read from the file `timers.txt` at the end of the simulation (regardless if the simulation succeeded or failed). For long simulations, you can use the **Refresh** button to read the timer values during the simulation.

The most significant contributions to the total time are highlighted in red. The more intense the color, the more time measured by the timer.

By clicking **Expand Highlighted**, all highlighted timers are expanded; this makes it easier to identify the most critical blocks and functions. (You can also **Expand All** or **Collapse All** timers.)

By right-clicking a timer and selecting **Expand All Levels**, the selected item and all its sublevels are expanded.

By clicking the headers, the data can be sorted according to the values in the selected column.

3.3.2 Steady state solver interface

Specifying a start time for dynamic steady-state finding

Steady State

Simulation mode

- Dynamic simulation
- Static steady-state finding
- Dynamic steady-state finding

Model manipulation

- Default steady-state initialization
- Remove "fixed = true" from continuous variables

Steady-state detection tolerance

- Default Tolerance: 1/s
- Strict
- Custom

Detection and verification

- Fail if steady state not achieved
- s

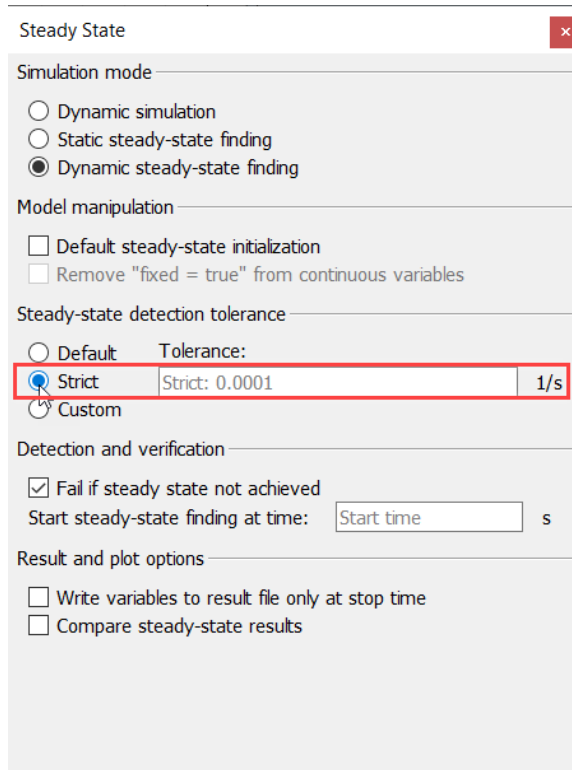
Result and plot options

- Write variables to result file only at stop time
- Compare steady-state results

The steady-state solver interface (reached by the command **Simulation > Steady State**) has been extended with a new option **Start steady-state finding at time:** for the **Dynamic steady-state finding** mode. This option sets a start time for the dynamic steady-state finding. That is, Dymola will not start to look for a steady state during simulation until the specified time is reached. This can be useful if a model starts in a steady-state, but an event triggers dynamic behavior at a time later than the start time of the simulation. By default, Dymola will start the steady-state detection from the beginning of the simulation. Changing this setting causes a re-translation. The setting corresponds to the flag `Advanced.Simulation.SteadyStateTerminationStartTime`.

Activating strict steady-state detection tolerance for dynamic steady-state finding

There is now an option to use a stricter tolerance for the mode **Dynamic steady-state finding**:



The `Strict` setting corresponds to the flag `Advanced.Simulation.SteadyStateTerminationStrict = true`. Changing to this stricter tolerance does not require re-translation. It also works with FMI export.

The new feature is the **Strict** option. However, to clarify that option, also the options already present (**Default** and **Custom**) must be clarified:

When performing a **Static steady-state finding**, the **Default** tolerance is $1e-8$. This value can be seen in the **Tolerance** field, preceded by the text `Default:`. During simulation, the value is scaled by state variables (the size of the states) and nominals.

When performing a **Dynamic steady-state finding**, the **Default** tolerance is $2e-2$ (or $1e-4$ if **Strict** is selected). The tolerance is then scaled according to the temporal scale of the problem (derived from the output interval and simulation start and stop time). This allows for quick enabling of **Dynamic steady-state finding** out-of-the-box. The result of this time scaling can be seen in the **Tolerance** field, preceded by the text `Default:` or `Strict:` to indicate if **Default** or **Strict** tolerance has been selected. During simulation, also the size of states and their nominals are scaling the value, for both selections.

For both **Static steady-state finding** and **Dynamic steady-state finding**, you can set a **Custom** tolerance. During simulation, also the size of states and their nominals are scaling the value, but the value is not rescaled in time. This allows for subsequent modification of stop time and number of output intervals without affecting the **Dynamic steady-state finding**.

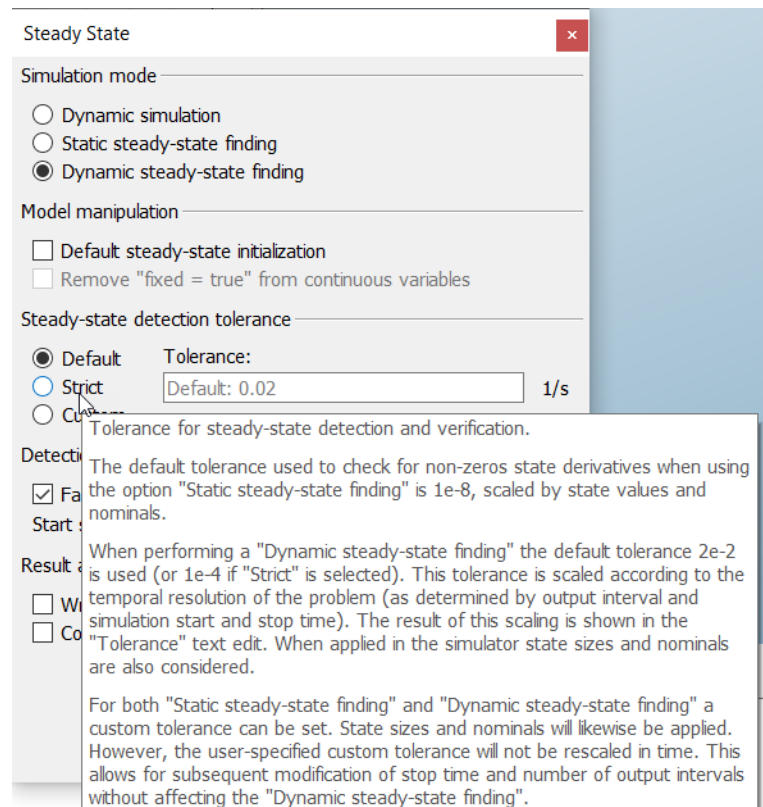
For example, it allows for setting a large stop time to minimize the risk of hitting the stop time before a steady state is reached.

The custom tolerance can be seen and changed in the **Tolerance** field, with no preceding text. The value corresponds to the value of the flag `Advanced.Simulation.SteadyStateTerminationTolerance`. If the **Custom** mode is not selected, which is default, the value of the flag is 0.

Notes:

- Using and changing the custom tolerance requires re-translation.
- Custom tolerances also work with FMI export.
- If you click the **Tolerance** field, and start entering a value, you automatically change the tolerance mode to **Custom**.

Please note the extended tooltip for the **Steady-state detection tolerance** group:



