

Dymola

Dynamic Modeling Laboratory

Dymola Release Notes

Dymola 2025x Refresh 1

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

Document version: 1

© Copyright 1992-2025 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
Schelevägen 27 – Floor 9
SE-223 63 Lund
Sweden

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

Contents

1	Important notes on Dymola	5
2	About this booklet	6
3	Dymola 2025x 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	Interactive editing applied in the Modelica text editor – automatic indentation while you type	10
3.2.2	Improved support for arrays with dimensions that are not known until simulation	13
3.2.3	Minor improvements	13
3.3	Simulating a model	15
3.3.1	Improved generation of analytic Jacobian	15
3.3.2	Scripting	15
3.3.3	Minor improvements	20
3.4	Installation	23
3.4.1	General improvements	23
3.4.2	Installation on Windows	27
3.4.3	Installation on Linux	28
3.5	Features under Development	28
3.6	Model Experimentation	34
3.6.1	Extended Monte Carlo Support	34
3.6.2	Integrated GUI for Model Optimization	40
3.7	Model Management	48
3.7.1	Version management: Improvements in the 3DEXPERIENCE app “Design with Dymola”	48
3.7.2	Model structure: Improved model editing API	57
3.8	Other Simulation Environments	59
3.8.1	Dymola – Scilab interface	59
3.8.2	Dymola – Matlab interface	59
3.8.3	Real-time simulation	59
3.8.4	Java, Python, and JavaScript Interface for Dymola	61
3.8.5	SSP Support in Dymola	61
3.8.6	FMI Support in Dymola	64
3.8.7	eFMI Support in Dymola	72
3.9	Advanced Modelica Support	73

3.10	Modelica Standard Library and Modelica Language Specification	74
3.11	Documentation	74
3.12	Appendix – Installation: Hardware and Software Requirements	75
3.12.1	Hardware requirements/recommendations	75
3.12.2	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 2025x 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, Visual Studio C++ compilers older than version 2015 are no longer supported:
 - From Dymola 2024x Refresh 1, Visual Studio 2012 is not supported anymore.
 - From Dymola 2022x, Visual Studio 2013 is not supported anymore. (Visual Studio 2012 was however still supported until Dymola 2024x, due to the logistics of changing the oldest supported version)

Intel

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

MinGW GCC

Dymola 2025x 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 2025x Refresh 1 has support for the WSL (Windows Subsystem for Linux) GCC compiler, 64-bit. For more information about WSL GCC, see section “Compilers” on page 75, the section about WSL GCC compiler.

Clang compiler

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

Installation on Linux

To translate models, Linux relies on a GCC compiler, which is usually part of the Linux distribution. Refer to section “Supported Linux versions and compilers” on page 79 for more information. Note that you can use Clang as code generator.

2 About this booklet

This booklet covers Dymola 2025x 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 2025x Refresh 1

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2025x Refresh 1. In particular, Dymola 2025x Refresh 1 provides:

Interactive editing in the Modelica text editor – automatic indentation while you type

Entering text in the Modelica text editor is improved by automatic indentation while you type new code. See section “Interactive editing in the Modelica text editor – automatic indentation while you type” starting on page 10.

Extended Monte Carlo Support

The Monte Carlo functions in the package `Design.Experimentation` have been expanded and moved to a new package: `Design.Randomization`. A key new feature for the `Design.Randomization` package is the option to perform Monte Carlo analysis on the entire trajectory instead of only at the last point. With trajectory analysis enabled, the Monte Carlo run is summarized by plots showing the mean value, standard deviation, and prediction intervals over time. The option to enable trajectory analysis is available in the Sweep UI Advanced Options dialog. Other new features in the package include:

- A better method of generating random numbers (with Latin hypercube sampling) which ensures faster convergence.
- Box-plots for the observed variables at the last point.
- Sensitivity analysis of parameters with Sobol indices.

For more information, see the section “Extended Monte Carlo Support”, starting on page 34.

Integrated GUI for Model Optimization

An integrated GUI for model optimization from the Optimization library from DLR Institute of Vehicle Concepts is added. The basic look and feel is similar to the present GUI for sweeping. See page 40.

FMI Improvements

- **FMU export: Input interpolation for FMI 3 Co-simulation FMUs** To improve the efficiency of Co-simulation FMUs, smoothing of inputs allows the integrator to continue the simulation without any reset. To further help the integrator, you can also activate predictor compensation (see page 64). Note that these features were already implemented for FMI 2 in the previous Dymola version.

- **FMU import: Source code FMU import in Dymola** You can now import FMUs with source code, both by command and by scripting (starting page 69). Notes!
 - FMUs of FMI version 1 are not supported for source code FMU import.
 - For source code FMUs generated by Dymola, importing such FMUs generated with a Dymola version earlier than 2025x Refresh 1 is not supported.

eFMI: Support for importing eFMUs in Simulink®

eFMI in Dymola 2025x Refresh 1 includes support to generate MATLAB® scripts that import an eFMU Production Code container generated by Software Production Engineering as a Simulink® C Function block. See page 72.

Improved support for arrays with dimensions that are not known until simulation

The previous limitation of maximum size of the arrays has been removed. See page 13.

Improved generation of analytic Jacobian

It is now possible to generate analytic Jacobian for models with dynamic state selection, see page 15.

Some minor items

- Dymola 2025x Refresh 1 supports the future Modelica Standard Library 4.1.0 version, see page 74.
- On-demand download and installation of libraries from GitHub, see page 23.
- FMI 1.0 export alternative removed from menus (however still available by scripting) – see page 66.

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:

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

- Aviation Systems Library, version 1.6.2
- Battery Library, version 2.8.1
- Brushless DC Drives Library, version 1.4.3
- ClaRa DCS Library, version 1.7.6

- ClaRa Grid Library, version 1.7.6
- ClaRa Plus Library, version 1.7.6
- Claytex Library, version 2025.1
- Claytex Fluid Library, version 2025.1
- Cooling Library, version 1.5.4
- Dassault Systemes Library, version 1.14.0
- Design Library, version 1.2.3
- Dymola Commands Library, version 1.19
- Dymola Embedded Library, version 1.0.5
- Dymola Models Library, version 1.10.0
- Electric Power Systems Library, version 1.7.0
- Electrified Powertrains Library (ETPL), version 1.11.0
- Fluid Dynamics Library, version 2.19.0
- Fluid Power Library, version 2025.1
- FTire Interface Library, version 1.3.2
- Human Comfort Library, version 2.19.0
- HVAC (Heating, Ventilation, and Air Conditioning) Library, version 3.4.0
- Hydrogen Library, version 1.4.2
- Model Management Library, version 1.4.1
- Multiflash Media Library, see Thermodynamics Connector Library
- Pneumatic Systems Library, version 1.7.2
- Process Modeling Library, version 1.3.0. **Note.** This library is currently not supported on Linux.
- SMArtInt Library, version 0.5.1
- Testing Library, version 1.10.0
- TIL Suite 2025.1:
 - TIL 2025.1
 - TIL Automotive 2025.1
 - TIL Heat Storage 2025.1
 - TIL PCM Storages 2025.1
 - TIL NTU 2025.1
 - TIL Hydrogen Energy Systems 2025.1
 - TIL Adsorption 2025.1
- Thermodynamics Connector Library (previously named Multiflash Media Library), version 1.3.0. **Note.** This library is currently not supported on Linux.
- VeSyMA (Vehicle Systems Modeling and Analysis) Library, version 2025.1
- VeSyMA - Engines Library, version 2025.1
- VeSyMA - Powertrain Library, version 2025.1

- VeSyMA - Suspensions Library, version 2025.1
- VeSyMA2ETPL Library, version 2025.1
- Visa2Base, version 1.18
- Visa2Paper, version 1.18
- Visa2Steam, version 1.18
- Wind Power Library, version 1.1.6

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 Interactive editing applied in the Modelica text editor – automatic indentation while you type

Interactive editing in the Modelica text editor has been implemented. While you type Modelica code, automatic indentation is applied for nesting and grouping-operators.

The following principles were followed for indentation:

- The current line is always auto-indented whenever space or enter is pressed.
- The next new line is always auto-indented whenever enter is pressed.
- Nested block termination changes the indentation of the current line (like when ending an if-statement or a class).
- Additional cases for conditional blocks change the indentation of the current line, but also define a new for the next (like else if), but only if the middle-keyword introducing the condition is the very first thing of the line.
- Multi-line expression groups and pending expression termination via ; change the indentation of the next line (like when closing a pending function argument list).
- When we have a multi-line string, string continuing lines are not indented; indentation is proceeded the next line after the one terminating the string. Although un-indented, any indentation changes due to expressions etc. are protocolled and considered when indentation continues.

Below an example of the new interactive indentation and syntax highlighting in the Modelica text editor (just assume the code is written from scratch, with every new line autoindented while typing; no further manual formatting). **Notes:**

- This it is not a complete Modelica model (declarations are missing etc.), but an illustration of how the interactive typing works now.
- In the text below only the highlighting of keywords are marked, not comment texts. However, both keyword and comment text coloring is automatically handled.
- As in previous versions, to color types, operators, etc., you must a syntax highlighting command.

To better illustrate the indentation, *after* the example, the indentation has been exchanged by hyphens, illustrating the number of spaces that are used in the indentations.

Result when typing.

```
model M
    // Pending statement group:
    Real r1 = 2
        * f()
        * g();
    // Pending expression groups:
    Real r2 = 1 + f(
        2
        + f(
            1,
            g(
                h(
                    j(
                        1,
                        2)),
                    3)),
            g(
                h));
    Real m[2,3] = {
        {
            1,
            2,
            3},
        {
            4,
            5,
            6}};
    Real n[2,3] = [
        [
            1,
            2,
            3];
        [
            4,
            5,
            6]];
    // Multi statements:
algorithm
    if a then
        for j in k loop
            if a then
                j = 1;
            end if;
            j = 2;
        end for; end if;
    // Multi-line string craziness:
    s = "Spaghetti code
is so nice,
hooray" +
    f() +
    g(", hooray
```

```

hooray, juchu!",
  1,
  " Looks
very nice indeed,
or?")
  + 1;
s2 = "What do you
say?";
// End of example!
end M;

```

Preceding hyphens for illustration!

```

model M
--// Pending statement group:
--Real r1 = 2
----* f()
----* g();
--// Pending expression groups:
--Real r2 = 1 + f(
----2
----+ f(
----1,
----g(
----h(
----j(
----1,
----2)),
----3)),
----g(
----h));
--Real m[2,3] = {
----{
----1,
----2,
----3},
----{
----4,
----5,
----6});
--Real n[2,3] = [
----[
----1,
----2,
----3];
----[
----4,
----5,
----6]];
--// Multi statements:
algorithm
--if a then
----for j in k loop
-----if a then

```

```

-----j = 1;
-----end if;
-----j = 2;
--end for; end if;
--// Multi-line string craziness:
--s = "Spaghetti code
is so nice,
hooray" +
----f() +
----g(", hooray
hooray, juchu!",
-----1,
-----" Looks
very nice indeed,
or?")
----+ 1;
--s2 = "What do you
say?";
--// End of example!
end M;

```

3.2.2 Improved support for arrays with dimensions that are not known until simulation

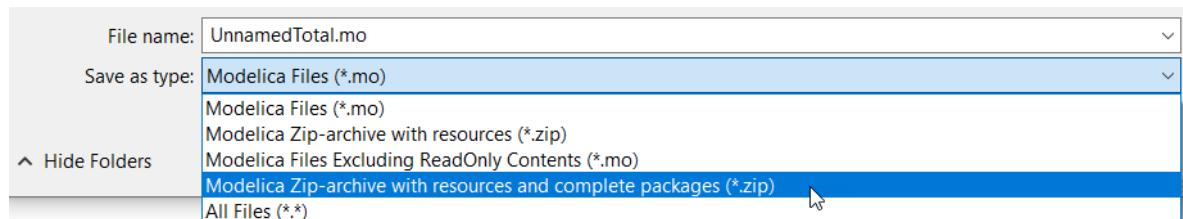
In Dymola 2025x Refresh 1, the limitation that the total number of array elements is statically limited is removed, Dymola automatically allocates sufficient memory for the array elements without any max size (as long as memory is available). This is done by automatically allocate dynamic memory when the static memory is not sufficient.

(The static memory is by default limited to 70 000 array elements, but can be increased by setting the flag `Advanced.Translation.RealBufferSize = 10*N`, where N in our case is the wanted limit for array elements in the static memory. This means that if we want a max size of 100 000 array elements in the static memory, we should set `Advanced.Translation.RealBufferSize = 1 000 000.`)

3.2.3 Minor improvements

Save Total option that stores entire packages

If you save using the command **File > Save... > Save Total**, you have a new option, when selecting **Save as type:**, to select **Modelica Zip-archive with resources and complete packages (*.zip)**:



The difference from the already present alternative **Modelica Zip-archive with resources (*.zip)** is that the new option saves complete packages/libraries, not only used parts. Note however that standard libraries are not saved.

The corresponding built-in function `saveTotalModel` has also been updated; see section “The built-in function `saveTotalModel` improved” on page 16.

Improved shadow warnings

In previous Dymola versions, if you tried to add a new model P in a package O.P, you got a warning about shadowing.

In Dymola 2025x Refresh 1, you also get a warning if you try to add a new model O in a package O.P.

Option to not display a created Design Structure Matrix (DSM) in the web browser

In previous Dymola versions, the command **Tools > Graph > DSM** created a Design Structure Matrix (DSM), stored it on file, and unconditionally opened the DSM in your web browser.

In Dymola 2025x Refresh 1, if you set the new flag `Advanced.File.OpenDSM = false`, the DSM is created and saved, but it is not opened in your web browser. This may be useful if you e.g. want to create many DSM files. (The default value of the flag is `true`.)

Allowing string parameters enabled by default

In previous Dymola versions, the default value of the flag `Advanced.AllowStringParameters` was `false`.