Steady-state solver interface improvements

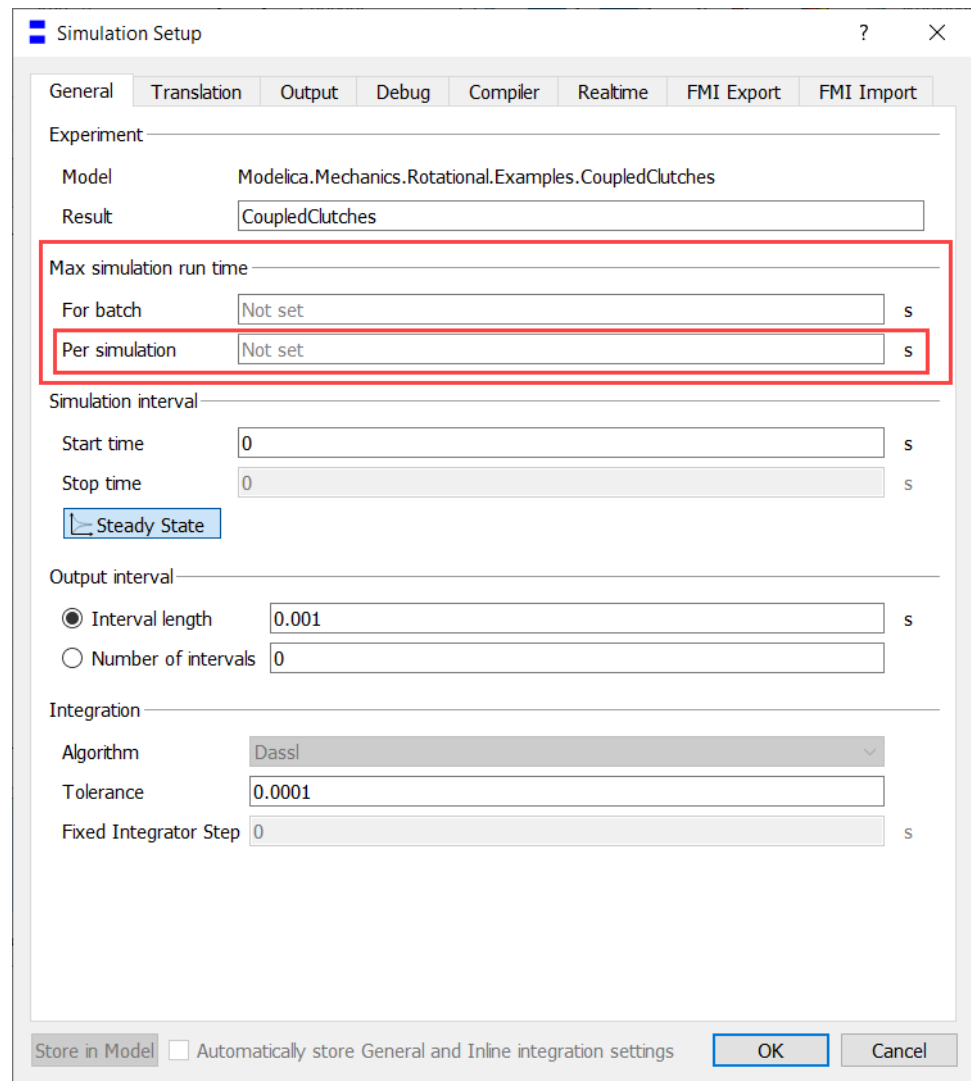
The steady-state solver interface has been simplified and clarified.

The options **Stop simulation at time:** and **Number of results to keep** have been removed from the steady-state solver interface. (They are still available in the Simulation Setup and, for the former option, also in the **Simulation** ribbon.)

The option **Steady-state detection tolerance** has been extended to a group with the same name. The group clarifies previous options, and includes the new **Strict** option. See previous section for details.

3.3.3 Max run time per simulation

There is now a new option to specify max run time for individual simulations. That is, each simulation gets its own built-in timer. In both cases, wall-clock time is used.



The option **For batch** existed already in previous versions of Dymola. A single timer is used, which is managed by Dymola. For a batch of runs, e.g. sweep parameters, there is one timer and all simulations will be stopped when that timer runs out. Changing this timer does not require re-translation.

The **Per simulation** option is new for Dymola 2023. Each simulation gets its own timer. Each simulation starts its own timer with the specified time when it starts the simulation and stops itself if this timer runs out. This option may be useful if a few of the simulations in a sweep takes very long time and eventually fails. Then these can be automatically stopped early, without affecting the other simulations. (Please compare with the **For batch** alternative, where an early simulation taking a long time may consume all available time for the batch, leading to any remaining simulations not being executed.)

The option **Per simulation** is embedded in the simulator, including an exported FMU, and is therefore independent of Dymola running the simulation. Changing this option requires re-translation.

The logging of both options has also been improved. See the following excerpts from the simulation log:

```
... Error message from dymosim
Stopping simulation after reaching max run time.
(Stop requested by Dymola.)
```

```
... Error message from dymosim
Stopping simulation after reaching the embedded
max run time (4 seconds) for this simulator.
```

3.3.4 Scripting

Scripting support for creating plot

The built-in function `createPlot` now has an additional argument `plotInAll`. This argument is already available in the built-in function `plot`. If you set this argument to `true`, all open result files will be searched for the variables specified to be plot, and all found occurrences are plotted. This is useful if you want to compare the result of different simulations.

Built-in function `clearFlags` improved

The built-in function `clearFlags()` has been improved; it does not reset the flag `Advanced.SaveSettings`. This prevents possible problems when starting Dymola with the command line option `-nosavesettings` and then using `clearFlags()`.

Improved saving of models by scripting

The built-in function `saveModel` for saving a model by scripting has been improved.

You can now store in the current model (like “**Ctrl+S**”) by setting the argument `path` to an empty string `""`. (The argument has no default; you have to provide a path.)

A new integer input argument `storeVariant` has been added, the possible values are:

- 0: save the model as one `.mo` file (default value)
- 1: save the model in directories
- 2: keep current structure of the model when saving

Scripting support for importing SSV files

You can import SSV files by the new built-in function `importSSV`. For details, see section “Importing SSV files by scripting” on page 46.

3.3.5 Minor improvements

Improved handling of the “smooth” operator for some solvers

The handling of the “smooth” operator has been improved for a number of solvers. This may reduce the number of events and improve performance for models that use “smooth”.

The solvers improved are the “Godess” solvers, that is, the solvers:

- Radau IIA
- Esdirk23a
- Esdirk34a
- Esdirk45a
- Dopri45
- Dopri853
- Sdirk34hw
- CerK23
- CerK34
- CerK45

Note that other solvers already have the above improvement.

Nonlinear solver statistics available for plotting

In previous versions, a setting was available to include CPU time and simulation events in the simulation results. The setting has now been renamed to **Include variables for CPU time, event counting, and nonlinear solver statistics**, to reflect both the old feature of including number of events, as well as including the new nonlinear system diagnosis.

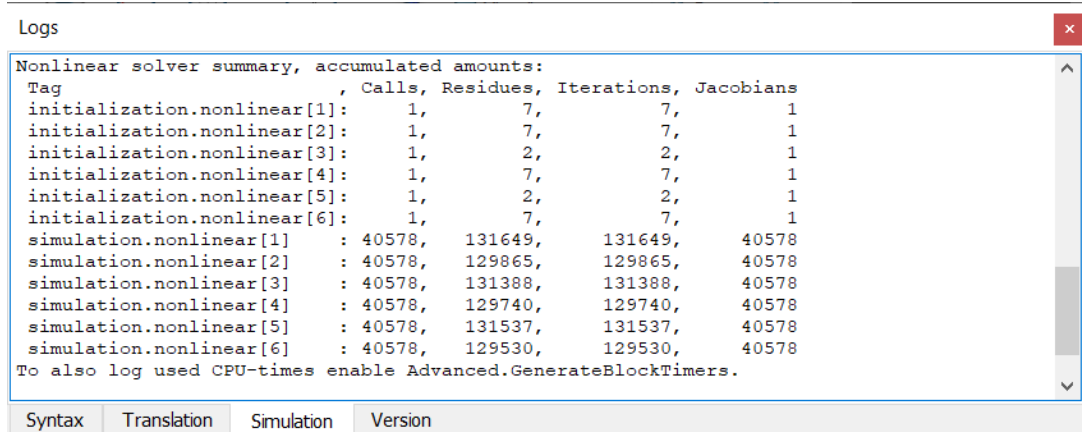
The setting is available in the simulation setup, reached by the command **Simulation > Setup**, the **Translation** tab. The setting corresponds to the flag `Advanced.Translation.OutputCPUtime = true`. (The value is by default `false`.)

When activated, except including CPU time and event counting, four new variables are created for each nonlinear system in the model. These variables can be found in the variable browser (see last image in this section)

These variables contain exactly the same diagnostics as is given in the simulation log by setting **Nonlinear solution summary** in the **Debug** tab of the simulation setup, but now it is possible to plot the diagnostics over time to see where the most effort is spent.

For the example Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6:

The nonlinear solution summary in the log:

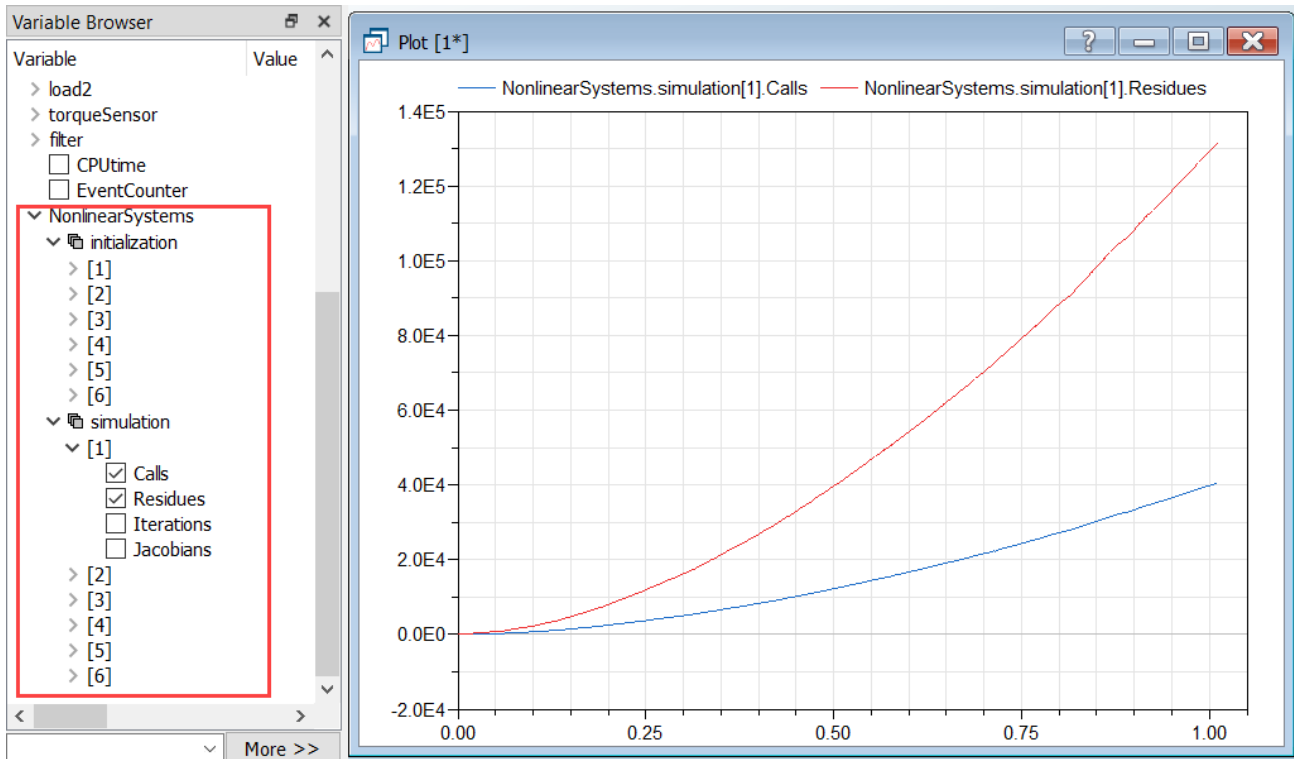


The screenshot shows a log window titled "Logs" with a close button in the top right corner. The log content is as follows:

```
Nonlinear solver summary, accumulated amounts:
Tag           , Calls, Residues, Iterations, Jacobians
initialization.nonlinear[1]: 1, 7, 7, 1
initialization.nonlinear[2]: 1, 7, 7, 1
initialization.nonlinear[3]: 1, 2, 2, 1
initialization.nonlinear[4]: 1, 7, 7, 1
initialization.nonlinear[5]: 1, 2, 2, 1
initialization.nonlinear[6]: 1, 7, 7, 1
simulation.nonlinear[1] : 40578, 131649, 131649, 40578
simulation.nonlinear[2] : 40578, 129865, 129865, 40578
simulation.nonlinear[3] : 40578, 131388, 131388, 40578
simulation.nonlinear[4] : 40578, 129740, 129740, 40578
simulation.nonlinear[5] : 40578, 131537, 131537, 40578
simulation.nonlinear[6] : 40578, 129530, 129530, 40578
To also log used CPU-times enable Advanced.GenerateBlockTimers.
```

At the bottom of the log window, there are four tabs: "Syntax", "Translation", "Simulation", and "Version".

The corresponding variables in the variable browser, with two signals plotted:



We can see from the plot that the equation is called more often as the simulation proceeds; the number of residual calculations seems to be explained by the increased number of calls to the equation.

Improved handling of start guesses for variables that describe the discrete state of mixed systems of equations

Save Start Values in Model

Source for start values

- Current Variable Browser content
- Initialize the model and save the results

Store options

- Store values in current model
- Store values in new model

Name:
MyCoupledClutches

Description:
Drive train with 3 dynamically coupled clutches

Extends:
MyCoupledClutches

Insert in package:

Open new class in:
This tab

Advanced options for storing start guesses

- Save changes in parameters and in initial values of states
- Overwrite parametrized start attributes for below selection
- Additionally, save changes in the start attributes of:
 - Iteration and mixed system variables
 - Iteration, torn, and mixed system variables
 - Outputs, auxiliary variables, and states

i Only save start guesses for additional variables at start time. Other usage may cause unwanted changes in the model parametrization. These advanced options are only intended for saving start guesses and must not be used to continue simulations from times later than the start time.

Advanced << OK Cancel

Already in Dymola 2022x you could select to store start guesses for variables that describe the discrete state of mixed systems of equations. In the above framed selection, the added text “mixed system” indicates this. This improvement makes it possible to store in which discrete configuration a mixed system is in after initialization.

Note that you can, from Dymola 2022x, also edit these start guesses in the variable browser.

Improved start-value priority

The handling of start value priority has been improved; the start-value priority has been updated to also consider the parameter-level if the start-value is bounded to a parameter. (This

change can be disabled by setting the new flag `Advanced.Translation.UseParameterLevelForStartValue = 0`. The default value of the flag is 2, meaning that parameter-level is considered everywhere. You can also set the flag to 1, that means only use parameter-level for non-iteration, trusting literals more.)

You can now also get an overview of the prioritized conflicting start-values setting the new flag:

```
Advanced.Translation.LogStartValuePriority = true
```

(The flag is by default `false`.)

Time range filter for Simulation Analysis Numeric Integration

There may be different states dominating the error during different times of the simulation. It is now possible to limit the time range of the integrator diagnostics data. This makes it possible to see, in the summary, what is relevant for the chosen time interval. For the corresponding plots, the benefits are that it is faster to zoom in at the interesting range and that downsampling of the plot data may be avoided. This reduces the risk of missing something important.

The numeric integration is reached by the command **Simulation > Simulation Analysis**, the **Numeric Integration** tab:

Simulation statistics read after the end of the simulation at time = 1.5 s.

Variable	Limits Step Size	Dominates Error	>10 % of Error
clutch1.ph ⁱ _rel	0	3	9
clutch1.w_rel	3	95	98
clutch2.ph ⁱ _rel	0	0	0
clutch3.ph ⁱ _rel	0	3	29
clutch3.w_rel	0	33	46
J1.ph ⁱ	0	0	10
J1.w	1	9	143

Start time: s

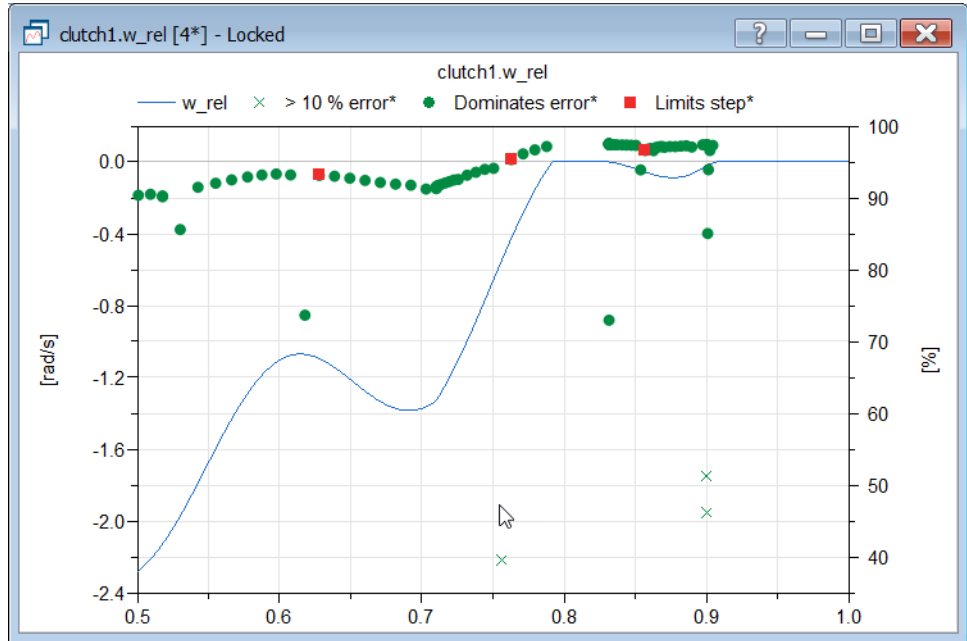
Stop time: s

You can use the sliders to set the start time and stop time of the range, or enter the values directly in the value fields. Click **Apply** to use the selected range.