In Dymola 2025x Refresh 1, the name of the flag is changed to `Advanced.Translation.AllowStringParameters`, and the default value is changed to `true`.

Inference for `absoluteValue`

Relative temperatures will now automatically use unit-conversion without offset even without annotation (`absoluteValue=false`) if it is equal to another relative temperature or the difference between two absolute temperatures.

Further improved unit checking

In Dymola 2025x, a number of improvements according to the suggested Modelica language improvement of unit checking (<https://github.com/modelica/ModelicaSpecification/pull/3491>) were implemented.

In Dymola 2025x Refresh 1, this unit checking has been further improved, in particular unit checking can infer units by combining constraints from multiple equations.

Highlighting of dynamic typing (inner/outer)

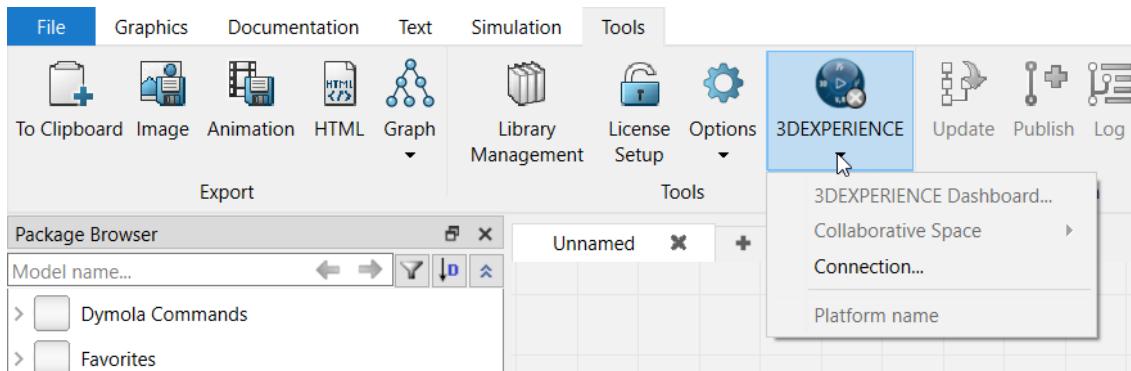
You can activate highlighting of dynamic typing (inner/outer) by setting the flag:

```
Advanced.Editor.Highlight.Outer = true  
(The default value of the flag is false.)
```

If you activate this feature, inner and outer components are shown sunken, as a socket, in the same way as replaceable components.

New command in the Tools ribbon

The Tools ribbon has been extended with a new icon that gives access to a number of commands related to using the 3DEXPERIENCE platform for versioning:



For more information, see section “3DEXPERIENCE command in Tools ribbon” on page 48.

3.3 Simulating a model

3.3.1 Improved generation of analytic Jacobian

- It is now possible to generate analytic Jacobian for models with dynamic state selection. As an example, consider the model Modelica.Mechanics.MultiBody.Examples.Elementary.ThreeSprings. Note that functions without derivatives and algorithms might still prevent analytic Jacobians.
- Note also that in general, the extra cost of generating analytic Jacobian for some large models has now been reduced.

3.3.2 Scripting

Stop execution instead of demand-loading unwanted library versions when scripting

When you use scripting, usually you avoid pop-up dialogs by setting the flag `Advanced.UI.SkipMessageDialogs=true`. This flag is automatically activated when you start Dymola with the command line option `/externalInterface`. That is the case when you use Python.

If you use scripting to load multiple libraries with dependencies, and you want to use certain library versions, you might in some rare cases demand-load unwanted library versions.

To avoid this, you can set the flag `Advanced.UI.DemandLoadExactVersion=true`. With this flag activated, the following is valid for some built-in functions:

- `openModel` only demand-loads libraries of the exact version defined in the library to open
- `openModelFile` returns `false` if the exact version specified by the argument `version` is not found.

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

Note. Enhanced information about how to load multiple libraries with dependencies by scripting is added in the Dymola manual for Dymola 2025x Refresh 1.

Using `loadResource` in conditional mode

The `Modelica.Utilities.Files.loadResource`-function can now handle non-evaluated strings. The benefit is that you can change a string parameter after translation to switch to a different resource - using a modelica-URI - exactly as before translation.

There are some limitations:

- For normal simulation it requires that the library containing the resource have been loaded before translating the model, and that the library has a "Resources" directory.
- For exported case, it is more limited: there must be another `loadResource`-call using a literal (or evaluated parameter) referring to the same modelica-URI (or an enclosing directory).

The FMU-case is also less relevant as string-parameters are by default evaluated for FMUs.

The exported FMU-case is used for FMUs and in some other cases when making an executable for export, assuming you keep the default of copying resources.

The built-in function `importFMU` improved

The built-in function `importFMU` now has a new Boolean input argument `sourceCodeImport` to allow source code FMU import. For more information, see section "FMU Import: Source code FMU import in Dymola" starting on page 69.

The built-in function `openModelFile` improved

The built-in function `openModelFile` now has a new Boolean input argument `newTab`. By default, the argument is `false`, corresponding to the previous behavior of opening the model in the current active tab. Setting it to `true` means that the model is opened in a new editor tab.

The built-in function `saveTotalModel` improved

The built-in function `saveTotalModel` now has a new Boolean input argument `completePackage`. By default, the argument is `false`, corresponding to the previous behavior of only saving used parts of packages/libraries. Setting it to `true` means that the

complete packages/libraries are saved. (By default, however, standard libraries/packages are not saved. This can be changed, as before, using the argument `skipStandard`.)

New, changed, or deleted advanced flags

New Advanced flags

New flags added in Dymola 2025x Refresh 1:

New flag	Default value	Description	Saved between sessions
Advanced.Beta.FMI.StructuredIO	false	<p>Only works if Advanced.FMI.StructuralDeclaration is true, preserve hierarchical structure of IO in FMI2.</p> <p>See section “Option to preserve the hierarchical structure of IO in FMI 2” on page 32.</p>	No
Advanced.Beta.Translation.Generate.AdjointDerivatives	false	<p>Generate adjoint derivatives for the ODE problem.</p> <p>See section “Generating adjoint derivatives for the ODE problem” on page 32.</p>	Yes
Advanced.Beta.Translation.GenerateEventChangeVariability	false	<p>As proposed, make variability in functions with GenerateEvents = true behave similarly as non-functions.</p> <p>See section “Improved variability check for GenerateEvents-functions” on page 33.</p>	No
Advanced.Editor.Highlight.Outer	false	<p>Highlight inner and outer components in diagram.</p> <p>See section “Highlighting of dynamic typing (inner/outer)” on page 14.</p>	Yes
Advanced.FMI.SourceCodeImport	false	<p>Whether source code or binaries will be used to run the FMU.</p> <p>See section “FMU Import: Source code FMU import in Dymola” starting on page 69.</p>	No
Advanced.File.OpenDSM	true	Open HTML DSM after generation.	Yes

		See section “Option to not display a created Design Structure Matrix (DSM) in the web browser” on page 14.	
Advanced.License.Type	"" (empty string)	Type license key (F=FlexNet, D=DSLS, empty=auto-detect). See section “Simplified license handling at start-up” on page 27.	Yes
Advanced.Plot. AdjustMeasurementCursor	true	Adjust the measurement cursor when changing horizontal range. For current plot window.	Yes
Advanced.SSP. ApplyUnitConversion	true	Apply unit conversions when reading SSP file. See section “Unit conversion when importing an SSP, SSD, or SSV file” on page 63.	Yes
Advanced.SSP. TopLevelPackageSuffix	"" (empty string)	Suffix appended to top-level package created by SSP import. See section “Option to add suffix for imported SSP or SSD file names” on page 63.	Yes
Advanced.Translation.Assert. IncludeValue	1	For assertions automatically include values of variables in assert messages; 0 not, 1 – if seem to be missing, 2 – always. See section “Option to include values in assert-messages” on page 73.	No
Advanced.Translation.Log. BusUndeclaredWarning	false	Give a message if bus variables are not declared. See section “Finding undeclared signals in expandable connectors” on page 20.	Yes
Advanced.Translation. LongDiagnostics	false	Use longer diagnostics messages in the simulation log. See section “Option to allow longer diagnostics/error messages” on page 20.	No
Advanced.Translation.MT	false	Link with /MT option and libraries (outside of FMUs) for Visual Studio.	Yes

		See section “Option to force all libraries in a compilation to use the linker option /MT” on page 27.	
Advanced.UI.DemandLoadExactVersion	false	When not showing dialogs we treat version mismatch as “Cancel”. See section “Stop execution instead of demand-loading unwanted library versions when scripting” on page 15.	Yes
Advanced.UI.ProgressBar	true	Enable progressbar in the ribbon. See section “New simulation progress indicator” on page 21.	Yes

Removed Advanced flags

Removed flags in Dymola 2025x Refresh 1:

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

Changed Advanced flags

Changed flags in Dymola 2025x Refresh 1:

Changed flag	Change	Description
Advanced.AllowStringParameters	Name and default value (to true).	Allow string parameters. Name changed to Advanced.Translation.AllowStringParameters. See section “Allowing string parameters enabled by default” on page 14.
Advanced.Beta.Translation.FunctionsUseConstantsInsteadOfExpressions	Name and default value (to true).	Use global constants instead of substituting expressions in functions (for arrays using non-built-in functions). Name changed, “Beta.” removed from name. See section “For arrays using non-built-in functions, an option to use global constants instead of substituting

		expressions in the functions” on page 21.
Advanced.Editor.MediaPropagation	Default value (to 1) and now saved between sessions .	The new default value means that by default, media propagation using components from Modelica.Fluid are used, with dialog for approval.
Advanced.FMI.FMUSourceCodeUniqueNaming	Default value (to true)	Whether to append unique id to file name all.c See section “FMU Export: By default use unique naming when generating source code FMUs” on page 68.
Advanced.Translation.EmbeddedOptimizeForOutputs	Name changed	Only generate code required for calculating the outputs. Name changed to Advanced.Translation.OptimizeForOutputs See section “Handling components not connected” on page 22.

3.3.3 Minor improvements

Finding undeclared signals in expandable connectors

To find signals that are added to an expandable connector (e.g. via a connect statement) but that are not declared in the definition of the expandable connector, you can set the flag:

```
Advanced.Translation.Log.BusUndeclaredWarning = true
```

When you translate the model, such signal are presented as warnings in the translation log.

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

Option to allow longer diagnostics/error messages in the simulation log

By default, the length of diagnostics/error messages are limited to 400 characters in the simulation log. To allow longer- messages, you can set the flag:

```
Advanced.Translation.LongDiagnostics = true
```

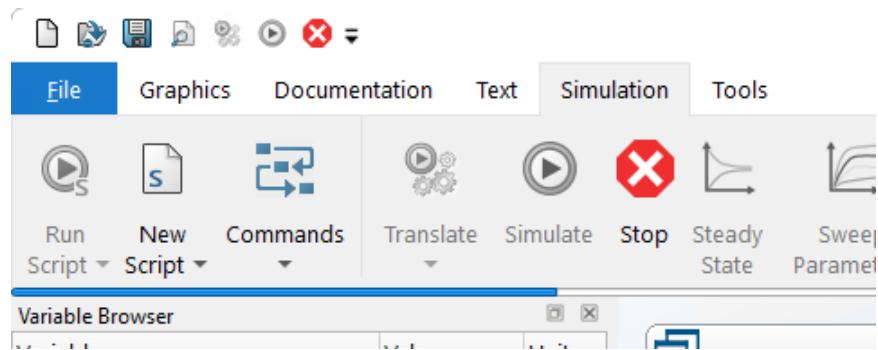
(The flag is by default `false`.) By setting this flag to `true`, the following is allowed in the simulation log:

- For string literals: maximum 4000 characters
- For strings: maximum 10000 characters

New simulation progress indicator

A new simulation ribbon progress indicator is available when a simulation is running. This feature is by default activated, but if you don't want to use it, you can set the flag `Advanced.UI.ProgressBar = false`. (The flag is by default `true`.)

Note that this indicator is currently not supported in Dymola dark mode.



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

This is a part of improving the code generation for Modelica functions that was implemented in Dymola 2025x. This option may result in an even faster function execution. However, it might require additional start values in the model. The option is by default activated, to disable it, set the flag `Advanced.Translation.FunctionsUseConstantsInsteadOfExpressions=false`. (The default value of the flag is `true`.)

(This feature was a beta feature in the previous Dymola version. The flag has changed name and default value in Dymola 2025x Refresh 1.)

Parallelization for external objects

For parallelization involving external functions, it is important to:

- Provide estimates of their complexity (in terms of the number of instructions per call).
- Ensure that they are thread-safe and declare that.

For the first item, you can now use the new annotation

```
annotation (__Dymola_OperationCount=<EstimatedNumberOfInstructionsPerCall>);
```

A typical external function intended for parallelization would thus be:

```
function foo
...
external"C" foo(...) annotation(Library="SomeLibrary");
annotation(__Dymola_ThreadSafe=true, __Dymola_OperationCount=5000);
end foo;
```

Handling components not connected to output

Note! This feature must currently be seen as a beta feature, although the flag is not a beta flag. This means that it is not yet documented in the manuals, only here in Release Notes.

In some cases, your application may contain alternative sets of components to execute. As an example, you may have several controllers, but you only want to use one, that is, connect one of them to the rest of the application for a certain simulation, and leave the others unconnected.

To handle such cases, you can set the flag:

```
Advanced.Translation.OptimizeForOutputs = true
```

(The flag is by default `false`.) When you set this flag, the only code generated is code for calculating the outputs of the complete model. This means that for any unconnected connector, or for any connector that goes no-where, the corresponding code is skipped. This can improve simulation speed, and in some cases solve hard initialization problems.

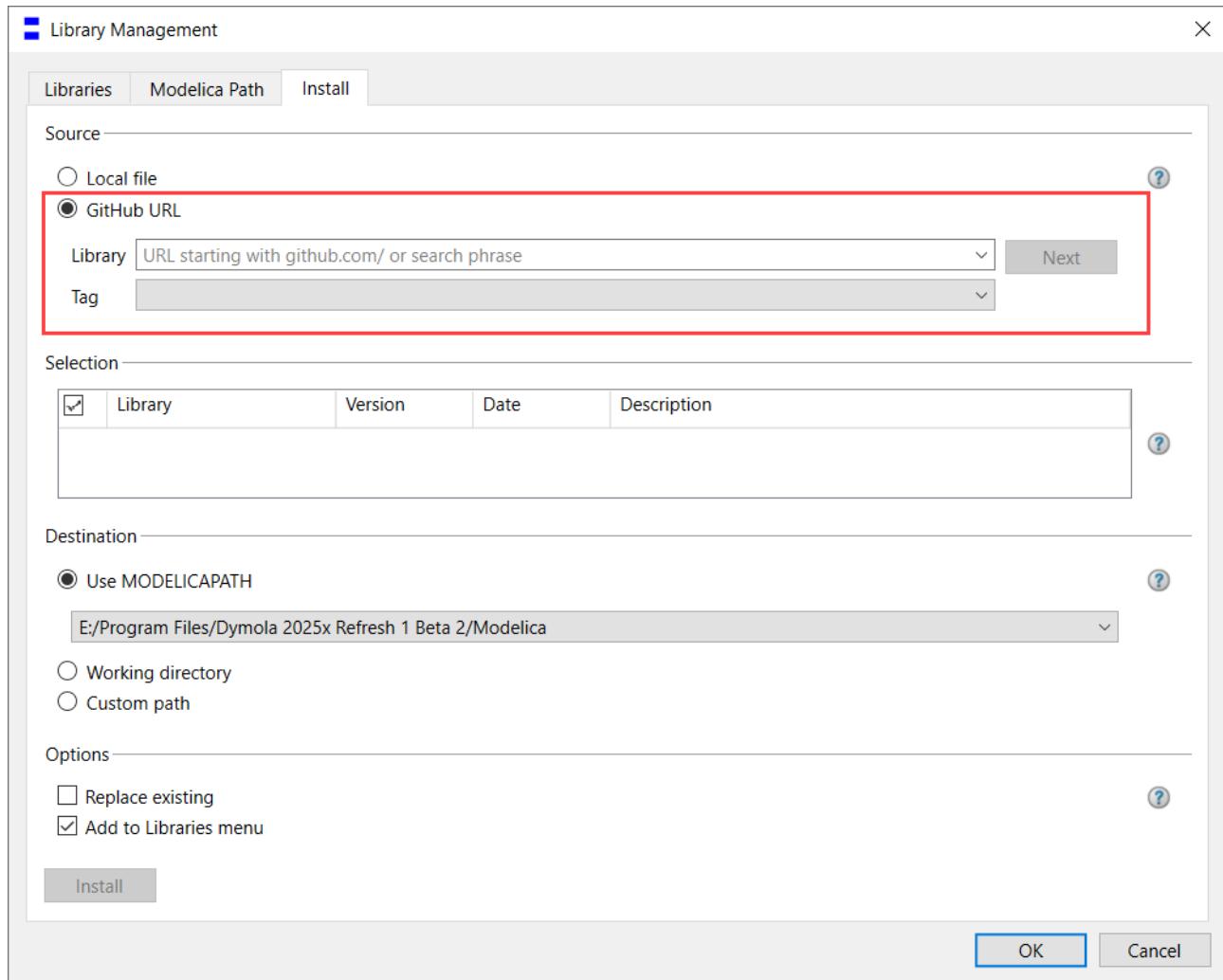
3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 75.

3.4.1 General improvements

On-demand downloading and installing libraries from GitHub

You can download and install libraries from GitHub by the command **Tools > Library Management**, the **Install** tab:



Note. This command is only for library usage. For library development, cloning a Git repository is recommended. You can use the command **Tools > Version > More > Git Clone** (if you have first selected to use Git as versioning system).

To do such a download and installation from GitHub, do the following:

In the **Library** field, start entering a GitHub URL or part of a library name, and then click **Next** (in this example we search for a library name):

Source

Local file

GitHub URL

Library ?

Tag ▼

▼ Next ?



The result is that the first hit found is displayed in the **Library** field, and if more hits are found, you can use the dropdown menu to select from them.

Once you have the library you want in **Library** field, click **Next**:

Source

Local file

GitHub URL

Library ?

Tag ▼

▼ Next ?



The result is that the available versions for the chosen library can be selected from the **Tag** field; the latest available version is displayed in that field:

Source

Local file

GitHub URL

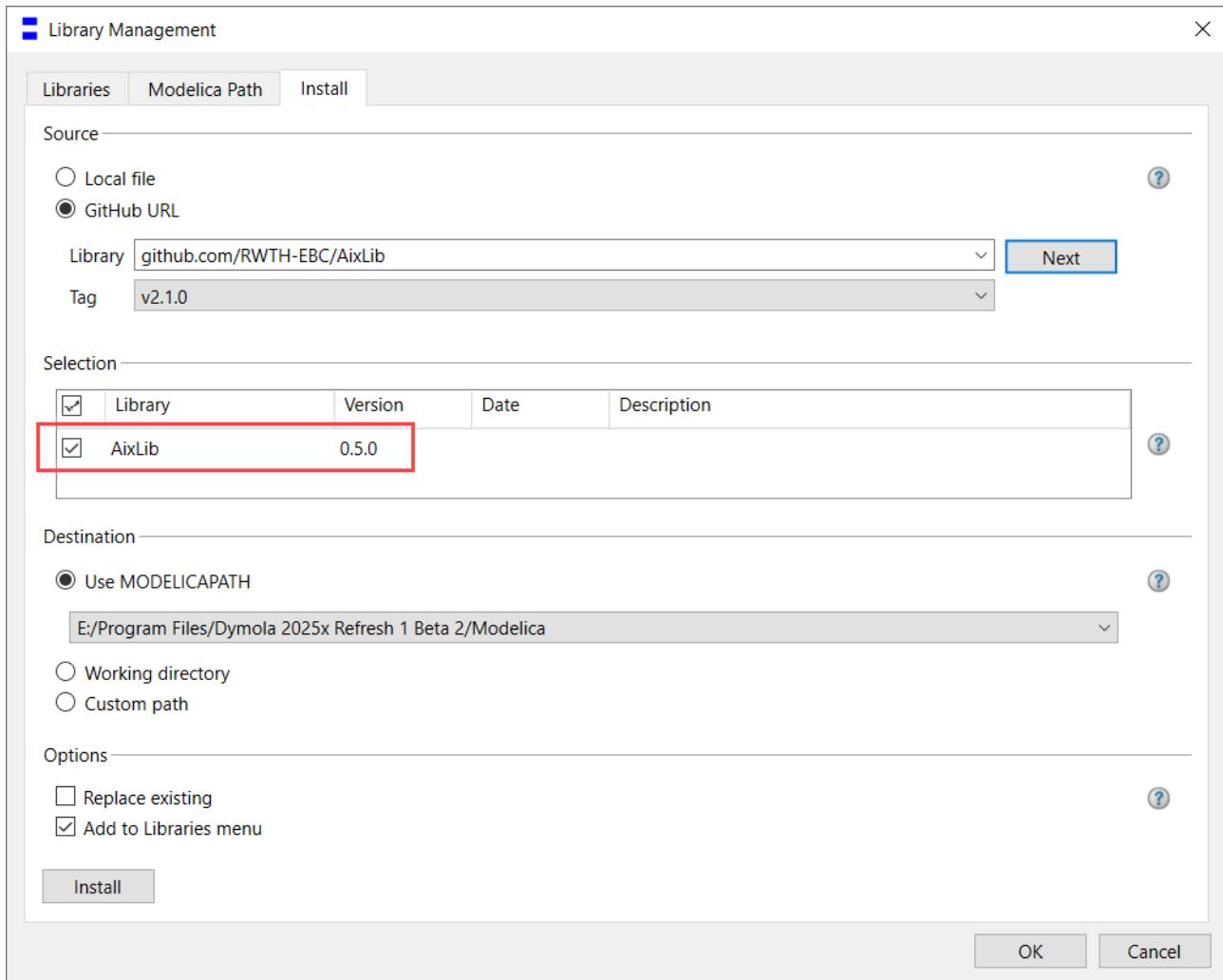
Library ?

Tag ▼

▼ Next ?



Select the version you want, and then click **Next**. The result is that the possible downloads of the selected library are presented in the **Selection** group (note that fetching this information may take some time):

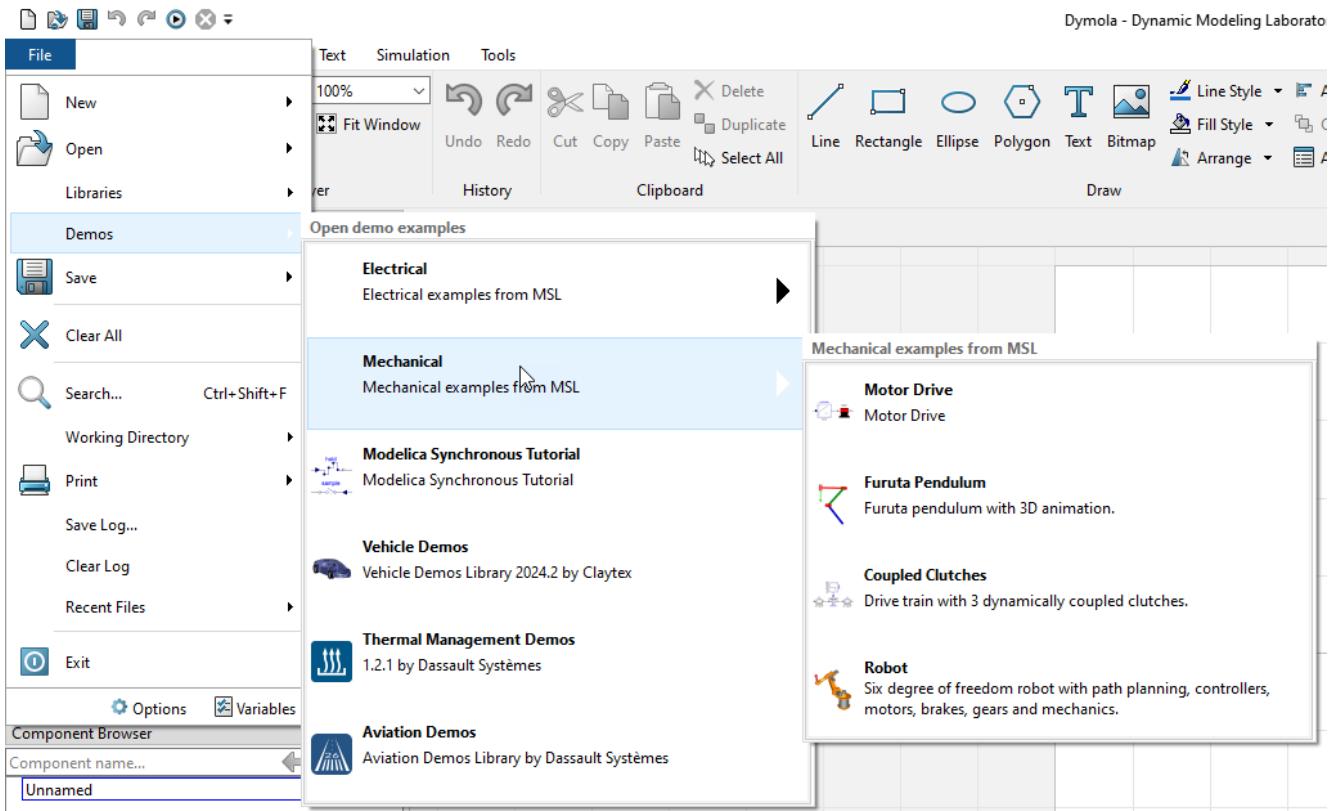


Select what you want to download and install from the **Selection** group, (in this case you have only one library) and specify your selections for the **Destination** and **Options** group.

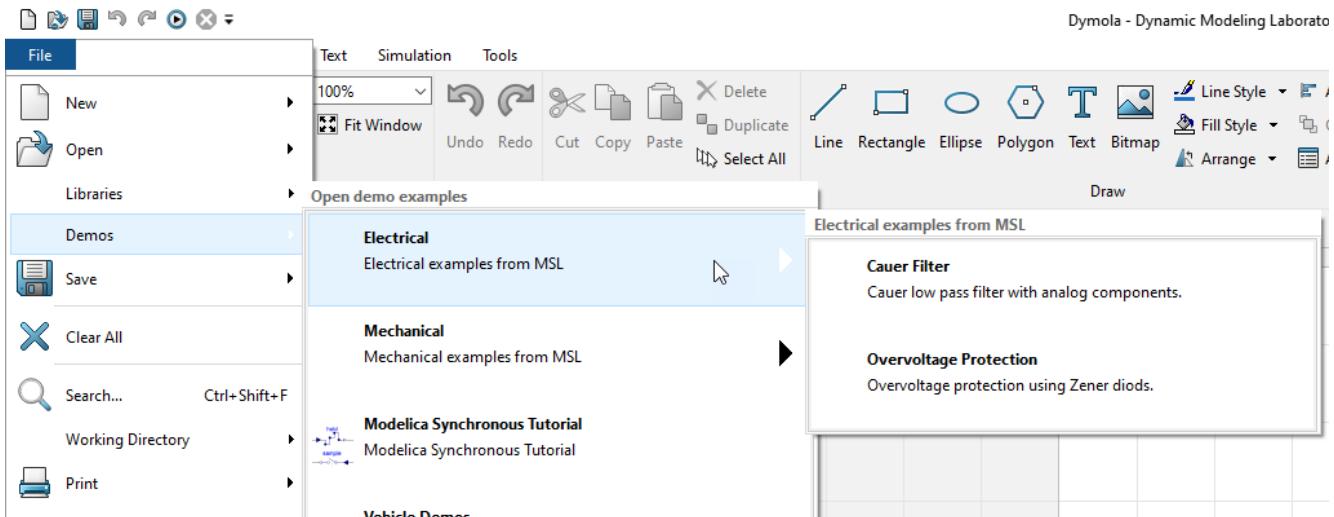
Now you can click **Install** to install your selections.