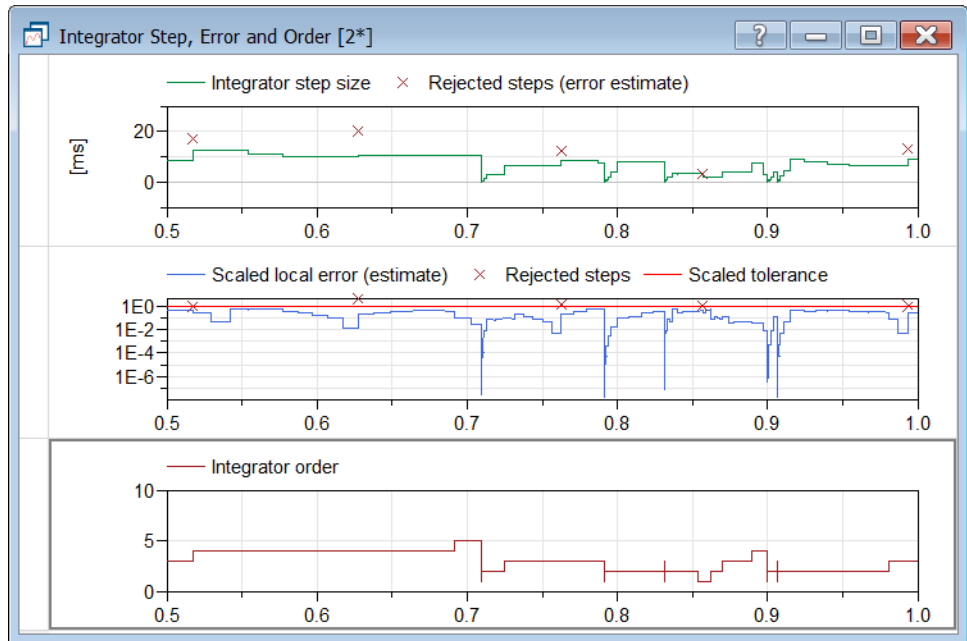
The values presented in the summary will be limited to the data gathered in the specified time range. Step size and error plots will also be limited to the specified range.

Note that the limits are specified inside Dymola and doesn't limit the output from the integrator.

Example of an error plot from clutch1.w_rel for the above range:

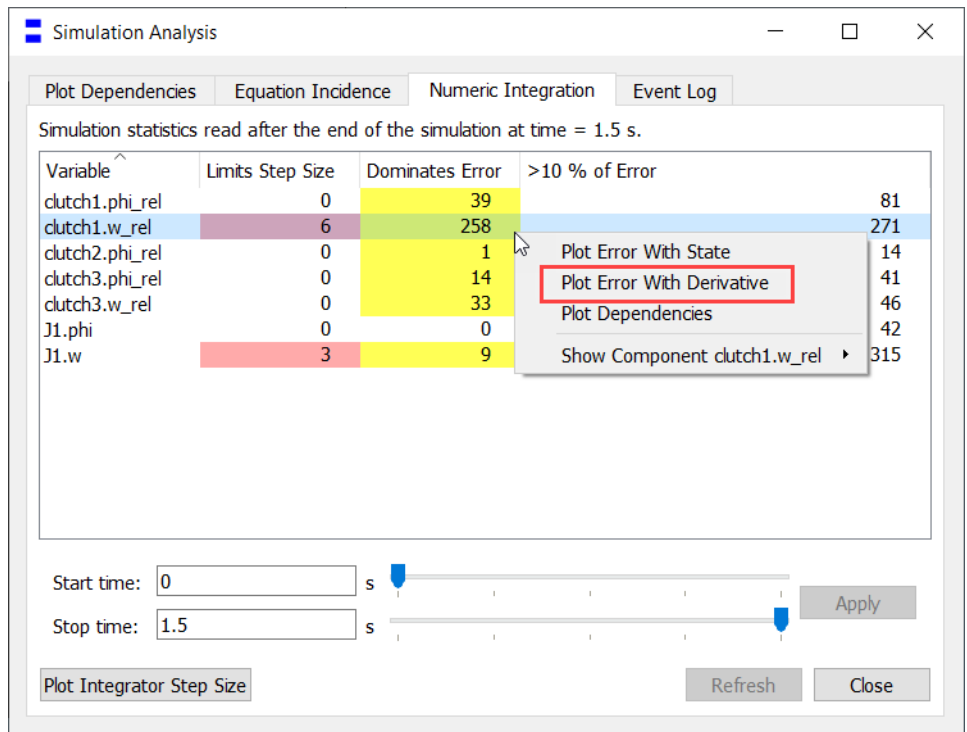


Example for the integrator step size for the above range:



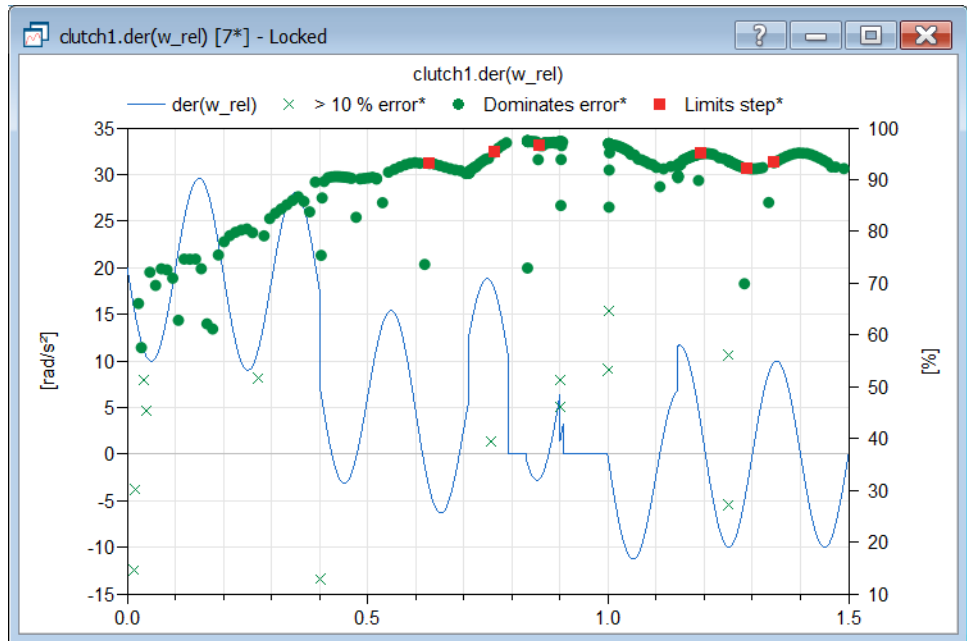
Plotting error with the state derivative instead of the state itself for Simulation Analysis Numeric Integration

For numeric integration of simulation analysis, you can now plot error with the state derivative instead for the state itself, by the context command **Plot Error with Derivative**:



(The numeric integration is reached by the command **Simulation > Simulation Analysis**, the **Numeric Integration** tab.)

The result of the command is:



Improved handling of alias names of variables in error messages and other outputs

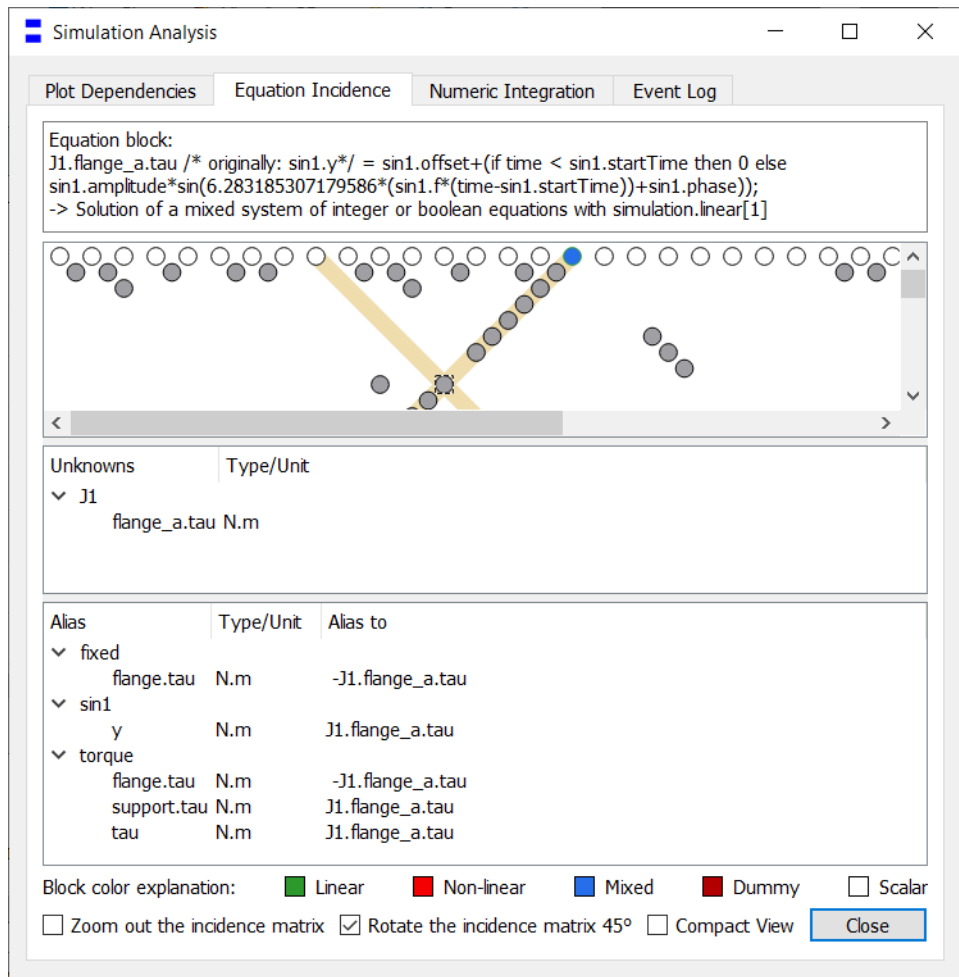
In Dymola 2023, the use of alias names of variables is improved. You can use the new flag `Advanced.Translation.KeepAliasInMessages` to decide what to present.