Improved GUI for opening demos

The GUI for the command **File > Demos** has been improved, with revised structure, more texts, and more demos. The mechanical examples have been collected in a separate section:



The **Electrical** is a new section:



Automatically avoiding double Libraries menu entries

Library developers may sometimes find duplicate entries in the **File > Libraries** menu. The reason is typically that the `MODELICAPATH` system variable contain multiple library sources. This can be checked (and fixed) using **Tools > Library Management > Modelica Path**.

New in Dymola 2025x Refresh 1 is that Dymola automatically tries to remove duplicate entries in the **File > Libraries** menu. Identical is defined as:

- Same library name
- Same description
- Same version number

If there are multiple entries, the first one in `MODELICAPATH` will be used.

Simplified license handling at start-up

Two license formats are supported in Dymola, FLEXnet and DSLS. To specify what type of license you want to use, you can start Dymola using command line options `/FLEXnet` or `/DSLS`.

However, you can start Dymola without specifying what license format to use. In such case, in previous Dymola versions, Dymola first tried FLEXnet, and if that failed, switched over to DSLS.

To be able to, for example, improve the diagnostics, in Dymola 2025x Refresh 1, a new variable `Advanced.License.Type` is used. It can have three values:

- F for FLEXnet
- D for DSLS
- "" (empty string) for unspecified format. This is the default.

When you set up a new license, the variable is automatically set accordingly. When starting a new session, the stored value of the parameter is used to specify what license format to use at start-up.

3.4.2 Installation on Windows

Option to force all libraries in a compilation to use the linker option /MT

When you use any Visual Studio compiler (or the corresponding Clang) you can use compiler linker options to specify the linking when compiling.

Out of the possible options, two are of particular interest here:

- `/MD` – Causes the application to use the multithread-specific and DLL-specific version of the runtime library.
- `/MT` – Causes the application to use the multithread, static version of the runtime library.

Currently the available libraries are not by default set up to use the same option, in some rare cases, you may have a mixture that causes problems when compiling.

To force all libraries to use the `/MT` linker option when compiling (outside FMUs), you can set the flag:

```
Advanced.Translation.MT = true
```

The flag is by default `false`, which means that the present behavior of allowing a mix is preserved. The value of the flag is saved between sessions.

Note that it is recommended to use the above flag instead of specifying `/MT` in the simulation setup. That is, *do not* specify this option by the command **Simulation > Setup**, the **Compiler** tab, in the **Custom options** group, the **Linker** input line – use the above flag instead.

Updated Qt version

Dymola 2025x Refresh 1 is built with Qt 6.8.1.

3.4.3 Installation on Linux

Updated Qt version

Dymola 2025x Refresh 1 is built with Qt 6.8.1.

3.5 Features under Development

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

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

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

The beta features that are put on flags are grouped by `Advanced.Beta` flags in **Tools > Options > Variables...**: An example from Dymola 2025x:

Flags and Variables		
filter variables		<input type="checkbox"/> Non-Default
Name	Value	Description
Evaluate	<input type="checkbox"/>	Evaluate parameters to reduce models (improves simulation speed).
Advanced		
> Animation		
Beta		
AllowInvertSemiLinear	<input type="checkbox"/>	Allow inverting the semiLinear function
Check		
TopLevelStructuralQuick	<input type="checkbox"/>	Quick top level structural check
EnableUsingModifierLogic	1	0 - always old variant, 1 smart, 2 always new modifier variant; to get enable depending on class parameters to work
FMI3		
ExportTerminalsAndIcons	<input type="checkbox"/>	Generate terminals and icons
Translation		
FunctionsUseConstantsInsteadOfExpressions	<input type="checkbox"/>	Use global constants instead of substituting expressions in functions (for arrays using non-builtin functions)
UI		
DefaultThickness	<input type="checkbox"/>	Use standard default thickness (0.25mm)
Check		

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 2025x Refresh 1, the following “under development” features are available:

Dymola Modelica Compiler tool (not on flag)

Overview

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

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

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

Operation

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

`dmc.exe`

It supports the following command line arguments:

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

Constraints

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

- There is no user interface to set up the license server, the C compiler, etc. Instead, DMC will read the Dymola startup script to get the initial setup.
- Commands that are graphical in nature are not implemented, for example, anything related to plotting and animation. Likewise, operations that use the graphical representation of the model, such as `exportDiagram()`.
- DMC uses a Dymola license and licenses for optional products, such as libraries. To ensure uninterrupted execution of a sequence of DMC commands, the license will remain checked out for a short period after the last DMC execution.
- DMC is in Beta-state for this Dymola 2025x Refresh 1 release.

Using DMC with Python

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

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

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

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

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

(This feature appeared in Dymola 2025x.)

Allowing inverting semiLinear calls

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

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

Smarter top-level structural check

A new beta-feature is that the top-level structural check can be restricted to only check the top-level model, ignoring sub-components, by setting the flag `Advanced.Beta.Check.TopLevelStructuralQuick=true` (in addition to `Advanced.Check.TopLevelStructural=true`). This is currently faster than regular check, and still detects relevant issues at the top. (Both flags are by default `false`.)

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

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

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

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

(This feature appeared in Dymola 2025x.)

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

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

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

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

FMI 3: Support for terminals and icons

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.)

Option to use default thickness according to specification

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

(This feature appeared in Dymola 2025x.)

Option to preserve the hierarchical structure of IO in FMI 2

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

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

(The flag is by default `false`.)

Notes:

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

As example of the effect, consider a hierarchical connector

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

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

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

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

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

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

Generating adjoint derivatives for the ODE problem

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

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

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

(The flag is by default `false`.)

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

Improved variability check for GenerateEvents-functions

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

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

(The flag is by default `false`.) The basic idea is that when events are generated for a function returning a Boolean based on a Real there should be no error that the Boolean is a continuous-time Boolean (as it isn't), but instead an error if the function uses `noEvent` internally.

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

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

3.6 Model Experimentation

3.6.1 Extended Monte Carlo Support

Changes in the Design Library

The Monte Carlo functionality in the package `Design.Experimentation` has been expanded and moved to a new package: `Design.Randomization`. The `Design.Randomization` package provides functions for running Monte Carlo simulations on Modelica models, processing the simulation output, and plotting the results.

The Design Library version is now Design 1.2.3., where the new features are related to the `Design.Randomization` package. The sweep UI has been appended with new options for Monte Carlo and running a Monte Carlo sweep will now call the corresponding functions `Design.Randomization.MonteCarloAnalysis` and `Design.Randomization.Analysis.MonteCarloAnalysis` in the `Design.Randomization` package.

Please note that the functions `Design.Experimentation.MonteCarloAnalysis` and `Design.Experimentation.Analysis.MonteCarloAnalysis` are still available but any new features and updates will be added to the corresponding functions in `Design.Randomization`.

New features

New (and better) sampling method: Latin Hypercube Sampling

Up until now, Monte Carlo analysis in Dymola used pseudo-random (PR) numbers to sample the parameter values. While it is a good simulation of randomness, it can leave some regions of the parameter space under-sampled, especially for higher-dimensional problems or models where computing the model output would be too time consuming.

Latin Hypercube Sampling (LHS) aims to reduce the number of samples needed to observe convergence of statistics. To construct N samples, LHS divides the parameter space into N uniformly partitioned intervals, such that it samples pseudo-randomly in each interval and then ensures that the samples are spread across all dimensions. This results in better coverage of the parameter space with the need of fewer samples.

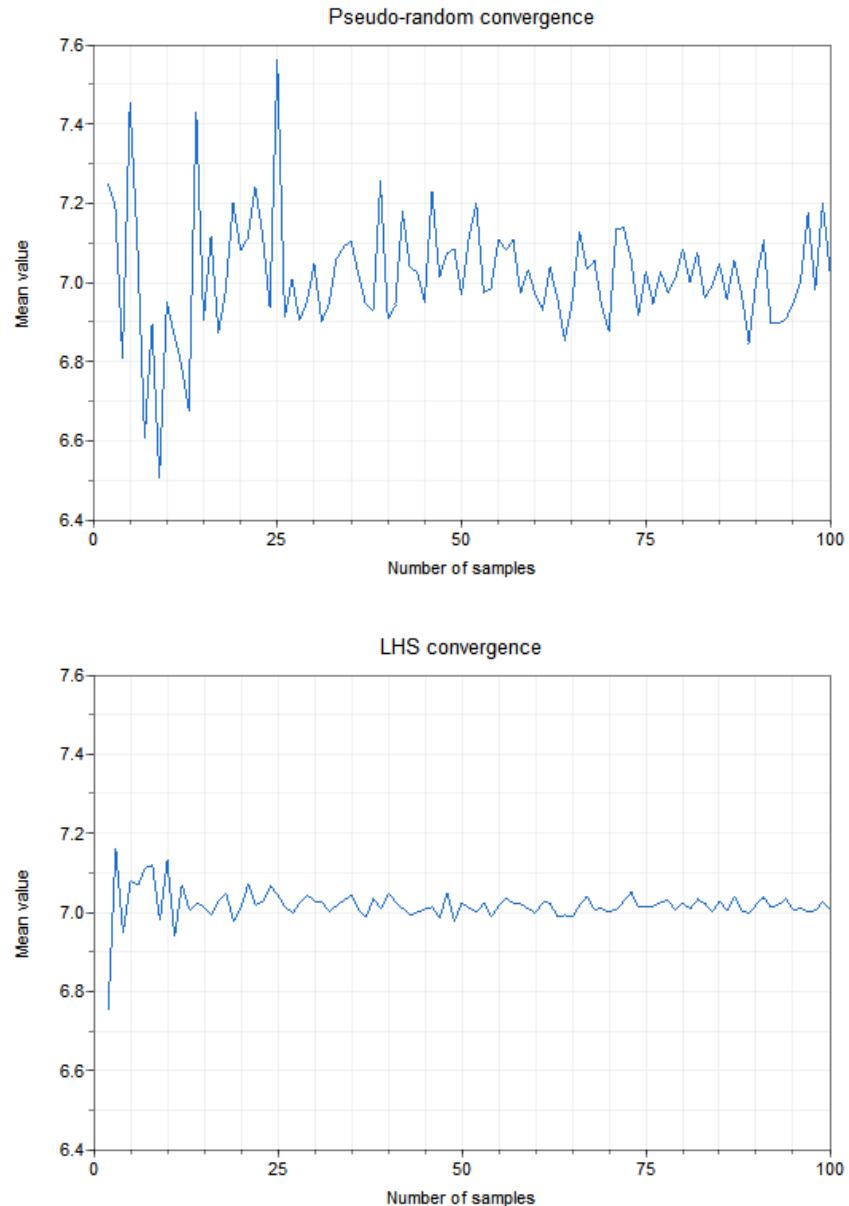
Illustration of a pseudo-random outcome of 5 samples on an interval:



Illustration of an LHS outcome of 5 samples on an interval:



For an example. We let two parameters of the Coupled Clutches model be random, the frequency **f** for clutch 1, and **J1**, the moment of inertia of clutch 1. The frequency **f** is modeled as a normally distributed variable with mean value 0.7 Hz, standard deviation 0.01 Hz, with a truncation on the minimum at 0.5 Hz. **J1** is modeled as a uniformly distributed variable between 1 kg m² and 2 kg m². Comparing the convergence analysis of the mean value of **J1.w** at time 1.5 s, it can be observed LHS converges about 10 times quicker for this example compared to pseudo-random sampling.

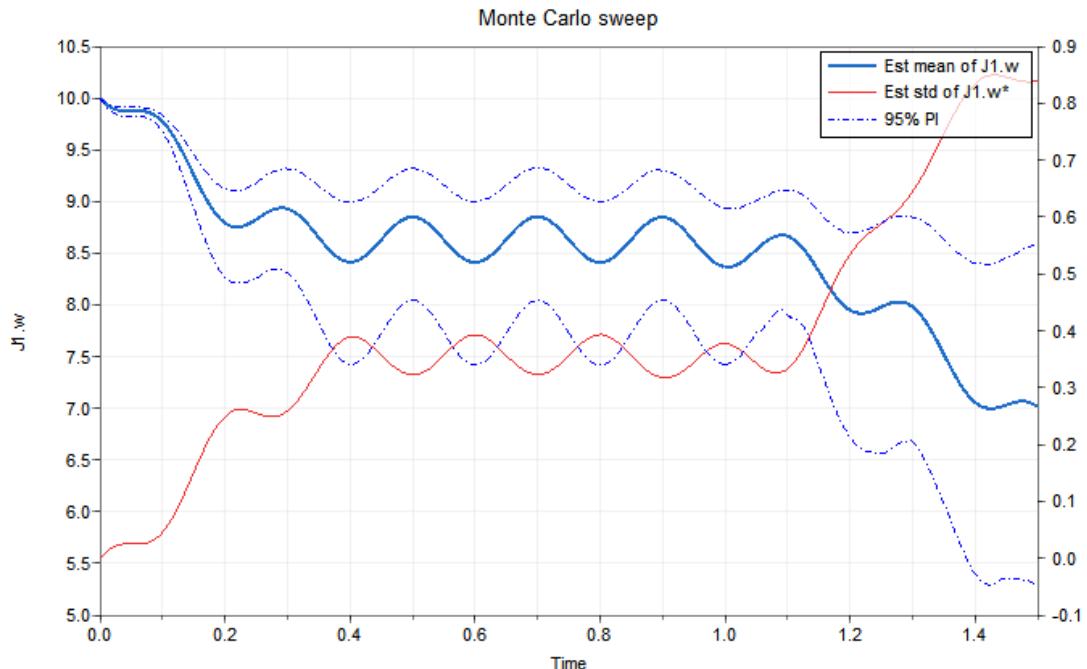


Analysis of the Entire Trajectory

Previously, Monte Carlo analysis only presented statistics on the end-point of simulations, summarizing results with metrics like mean value, standard deviation, histograms, and scatter plots. Now, the statistics of the entire trajectory are available.