The value of the flag can be:

- 0: Use alias name in texts; this value corresponds to the alias name use in previous Dymola versions.
- 1: Use original variable name in texts; this is the default value of the new flag.
- 2: Use alias name in texts, but also present the original variable name as comment in the texts.
- 3: As “2”, but use this also in `dsmodel.mof`.

The flag is used when creating error messages, information messages, logging, plot dependency information, and equation incidence texts.

As an example, consider equation incidence with the flag value 2:

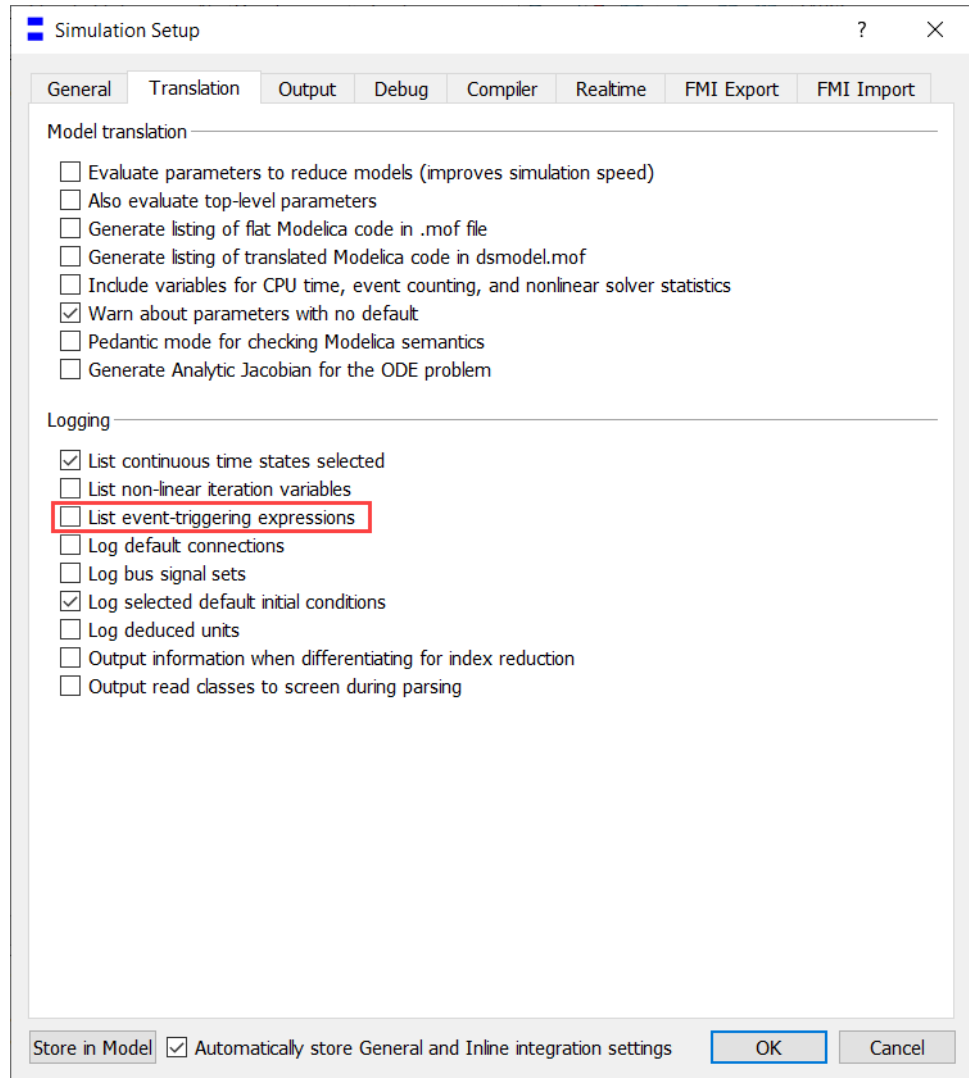


In the equation block (top pane), the alias name `J1.flange_a.tau` is used, and the original value `sin1.y` is presented in a comment.

The fourth pane is new; here the aliases are presented, for the selected block.

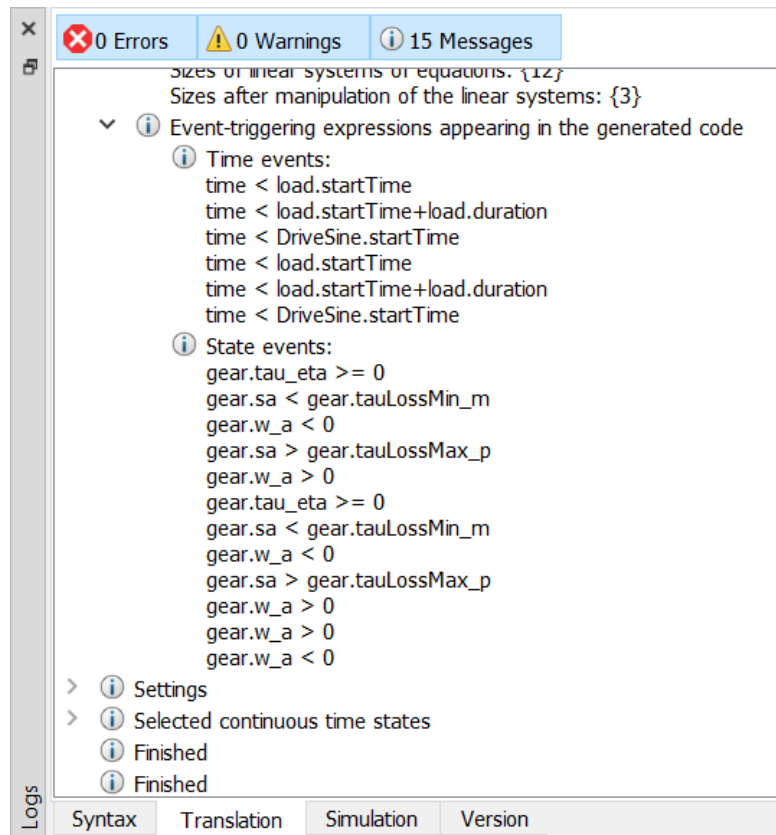
Listing possible event-triggering expressions

If you activate the new setting **List event-triggering expressions** in the **Translation** tab of the simulation setup (reached by the command **Simulation > Setup**), then all expressions that may trigger events are listed under Translation statistics in the translation log.



The setting is by default not activated. This corresponds to the flag `Advanced.Translation.Log.EventExpressions = false`.

An example of the translation log from simulating `Modelica.Mechanics.Rotational.Examples.LossyGearDemo1` with the above setting activated:

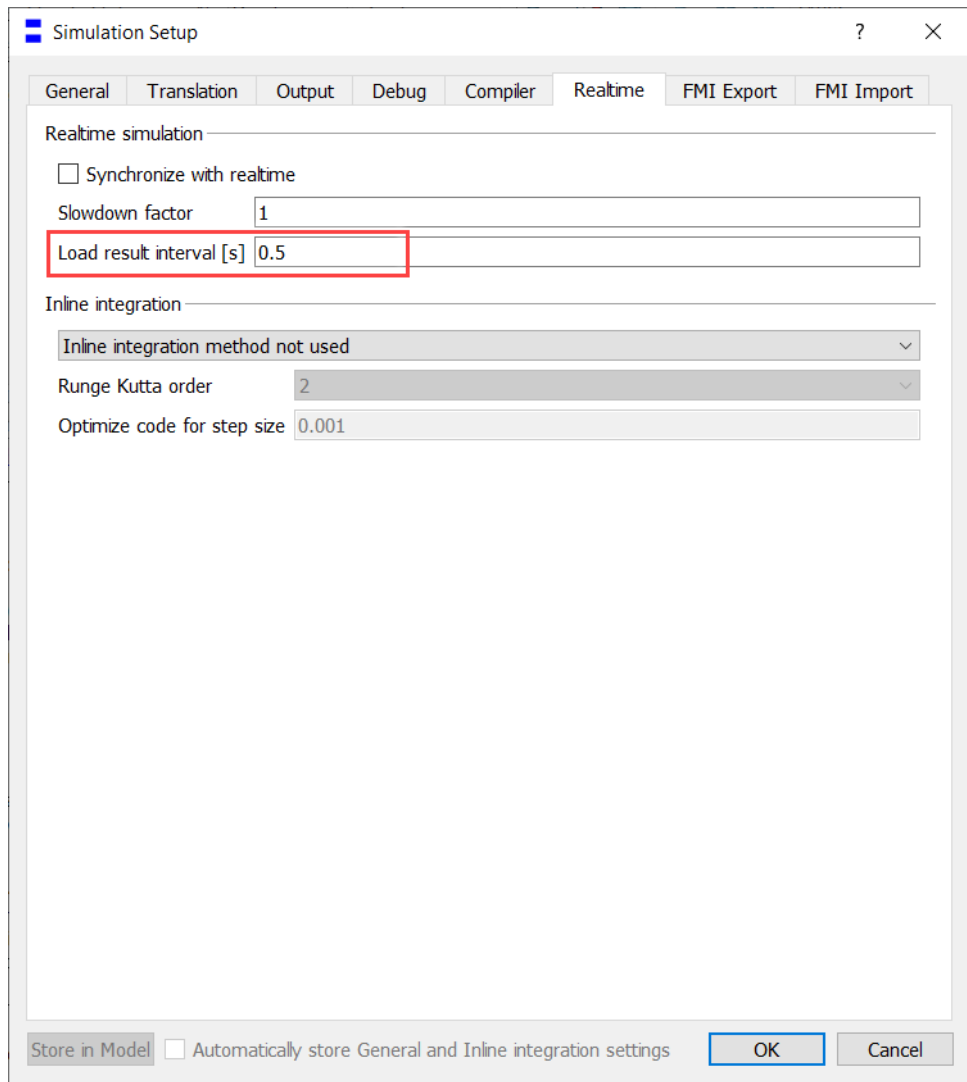


Note that the list in the translation statistics do not include input time events and dynamic state selection events.

Note also that these expressions only potentially may trigger events. It may be that only a subset actually triggers events during simulation. To check which event-triggering expressions actually *did* cause events during a particular simulation, consider the **Event Log** tab in the simulation analysis dialog.

Improved synchronizing of writing and reading the simulation result file

In previous Dymola versions, the frequency of online reading of results from the simulation result file (for example, to update plots) was specified by the setting **Load result interval** in the **Realtime** tab of the simulation setup (reached by the command **Simulation > Setup**).



In Dymola 2023, this is still the case, but now also the writing of values to the simulation result file is specified by this setting. This makes the simulation of some models less time-consuming.

Note that the default value of 0.5 seconds is suitable in most cases. If you set a lower time, this may slow down the speed of the simulation.

Use of inline integration displayed in simulation log

In Dymola 2023 the simulation log includes information about if an inline solver is used, and in that case, which inline solver. An example:

```
Logs
Log-file of program ./dymosim
(generated: Tue Jan 25 12:51:32 2022)

dymosim started
... "Modelica.Mechanics.Rotational.Examples.CoupledClutches" simulating
... "dsin.txt" loading (dymosim input file)
... "CoupledClutches.mat" creating (simulation result file)

Integration started at T = 0 using INLINE integration method
Implicit Runge Kutta of order 2, optimized for step size 0.001

Integration terminated successfully at T = 1.2
CPU-time for integration      : 0.033 seconds
CPU-time for one grid interval : 0.066 milliseconds
CPU-time for initialization   : 0.001 seconds
Number of result points      : 512

Syntax Translation Simulation Version
```

3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 50.

3.4.1 Installation on Windows

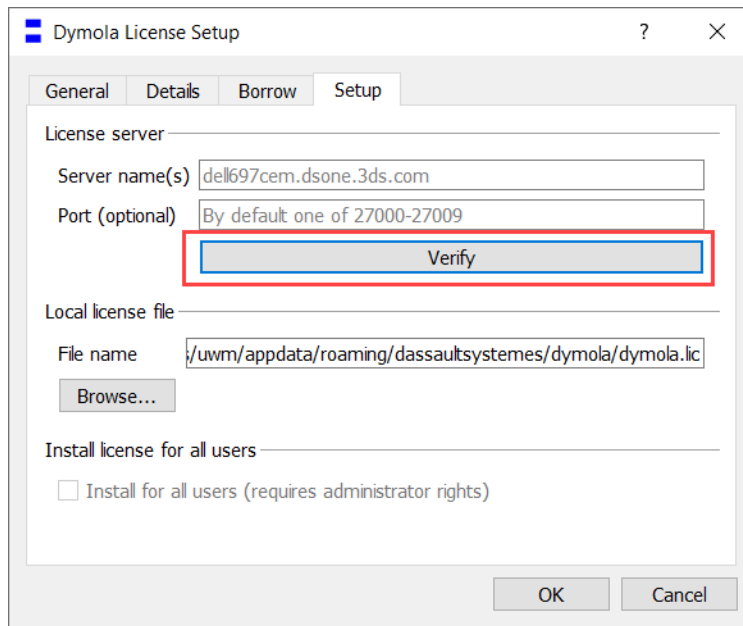
Checking selected license server in the license setup

To be able to check the selected license server in the license setup, there is now a new button **Verify** that performs the same check as is done when you select and apply a new license server.

This makes it possible to:

- Checking the present license server.
- Checking a selected license server before applying it.

Note that also DSLS license servers can be checked.



3.4.2 Installation on Linux

Checking selected license server in the license setup

See the corresponding section in “Installation on Windows”.

3.4.3 Dymola license server on Windows and Linux

Upgrade of FLEXnet Publisher version (on Windows and Linux)

FLEXnet Publisher in Dymola 2023 is upgraded to version 11.16.2.1. Therefore, the FLEXnet Publisher license server used for Dymola must also be upgraded to a version number at least 11.16.2.1, to be compatible. To do this, do the following:

- Copy the new vendor daemon (dynasim.exe) to the license server.
- Update lmgrd.exe to version 11.16.2.1.
- Restart the license server.

The updated FLEXnet Publisher enables you to work on for example (for Windows) Windows Server 2019.

Note that FLEXnet Publisher license server 11.16.2.1 works with previous Dymola versions as well.

64-bit license server (on Windows and Linux)

The 32-bit license server on Windows is replaced by a 64-bit license server. The primary reason for this is to be consistent with Linux, where newer versions will only be provided as 64-bit.

The web-based license server lmadmin removed (on Windows and Linux)

On Windows and Linux, the web-based FLEXnet license server manager `lmadmin` is removed from the Dymola distribution. You can still use the command line based `lmgrd`. Note that you can yourself download `lmadmin` if you want to.

3.5 Model Experimentation

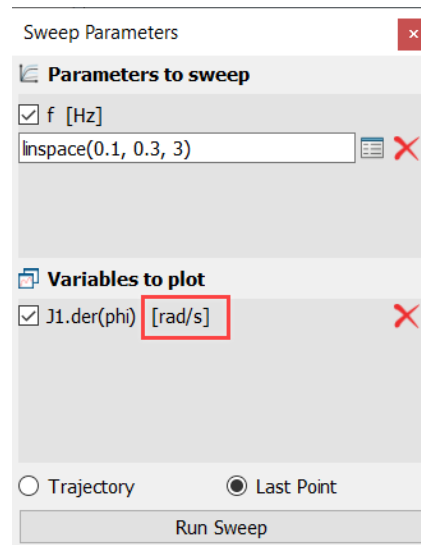
3.5.1 Improvements for sweeping parameters

Improved display unit handling for sweeping

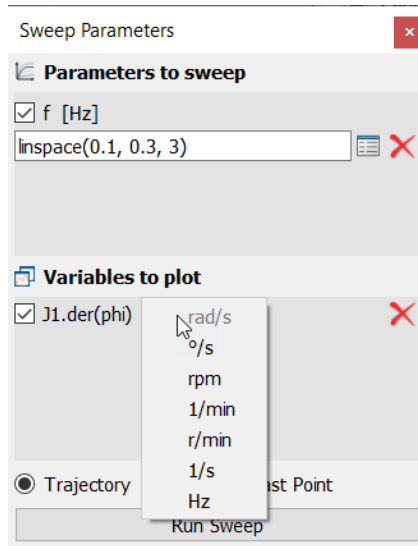
The following improvements for handling of display units for sweeping have been implemented:

Display and change option for display unit for plotted variables in the sweeping dialog

The display unit is now displayed also for plotted variables:

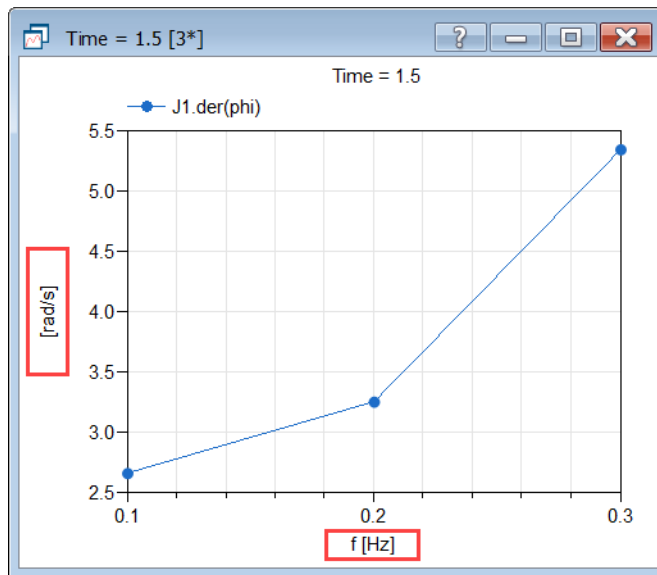


You can change the display unit for a plotted variable by clicking on it, like for parameters to sweep:



Selection of display units respected and displayed in the sweeping plots

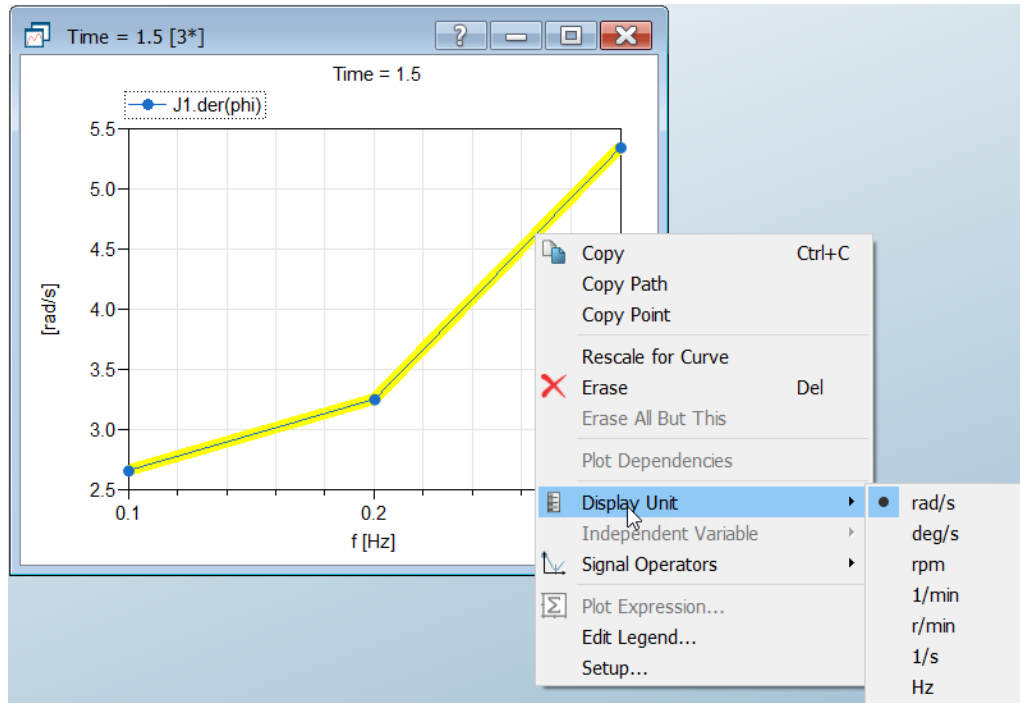
The choices of display units of parameters to sweep and variables to plot are respected and displayed in the resulting plot. As an example, for sweeping one parameter, last point:



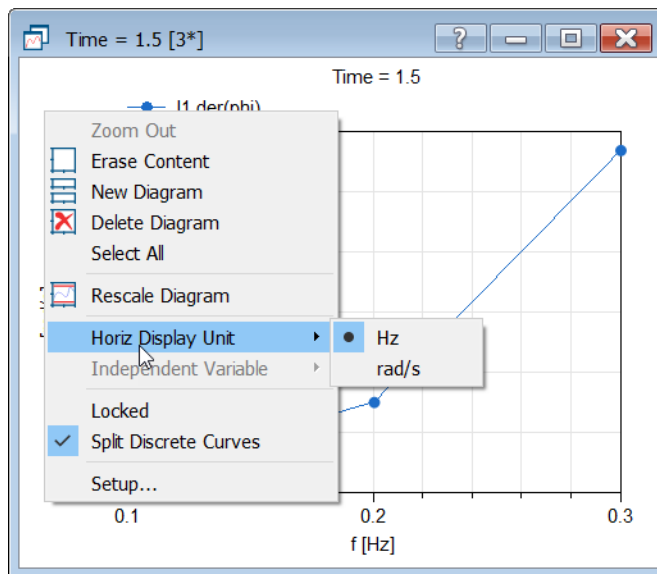
For sweeping one parameter, options to change the display units after the sweep

When you sweep one parameter, you can change the display unit of the plot variables after the sweep, by either right-clicking the corresponding curve and selecting **Display Unit**, or

selecting the curve and using the Display Unit button of the Plot ribbon. An example of the former:



To change the display unit of the swept parameter, you can right-click in an empty space in the plot window and select **Horiz Display Unit**:



Note that changing display units after the sweep is not possible when sweeping more than one parameter.

3.6 Other Simulation Environments

3.6.1 Dymola – Matlab interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2017a (ver. 9.2) up to R2021b (ver. 9.11). 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.

3.6.2 Real-time simulation

Compatibility – dSPACE

Dymola 2023 officially supports the DS1005, DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2023 generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases:

- dSPACE Release 2017-A with Matlab R2017a
- dSPACE Release 2017-B with Matlab R2017b
- dSPACE Release 2018-A with Matlab R2018a
- dSPACE Release 2018-B with Matlab R2018b
- 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 R2020b and R2021a
- dSPACE Release 2021-B with Matlab R2020b, R2021a, and R2021b

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`, that replaces `dym_rti_build` for building dSPACE applications from models containing DymolaBlocks. The new function uses the new dSPACE RTI function `rti_build2` instead of the old function `rti_build`.

A corresponding new multi-processor build function, `dym_rtmp_build2`, is also introduced.

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

Note on `dym_rti_build` and dSPACE Release 2017-A and later

The function `rti_usrtrcmmerge` is no longer available in dSPACE Release 2017-A and later. As a consequence, 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 R2017a (Simulink Real-Time ver. 6.6) to R2021b (Simulink Real-Time ver. 7.2). Only Microsoft Visual C compilers have been tested.

3.6.3 Dymosim DLL

Postponed discontinuation of dymosim DLL support

Dymosim DLL is still supported in Dymola 2023. However, Dymola 2023 is the last planned Dymola version to support dymosim DLL. From the next Dymola version, Dymola 2023x, dymosim DLL is not supported. It is replaced by the use of FMI.

3.6.4 SSP support

SSP export

Dymola can export Modelica models according to the System Structure and Parametrization (SSP) file format. The details of SSP can be found at <https://ssp-standard.org>. Dymola performs the following steps when exporting to SSP:

- The top-level model structure is stored as a System Structure Description (SSD). That structure includes top-level components and connectors, and any connections between these.
- Components are stored in two formats depending on implementation.
 - If the component is an imported FMU, the FMU itself is copied to the resources directory of the SSP. The SSD will contain a reference to that FMU copy.
 - Otherwise, it is assumed that the component is represented by a Modelica model. In that case, the full path of the component's model is stored in the SSD. The Modelica model itself is not stored in the SSP file.

- Model documentation, user metadata annotation, parameter types, and component modifiers (including nested structures) are stored in the SSD.

SSP export is available by the command **File > Save > Export SSP...** or as the built-in function `exportSSP`.

Importing SSV files

The System Structure Values (SSV) file format is really a part of SSP, but it has gained interest as a freestanding entity. You can store model parameterizations in SSV files, and apply these parameterizations on Modelica files and FMUs.

The idea is:

- Start out with a Modelica model as your base model.
- Open an SSV file to read parameter data.
- Create a new Modelica model from the base model, applying the parameters (with values) as modifiers in an `extends` clause.

Importing SSV files by scripting

In Dymola 2023, you can import an SSV file by the new built-in function `importSSV`. The input arguments are:

- `fileName` – a text string that is the name of the SSV file to import.
- `modelName` – a text string that is the name of the existing (base) model that the parameters are applied to.
- `packageName` – a text string that is the name of the Modelica package where the new model is to be inserted. (The name of the model is taken from the parameter set in the SSV file.) By default, this is an empty string, meaning that the created model is placed at the top level of the package hierarchy.

There is one output argument `result`. It is a text string that, in case of success, contains the full path of the created model. If the function call was not successful, the text string is empty.

Importing SSV files by a command

The existing command **File > Open > Import SSP...** can now also be used to import SSV files. When applying this command to an `.ssv` file (instead of an `.ssp` file), the command will call the `importSSV` function described above. The current root model will be used as the base model, and, by default, the newly created model is placed at the top level of the package hierarchy.

3.6.5 FMI Support in Dymola

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

FMU export

Max run time for an FMU

For this feature, see the option **Per simulation** in the section “Max run time per simulation” on page 24.