Key new features include:

- Mean value and standard deviation over time
- Percentile curves (prediction intervals): For example, a 90% prediction interval is the range within which 90% of future simulation outcomes are expected to fall, based on the model's variability and uncertainty.

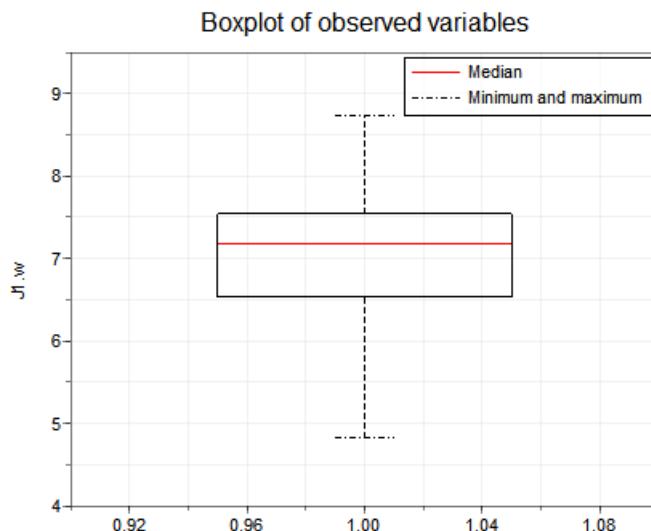
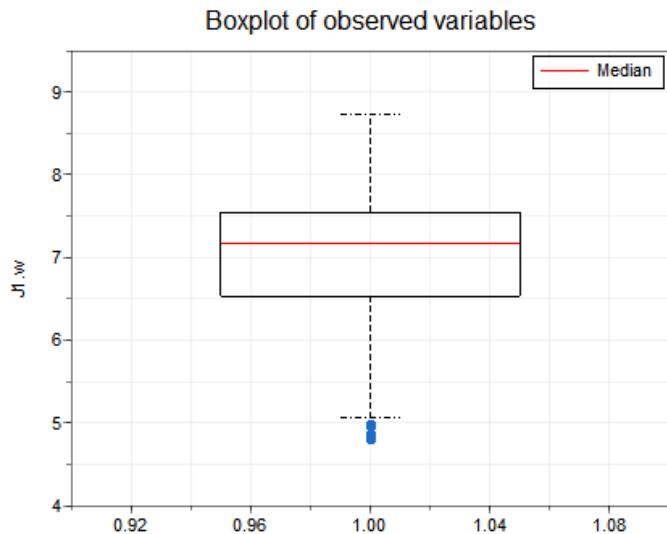


End-point analysis

Along with the mean value, standard deviation, histograms, and scatter plots, two options for box-plots of the observed variables are now available as well.

- Option 1: A detailed box plot showing the interquartile range (IQR), median in red, and outliers in blue dots for a clearer view of data spread and anomalies.
- Option 2: A simplified version with the box and whiskers representing the minimum and maximum values of the observed variables.

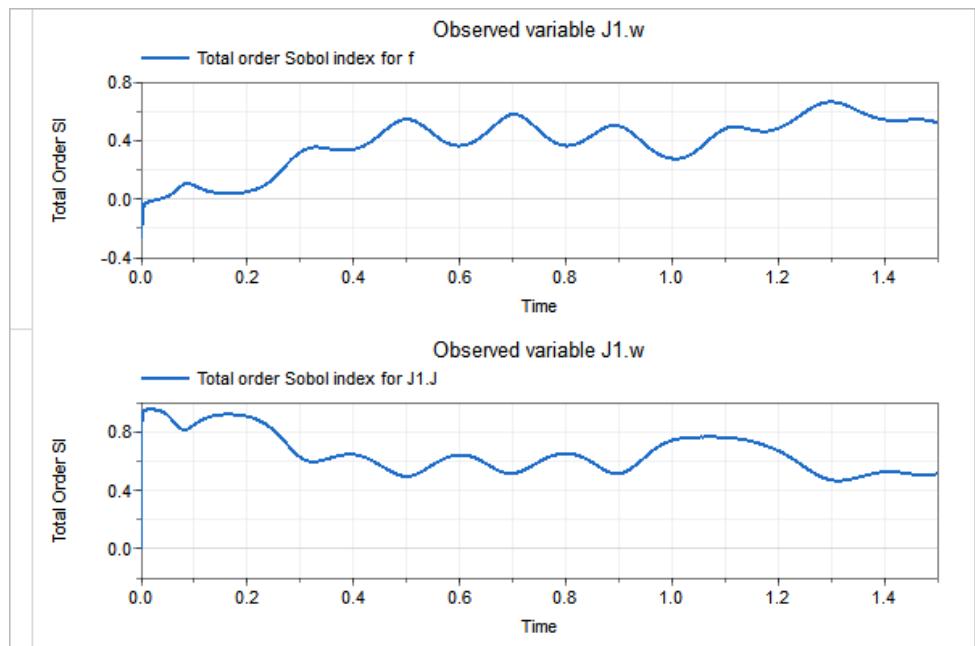
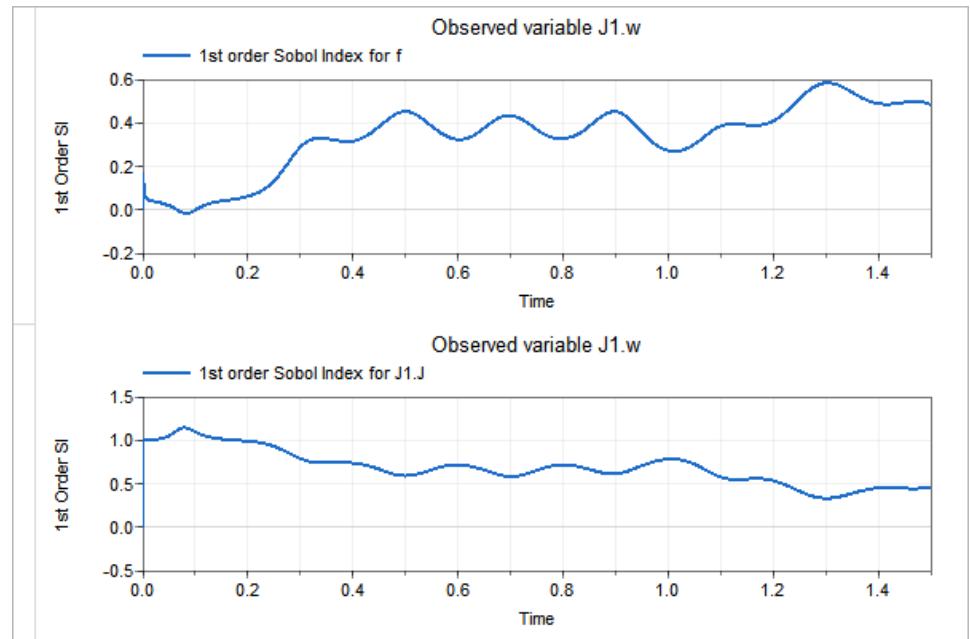
The box-plot aims to show the distributions of the data in a clear way. The main body of the box shows the central 50% of observed values, where the red line indicates the median. The whiskers are different depending on which option of box-plot is used. For option 1, any value that falls outside the whiskers is indicated as an outlier in a blue dot.



Global Sensitivity Analysis: Sobol Indices

It is now possible to perform Global Sensitivity Analysis on models, in the form of first- and total-order Sobol indices.

- First-order Sobol indices measure the proportion of output variance that can be attributed exclusively to variations in a single input parameter, holding all other parameters fixed.
- Total-order Sobol indices on the other hand measure the contributions to the output variance by including both its own effect and all interactions with other parameters.



Randomization package

A *User's guide* is provided for the package, but we state some core functionality here.

The Monte Carlo procedure is carried out by running the function `Randomization.Analysis.MonteCarloAnalysis`, which will return the sampled parameter values, resulting trajectories of the observed variables, as well as which simulations succeeded. The default for sampling is Latin Hypercube sampling (LHS), since this method usually gives faster convergence for a small number of uncertain parameters (less than 20). (The alternative is pseudo-random (PR) sampling.)

The following distributions are supported, with optional truncation:

- Uniform (PR, LHS)
- Normal (PR, LHS)
- Lognormal (PR, LHS)
- Pareto (PR, LHS)
- Beta (PR)
- Exponential (PR, LHS)
- Weibull (PR, LHS)
- Poisson (PR, LHS)
- Erlang (PR)
- Circular uniform (PR)

For processing the results given from `Randomization.Analysis.MonteCarloAnalysis`, consider running functions from the `Statistics` package.

The `Statistics` package includes functions for estimating:

- Confidence intervals for the mean
- Prediction intervals
- Percentile calculations
- Expected mean and variance
- Covariance

To carry out sensitivity analysis using Sobol indices, you can use the function `Randomization.Analyis.SensitivityAnalysis`. The method of estimating the indices is based on the Monte Carlo approach as explained in the book *Global Sensitivity Analysis: The Primer*, by Saltelli *et al.* The approximation of these indices can be quite numerically unstable so make sure to use enough samples. One sanity check is that the indices should not be negative (but negative values with small magnitude can be considered as 0). Please also note that the algorithm for estimating these indices requires three separate Monte Carlo runs compared to only one for regular uncertainty propagation. Together with the large magnitude of samples means that sensitivity analysis can take a while to process. Often the needed number of samples is more than 4000. For quicker simulation time, consider only last point simulation, and if confidence intervals are of interest, select a number for bootstrap samples, which will resample from the simulation to calculate mean value and variance of the indices.

These values are printed to the command log, and can be used, for example, in the `ConfidenceInterval` function.

It is also possible to run the function `Randomization.MonteCarloAnalysis` which will sample, simulate, process the data, and plot it, with the option of performing and plotting results of the sensitivity analysis as well.

3.6.2 Integrated GUI for Model Optimization

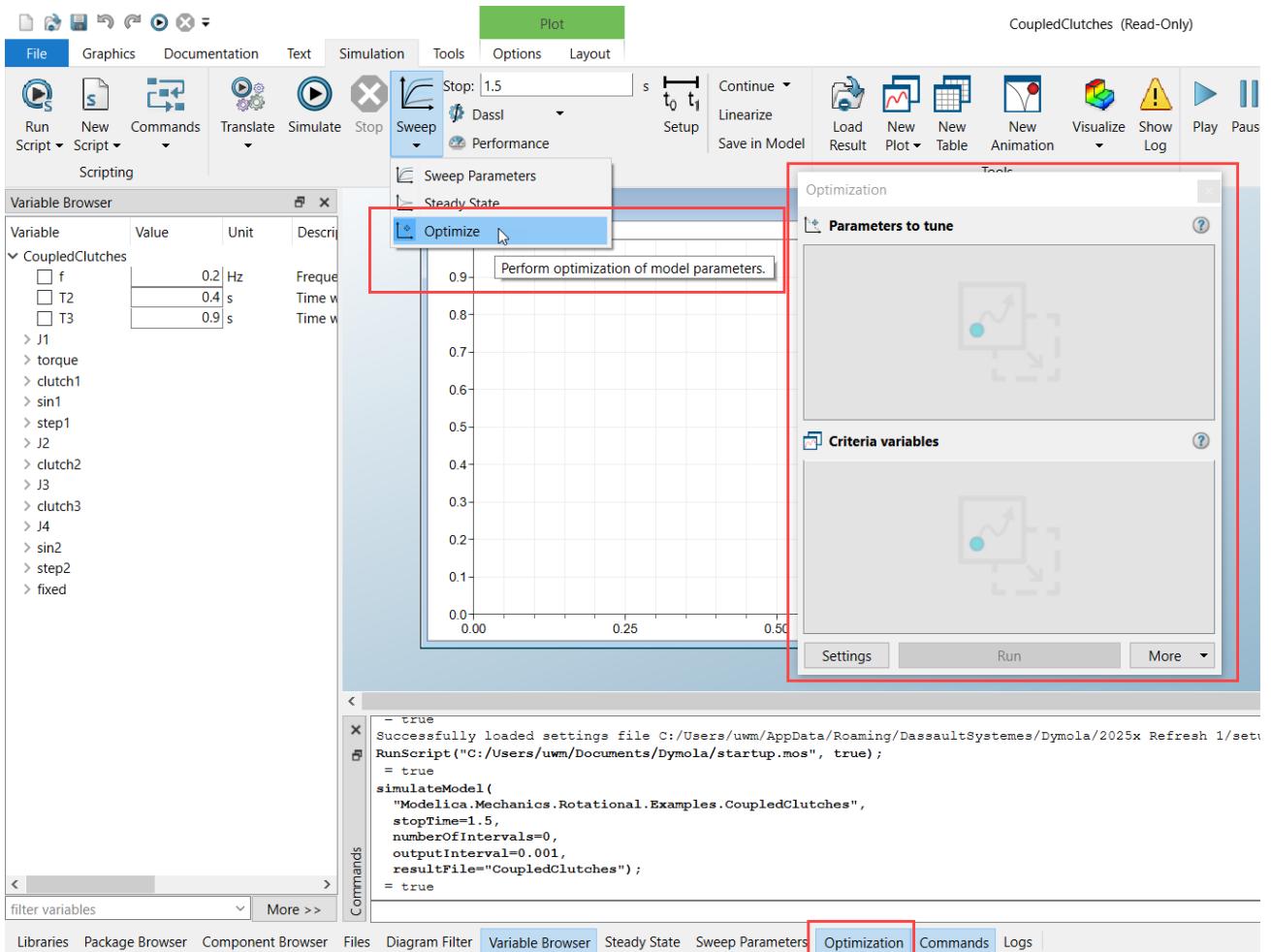
Introduction

An integrated GUI for model optimization is added. The basic look and feel is similar to the present GUI for sweeping.

Important:

- The integrated GUI is built on the model optimization from the Optimization library from DLR Institute of Vehicle Concepts. This means that with only the “Dymola standard” license you can run optimization with up to 3 tuners on a single core. For more tuners or multi-core optimization, you must have the Design Optimization (DOZ-x) license option.
- There are no new features implemented except the GUI functionality. Note that the functional GUI from the library can still be used, both for more detailed model optimization, but also for other advanced features, like multi-case optimization and trajectory optimization.

The integrated GUI for optimization is reached by a new command **Simulation > Sweep > Optimize**:



(You can also display the Optimization window by right-clicking in the empty space in the ribbon area and select **Optimization**.)

The Optimization window has two panes, one for parameters to tune and one for criteria variables. It is used the same as way as the sweep parameter window, after having simulated the model to study, you drag the parameters and criteria to each pane in the window, respectively.

For the parameters, you must specify a start value, and min/max bounds for each parameter.

For the criteria variables, you must specify a demand, which weights the criteria in the coming optimization run.

You can specify setting by pressing the **Settings** button in the Optimization window.

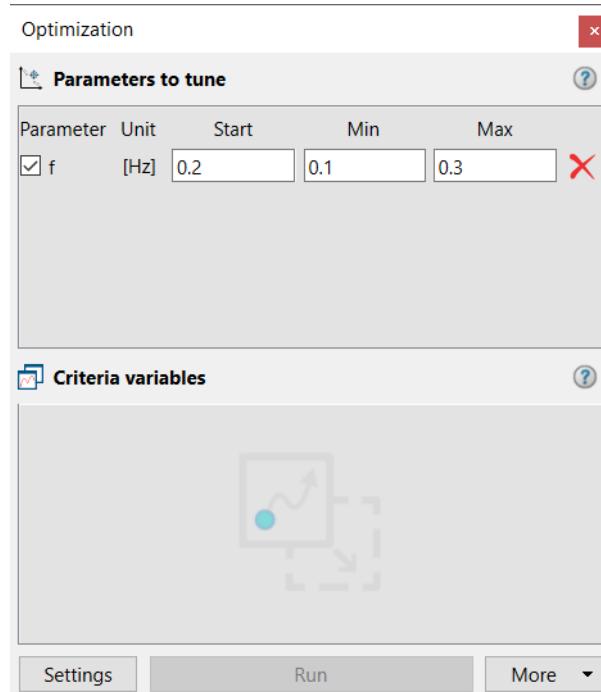
Pressing **Run** in the Optimization window executes the corresponding function from the Optimization library.

The result of the optimization is displayed as a report, and a number of files are saved.

Running an example

As an example, consider using the Coupled Clutches demo and tune the parameter **f** using the criteria variables **J1.w** and **J1.a**. To do that, do the following, in order:

- Use the command **File > Demos > Mechanical > Coupled Clutches** to open the Coupled Clutches demo, and the command **Simulation > Simulate** to simulate it.
- Use the command **Simulation > Sweep > Optimize** to open the Optimization window. It will look like the figure above.
- From the variable browser, drag **f** to the **Parameters to tune** pane in the Optimization window. You will get:



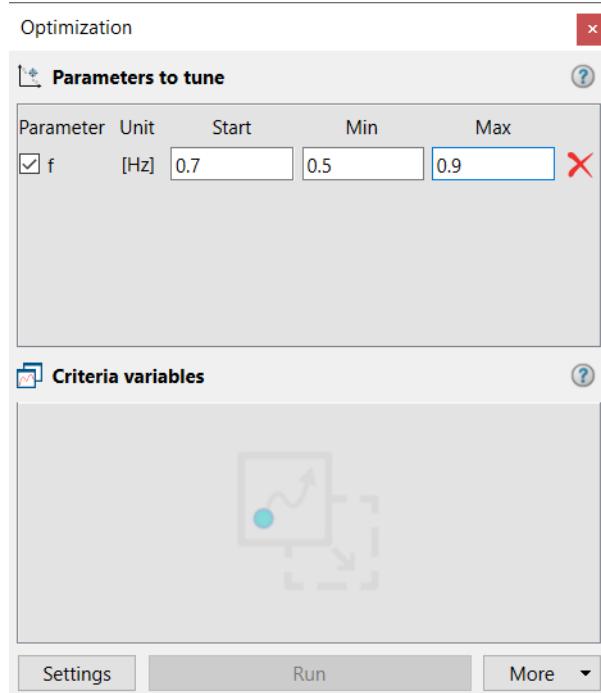
Start, **Min**, and **Max** values are by default prefilled. The start value is taken from variable browser, and min and max values are calculated from that value, as $-/+ 50\%$ of that start value.

Note that if you change unit, the values will be automatically rescaled, as an example, changing the above unit to rad/s will give:

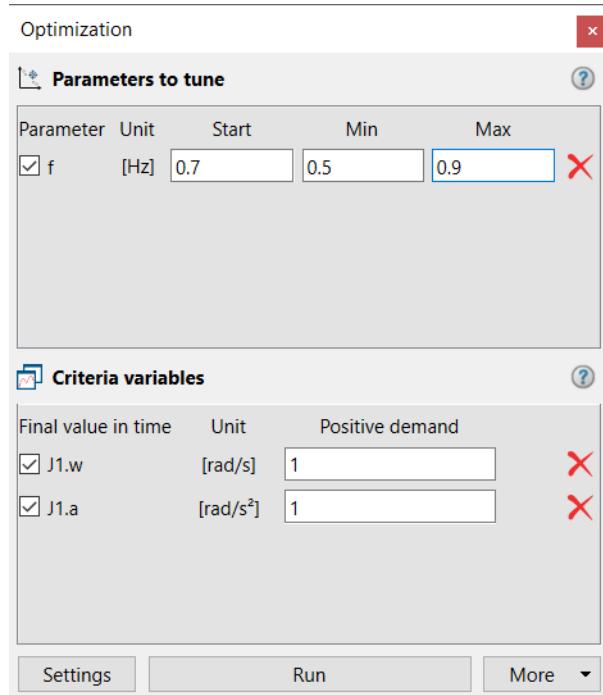
Parameter	Unit	Start	Min	Max
<input checked="" type="checkbox"/> f	[rad/s]	1.25664	0.628319	1.88496

However, this is for the option of looking at the values in other units. When you start optimization, the unit is reset, and the values are rescaled again.

- Keep Hz as unit, and specify the start value to 0.7, the min value to 0.5, and the max value to 0.9. You will get:

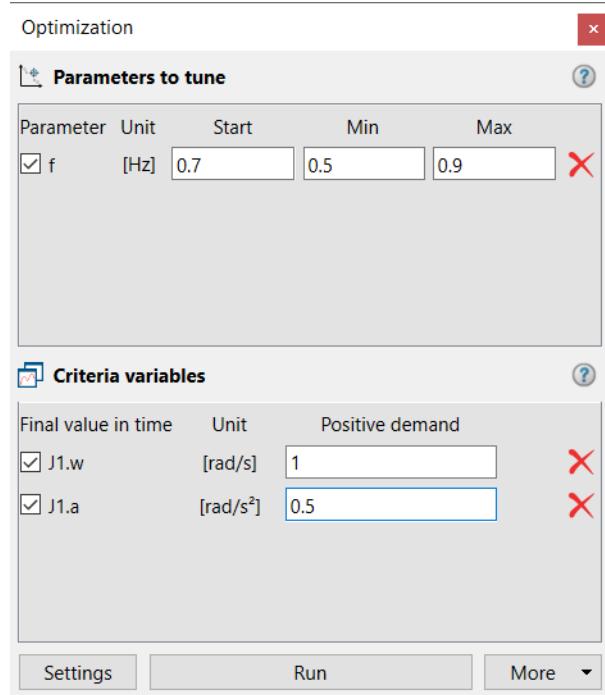


- From the variable browser, drag **J1.w** and **J1.a** to the **Criteria variables** pane. You will get:

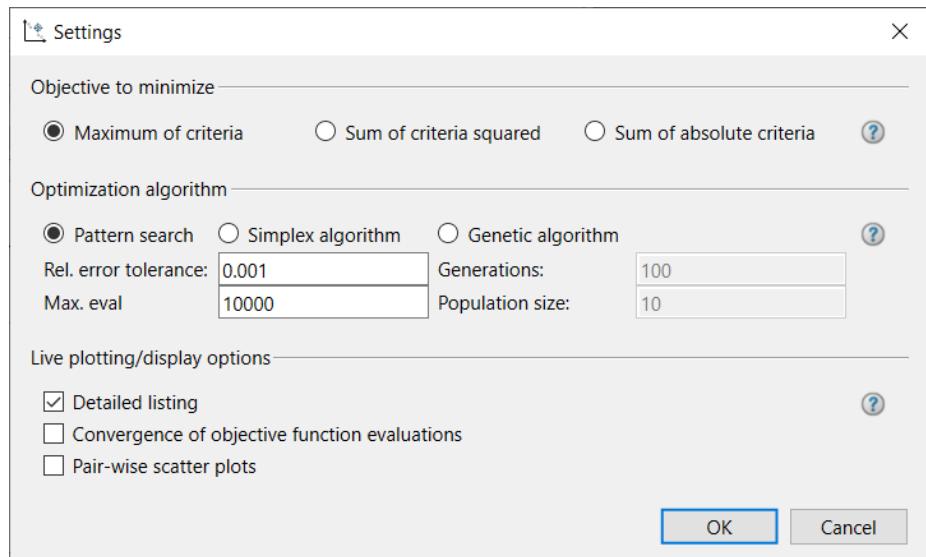


By default, the **Positive demand** (inverse weight) is prefilled as **1**. The unit widget corresponds to the unit of the demand value, which has the same unit as the criteria that was dragged in. Like for the parameters to tune, changing the unit will convert the demand value, but that is only as information in the user interface. **All values in the UI are automatically converted to their original units for the optimization.**

- Change the **Positive demand** (inverse weight) for J1.a to **0.5**. This will give:



- For this example, you can keep the default value for the settings, that is, clicking **Settings** will give you:

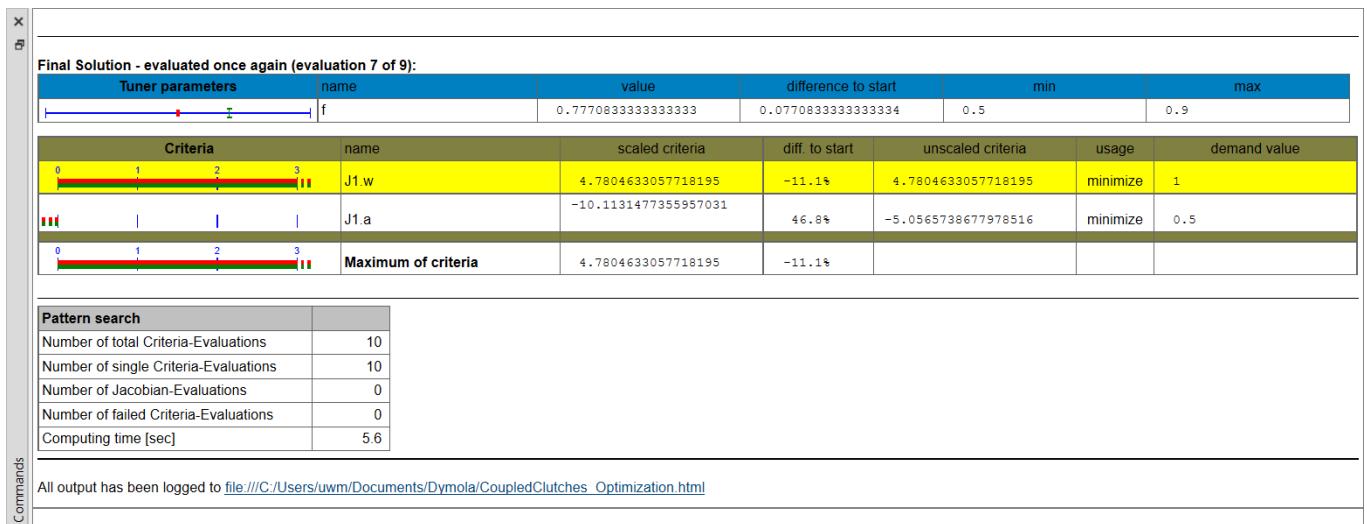


Note that clicking the question marks will display documentation for the corresponding group.