FMU import

Improved conversion options of unknown units for FMI 2.0

Two new flags are available for unit conversion when importing an FMU:

The flag `Advanced.FMI.ConvertUnits` is an integer flag that specifies when nonstandard Modelica units should be converted to their declared base units. The possible values of the flag are:

- 0 – Don’t convert any units.
- 1 – (Default) Convert nonstandard units with factor 1 and offset 0.
- 2 – Convert all nonstandard units. Units with factor $\neq 1$ and offset $\neq 0$ will have a different value in the Modelica wrapper.

The flag `Advanced.FMI.LogAllIgnoredUnitConversions` is a Boolean flag that is default `false`. If set to `true`, all units with skipped unit conversion will be listed.

Improved default value of the integration tolerance for FMI 2.0 Co-simulation FMUs

In previous Dymola versions, the default value of the integration tolerance of an imported FMU could be specified by the parameter `fmi_rTol` which was sent into the function `fmi2SetupExperiment`. The default value used was $1e-6$.

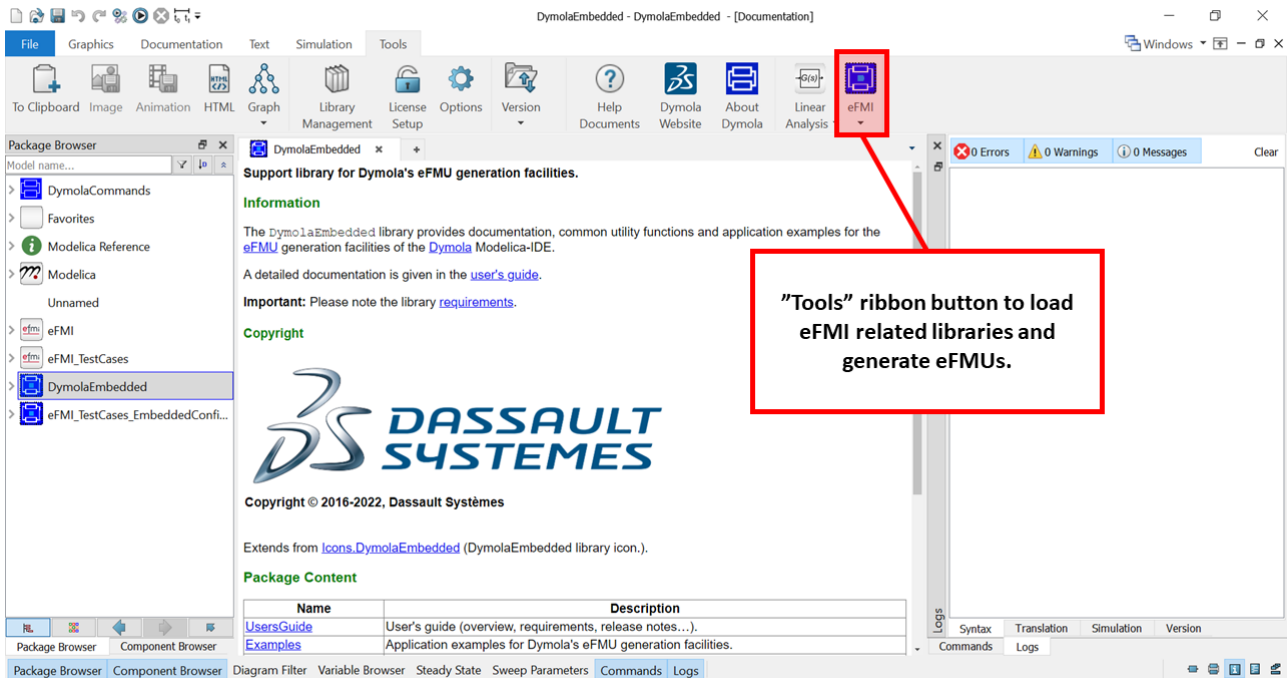
In Dymola 2023, you can select what default value to use by the new parameter `fmi_setTol`. The alternatives are:

- `fmi_setTol` is `false` (default value): The value of the integration tolerance in the exported FMU is used as default integration tolerance. (The integrator tolerance in an exported FMU from Dymola is usually taken from the simulation setup.)
- `fmi_setTol` is set to `true`: The parameter `fmi_rTol` is used to set the default integration tolerance of the FMU. The default value of `fmi_rTol` is the value given in the attribute `tolerance` of `defaultExperiment` in the `modelDescription.xml`, if this attribute exists. If it does not exist, the value $1e-4$ is used.

3.6.6 eFMI Support in Dymola

Dymola now supports eFMI according to the public available [eFMI Version 1.0.0-alpha.4](#) specification. Dymola's eFMI facilities comprise generation of eFMUs with Algorithm Code, CATIA ESP based Production and Binary Code containers, and the usage of generated eFMUs for Co-simulation from within Modelica models.

The **eFMI** button in Dymola's **Tools** ribbon can be used to load Dymola's eFMI facilities (by the menu entry **Load Libraries...**)



For eFMU generation, a Microsoft Windows Dymola installation must be used. A Dymola Source Code Generation License is required.

Further requirements, limitations, and documentation – in particular how to configure the eFMU generation - can be found in the “Users guide” of the DymolaEmbedded library. Just load the library via the **Load Libraries...** menu entry of the **eFMI** button in the **Tools** ribbon.

For an overview of eFMI, please see the paper [eFMI: An open standard for physical models in embedded software](#).

3.7 Visualize 3D

3.7.1 Surface plots

Using the `generateMeshGrid` function to generate parametric surface matrices

Already in previous Dymola versions, it was possible to format data for surface plotting, to be plotted by for example `Plot3D.plotSurface`. That was done using an internal function. This function is now made more visible by being added to the Utilities folder, as `Plot3D.Utilities.generateMeshGrid`. Help text is also added.

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 3.5.

Note that the Modelica Standard Library version 4.0.0 is compliant with the Modelica Language Specification 3.4.

3.9 Documentation

New white paper: Using Windows Subsystem for Linux with Dymola

Already in Dymola 2022x this white paper was available. It describes the installation of Windows Subsystem for Linux (WSL) and the corresponding setup needed in Dymola. Information about the use of WSL in Dymola is included, as well as useful compiler options.

The paper is available using the command **Tools > Help Documentation**, under the section “White Papers”. It is also available [here](#).

General note about the manuals

In the software, distribution of Dymola 2023 Dymola User Manuals of version “March 2022” will be present; these manuals include all relevant features/improvements of Dymola 2023 presented in the Release Notes.

Limited Availability (LA) features are not documented in the manuals.

3.10 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2023 are listed.

3.10.1 Hardware requirements/recommendations

Hardware requirements

- At least 2 GB RAM
- At least 400 MB 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 2023 is supported, as 64-bit application, on Windows 8.1, and Windows 10. 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 2023 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.

- Visual Studio 2012 Express Edition (11.0)
- 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 www.Dymola.com/compiler.
- Visual Studio 2019 Community (16)
- 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 www.Dymola.com/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

- Visual Studio 2012 (11.0)
- Visual Studio 2015 (14.0)
- Visual Studio Professional 2017 (15)
- Visual Studio Enterprise 2017 (15)
- Visual Studio Professional 2019 (16)
- Visual Studio Enterprise 2019 (16)

Intel compilers

Note!

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

MinGW GCC compiler

Dymola 2023 has limited support for the MinGW GCC compiler. The following versions have been tested and are supported:

- For 32-bit GCC: version 5.3, 6.3, and 8.2
- For 64-bit GCC: version 5.3, 7.3, and 8.1

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

To download any of these free compilers, please visit <http://www.Dymola.com/compiler> 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 GCC compiler. Usually, you can select this as an add-on when installing GCC.

Current limitations with 32-bit and 64-bit 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. **Note!** When migrating to Modelica Standard Library (MSL) version 4.0, MinGW gcc versions older than 5 are not guaranteed to work for source code export.
- 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.

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.

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

- The WSL Linux environment can compile the generated model C code from Dymola in order to produce a Linux executable dydosim or a Linux FMU. (To generate Linux FMUs, you must use a specific flag as well.)

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 2023 supports DSLS R2022x. Earlier DSLS versions cannot be used.

Linux

Supported Linux versions and compilers

Dymola 2023 runs on Red Hat Enterprise Linux 8.2, 64-bit, with gcc version 8.3.1, 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 2023 it is Qt 5.15
- 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.

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.

You can however still change the compiler by changing the variable `CC` in `/opt/dymola-<version>-x86-64/insert/dsbuild.sh`. As an example, for a 64-bit Dymola 2023 application:

```
/opt/dymola-2023-x86_64/insert/dsbuild.sh
```

Dymola 2023 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 Ubuntu: `sudo apt-get install g++-multilib libc6-dev-i386`
- Dymola is built with Qt 5.15.0 and thereby inherits the system requirements from Qt. This means:
 - Since Qt 5.15 no longer supports embedding of the XCB libraries, these must now be present on the platform running Dymola. See the table in <https://doc.qt.io/qt-5.15/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.
 - The library `libxcb-xinput.so.0` might require explicit installation.

- 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 2023 supports DSLS R2022x. Earlier DSLS versions cannot be used.

"