Two options for plotting are available if **Detailed listing** is activated:

- **Convergence of objective function evaluations** – Plots the current objective function evaluation obtained from listing during the optimization run.
- **Pair-wise scatter plots** – Pair-wise scatter plots displaying the path of the tuners and objective function evaluation for each current best calculation.
- Press the **Run** button to perform the optimization. When the optimization is finished, you will get the following result:

In the Commands window, you will get the end of the optimization report, looking like this:



The screenshot shows the Dymola Optimization window with the following content:

Final Solution - evaluated once again (evaluation 7 of 9):

Tuner parameters	name	value	difference to start	min	max
	f	0.7770833333333333	0.0770833333333334	0.5	0.9

Criteria

Criteria	name	scaled criteria	diff. to start	unscaled criteria	usage	demand value
0 1 2 3	J1.w	4.7804633057718195	-11.1%	4.7804633057718195	minimize	1
	J1.a	-10.1131477355957031	46.8%	-5.0565738677978516	minimize	0.5
0 1 2 3	Maximum of criteria	4.7804633057718195	-11.1%			

Pattern search

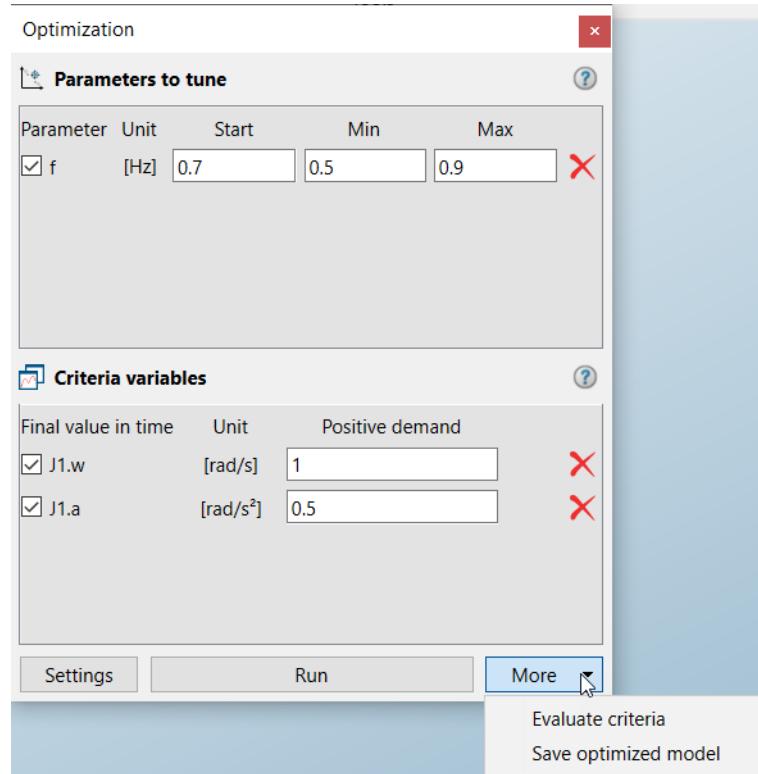
Pattern search	
Number of total Criteria-Evaluations	10
Number of single Criteria-Evaluations	10
Number of Jacobian-Evaluations	0
Number of failed Criteria-Evaluations	0
Computing time [sec]	5.6

All output has been logged to file:///C:/Users/uwm/Documents/Dymola/CoupledClutches_Optimization.html

The last line tells you where to get the full report in HTML text. Click this link to get the full report.

Additional options in the Optimization window

For the Optimization window, there are two additional options; pressing **More** will display these options:



The options are:

- **Evaluate criteria** – Evaluates the criteria and objective function with the given start values for the tuners, effectively performing one simulation.
- **Save optimized model** – Performs two actions:
 - Saves the best tuner values for the model from the last optimization run. Note that the file `OptimizationBestModel.mo` (which is created after an optimization run, see below) must exist in the current file directory.
 - Displays a dialog to choose a name for the optimized model as well as where to store it. This option will take the file `OptimizationBestModel.mo`, rename it, and insert it into the chosen package. If no package is chosen, it will save the model in the top-level of the project.

Files created from an optimization

An optimization run will create the following files in the working directory:

Modelica files (.mo):

- The full setup for the optimization, including what parameters to tune and the criteria variables, is saved in a file <modelName>_OptimizationSetup.mo. (For the example above, the file name will be CoupledClutches_OptimizationSetup.mo.)
- The model with the best optimization result included is saved as OptimizationBestModel.mo. **Note.** From the Optimization window, you can use the command **More > Save optimized model** to rename this file and insert in a selected package. Please see this command above.

Text files (.txt)

Note that the below files contain matrices that can be read in Dymola using, for example, the function call `readMatrix`.

- The best tuner and criteria values are saved in OptimizationBest.txt.
- All tuner values with criteria are saved in OptimizationHistory.txt.

HTML files (.html)

The HTML log of the run is saved as <modelName>_OptimizationSetup.html.

Controlled optimization interruption

When you run an optimization, the button **Run** in the Simulation window is exchanged to a **Finish** button. Clicking this button conclude the optimization in-between criteria evaluations and save all relevant files. Note the difference to the **Stop** button in the Simulation ribbon, that button will terminate the optimization in the middle of a simulation, and files will not be created.

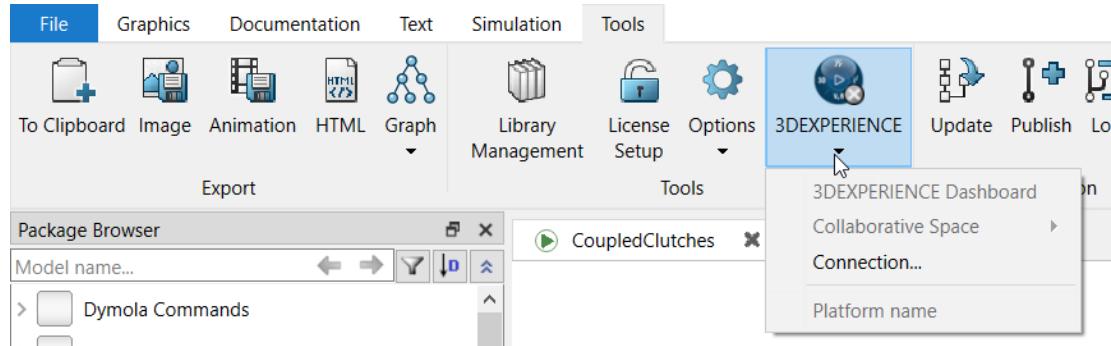
3.7 Model Management

3.7.1 Version management: Improvements in the 3DEXPERIENCE app “Design with Dymola”

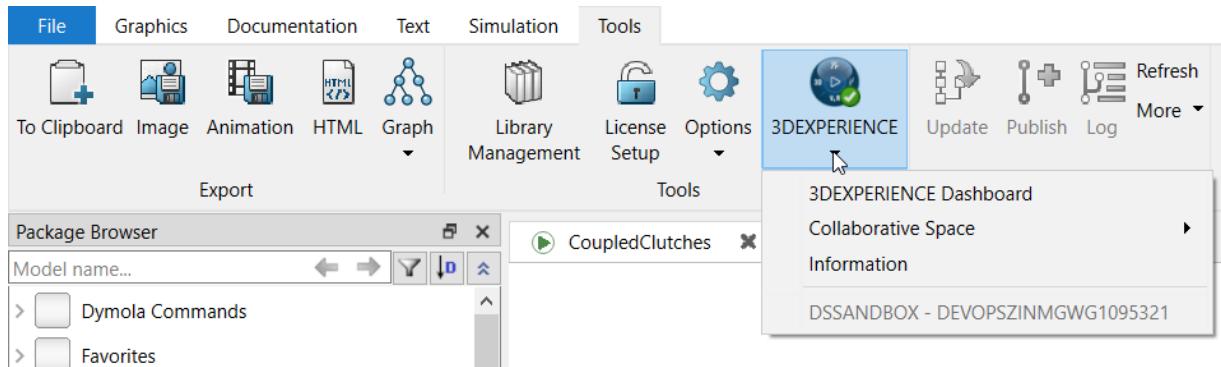
3DEXPERIENCE command in Tools ribbon

The Tools ribbon has been enhanced with a 3DEXPERIENCE command, with a number of sub-commands. These commands are duplicates of the corresponding context commands, to have them available from the ribbon as well.

If Dymola is not connected to the 3DEXPERIENCE platform, the command looks like:

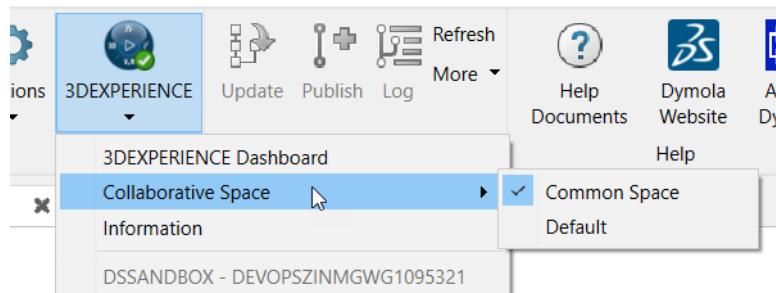


If Dymola is connected to the 3DEXPERIENCE platform (you can connect by clicking the **Connection...** command above), the command looks like:

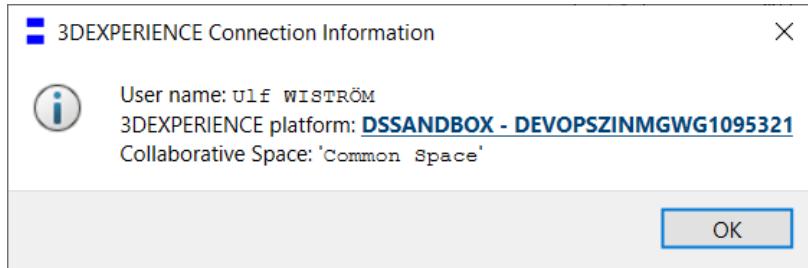


The contained commands are:

- **3DEXPERIENCE Dashboard** - opens the corresponding 3DEXPERIENCE dashboard.
- **Collaboration Space** – to select the collaboration space to use:



- **Connection...** - connect to the 3DEXPERIENCE database. You have to supply user name and password to connect.
- **Information** – displays information about the connection (when connected):

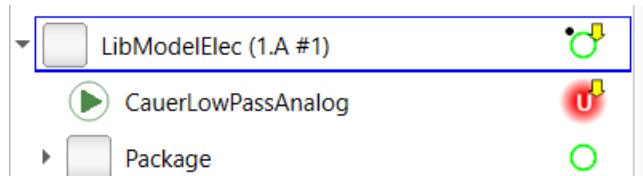


- In the last part of the command box, information about the platform name is given.

Indication in the package/project browser that a local model is not the last iteration

If you have checked out a model, and work on that one, someone else may publish changes of the same file from another machine. In that case, you will have indications in the package that the file you work on locally is not from the latest iteration in the database.

You will have an indication on both the top-level package and the model. In the following example, you have checked out the model CauerLowPassAnalog and worked with it locally, but now somebody else has published a new iteration in the database.



On the top-level package, you now have a yellow arrow telling you that one or more models in the package is not from the latest iteration in the database. The model CauerLowPassAnalog that is now not from the latest iteration in the database has a yellow arrow and a red icon with a "U" (Unmerged).

Note that you can get this situation in other cases as well; another example is when you open an old locally saved model that is not from the latest iteration in the database, and right-click it and select **Version > Checkout**.

If any of the above happens, you have two choices:

- To merge the latest iteration in the database with your locally saved file. To do so, right-click the model in the package browser and select **Version > Update**. This will open the merge tool that will help you perform the merge. The merged file will be opened in your package browser. If you have no merge tool configured, a merged file will be saved locally with a specific suffix, in this case `CauerLowPassAnalog.mo~` and the latest iteration in the database will be opened in the package browser. A message about a detected conflict will appear in the console, in the versioning tab. Any further merging work here has to be done manually. For this reason, it is strongly recommended that you configure a merge tool.

- To reject your local changes and use the new iteration in the database, right-click the model in the package browser and select **Version > Revert**.

Generic login with “Remember me” option

When you start Dymola and connect to the 3DEXPERIENCE platform, you now get a generic Dassault Systèmes login dialog. You enter user name and password as before, but you also have a “Remember me” option that you can activate.

If you activate this option, it means two things:

- If you close Dymola, you can open it again and connect to the 3DEXPERIENCE platform without having to enter user name and password, until 24 hours after the previous login.
- If you run your session for a long time, you have to login again after 24 hours. If you have not activated the “Remember me” option, you will have to log in after a shorter time.

Open libraries by drag and drop from 3DEXPERIENCE search results

From Dymola 2025x Refresh 1, you can drag a library saved by Design with Dymola from a 3DEXPERIENCE platform web page, typically a page containing search results, to the package browser in Dymola to open it there.

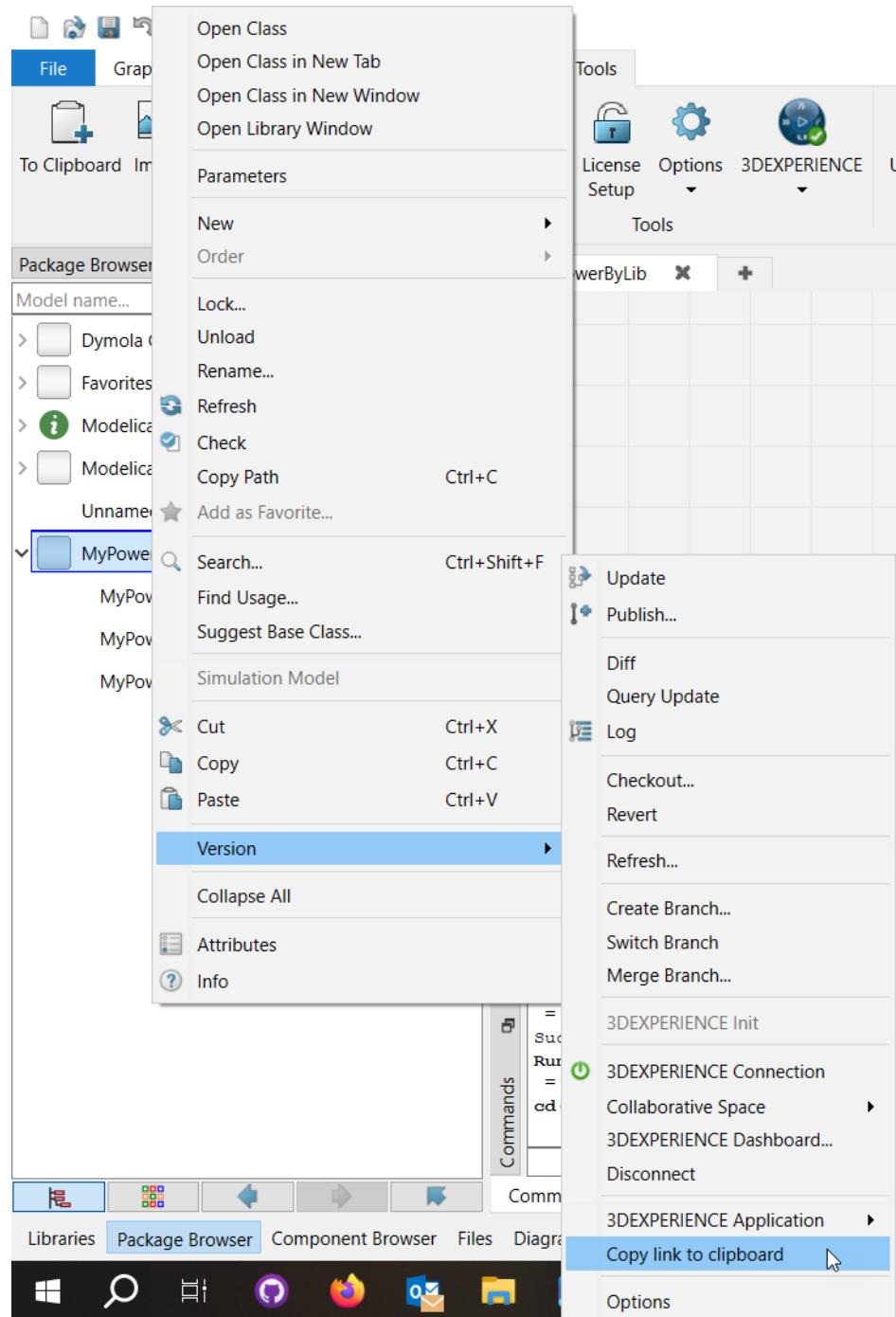
If Dymola is not connected to the 3DEXPERIENCE platform, you will get a login dialog to connect. If Dymola has never been connected to the 3DEXPERIENCE platform, you will get a message that you must start Design with Dymola from the 3DEXPERIENCE Compass once to configure it.

Open Design with Dymola from a search result

You can start Design with Dymola from a 3DEXPERIENCE search result file created by Design with Dymola, by right-clicking it and selecting **Open With > Design with Dymola**. This was possible also in previous Dymola versions. Note that you need to have a license for Design with Dymola to open it.

Creating a link to a selected model

To create a link to a Design with Dymola model, you can right-click the model in the package browser and select the command **Version > Copy link to clipboard**:

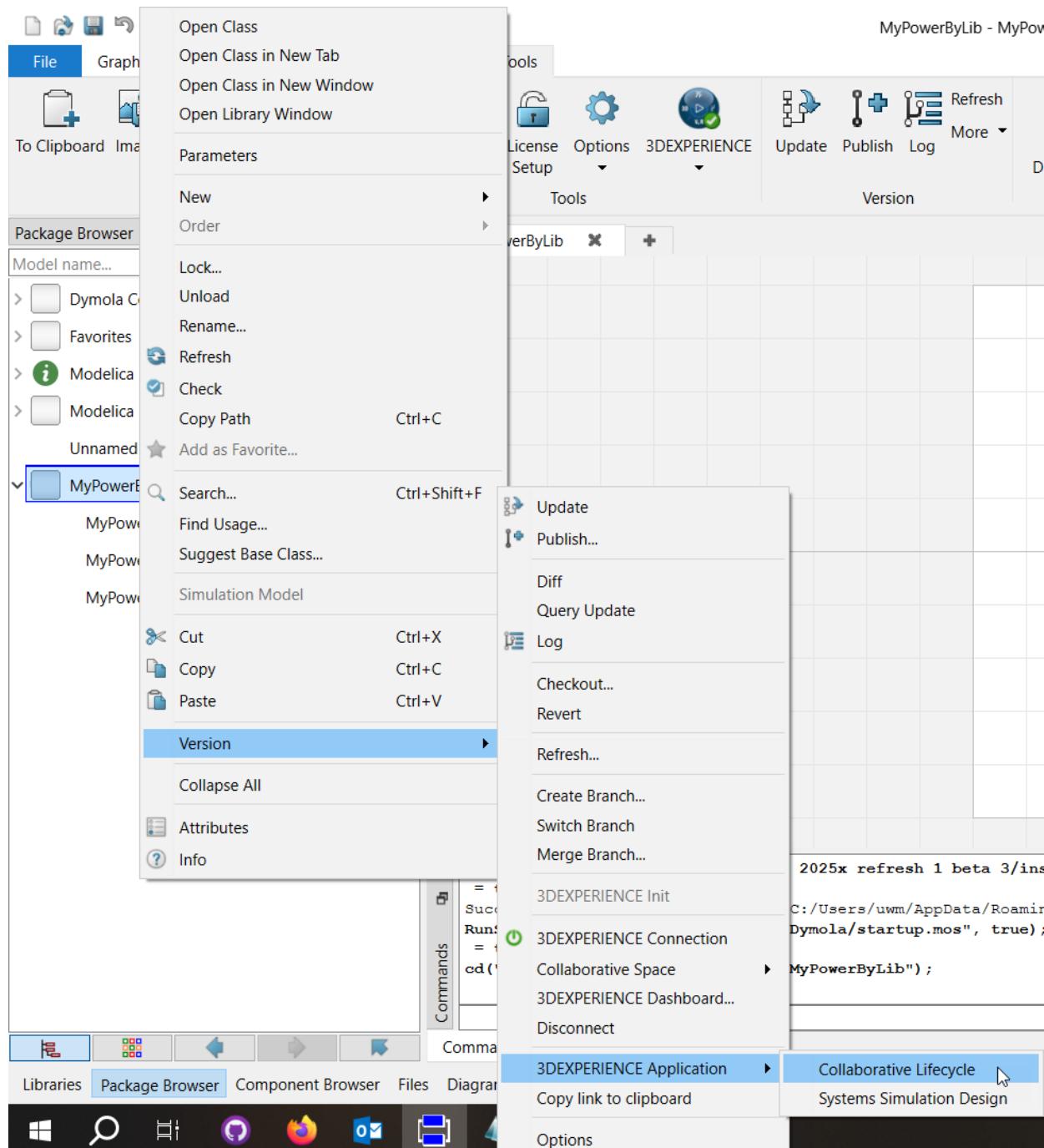


You can paste the link into, for example, a Microsoft Word document or a browser.

Note. When you use the link to open the model, you must have license for Design with Dymola, as well as access right to the model itself.

Opening a selected model in the 3DEXPERIENCE app “Collaborative Lifecycle” (SIA)

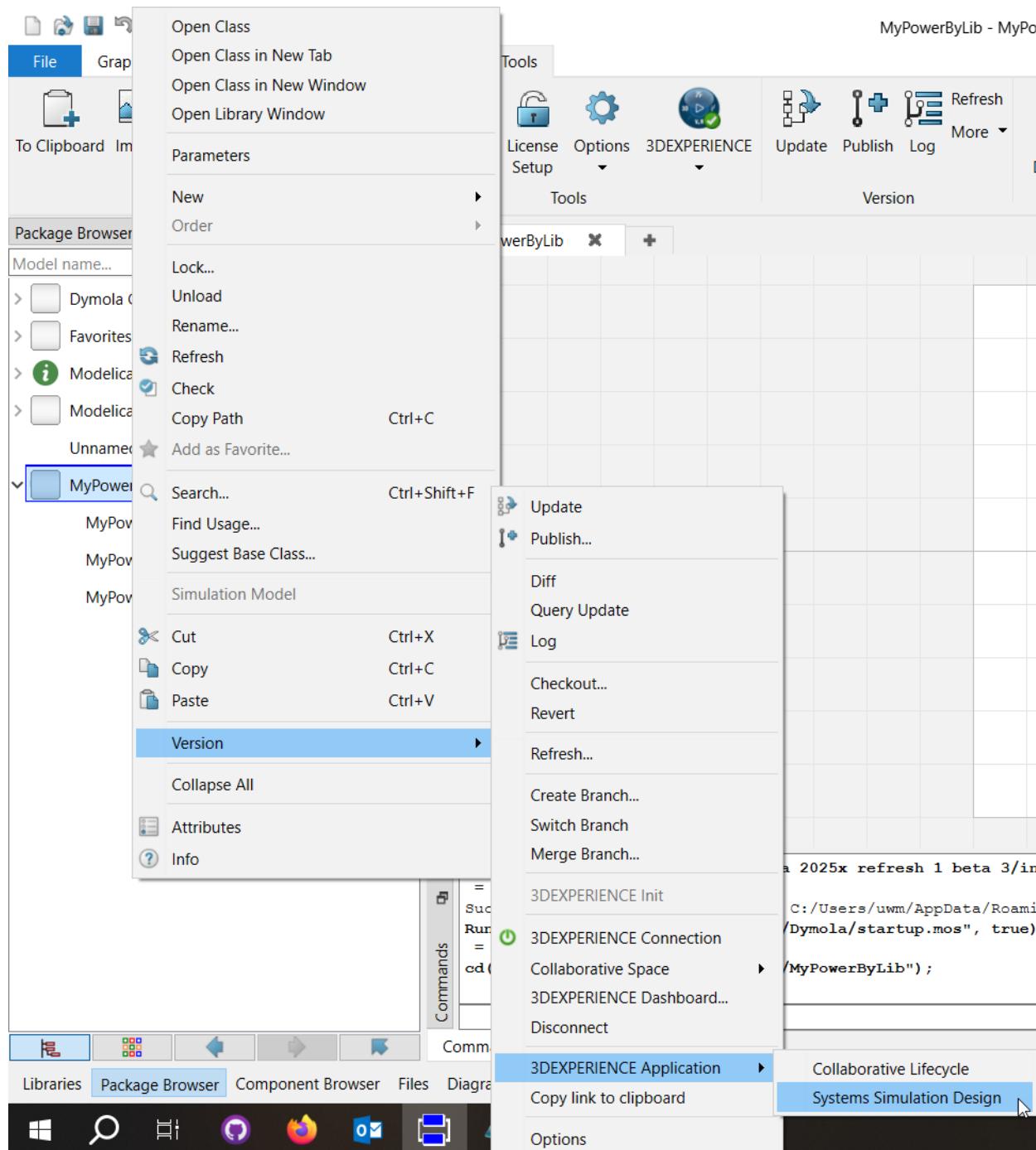
You can open a selected model in the 3DEXPERIENCE app “Systems Simulation Design” (SIA). You can do that by right-clicking a model in the package browser and then selecting the command **Version > 3DEXPERIENCE Application > Systems Simulation Design**:



Note. You must have a license for “Collaborate Lifecycle” to use this command.

Opening a selected model in the 3DEXPERIENCE app “Systems Simulation Design” (SIA)

You can open a selected model in the 3DEXPERIENCE app “Systems Simulation Design” (SIA). You can do that by right-clicking a model in the package browser and then selecting the command **Version > 3DEXPERIENCE Application > Systems Simulation Design**:



Note. You must have a license for “Systems Simulation Design” to use this command.

3.7.2 Model structure: Improved model editing API

Introduction

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

New AST functions

- New examples (in the **Examples** subpackage):

Name	Description
<code>exportModelicaTypes</code>	<p>This function will print a list of all Modelica connectors and types to a file. The information (in CSV format) includes name, description, and various attributes including connector causality, quantity and unit. The application here has been to export Modelica types in a format that “makes sense” in an SSP context, which is evident in the use of “kind” to designate causality and “unspecified” to mean acasual connector.</p> <p>As an example of the <code>ModelManagement</code> library, it applies several functions to recursively extract classes, process each one to get information and attributes, and print the information to file.</p>
<code>setupExperiment</code>	This function copies an existing model (so we can modify it) and then modifies the experiment annotation. It should be noted that any existing top-level model called “simModel” will be deleted.

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

Name	Description
<code>ExtendsModifierString</code>	Returns the value of a string attribute of a class, including the cases where the attribute is defined in a base class.
<code>GetAnnotationBoolean</code>	Returns the Boolean value of the specified annotation. If the annotation is missing, the provided default value is returned.
<code>GetAnnotationReal</code>	Returns a Real number containing the value of the specified annotation. If the annotation is missing, the provided default value is returned.
<code>Reformat</code>	Removes any existing formatting and applies the default Dymola heuristics for formatting Modelica models.

Save	Saves a Modelica model that has been read from a file.
SetAnnotationBoolean	Sets the annotation to the given Boolean value.
SetAnnotationReal	Sets the annotation to the given Real value.
SetAnnotationString	Sets the annotation to the given String value.

- For editing components in a model (in the **Components** subpackage):

Name	Description
DeleteComponent	Deletes a component and by default any connections in that class going to or from the component. It does not search for connections in derived classes.

- For editing connections (in the **Connections** subpackage):

Name	Description
SetConnectionRoute	Sets the route (points of the line) of a connection between two connectors in the specified model. The designated connectors may be either top-level connectors or connectors of a component (using dot-notation).

- For editing equations (in the **Equations** subpackage)

Name	Description
DeleteEquations	Deletes equations in a Modelica class. Optionally graphical equations can be deleted too (they are technically equations) but by default only equations that are not connections and non-graphical connections are deleted. State-machine transitions are treated as connections.

- Miscellaneous functions (in the **Misc** subpackage):

Name	Description
AllClassesInPackage	This function returns all classes, recursively anywhere in the substructure. Protected classes are not included. For other applications where you may want classes of a particular kind, this function can be duplicated and modified as needed.

SplitModelicaPath	Returns the individual identifiers that make up a Modelica path separated by “.” (dot). This function handles quoted Modelica identifiers (QIDENT) correctly.
-------------------	---

Changed AST functions

- For editing components (in the **Components** subpackage):

Name	Description of change
SetComponentType	Removed default for the input justSet.

3.8 Other Simulation Environments

3.8.1 Dymola – Scilab interface

In previous Dymola versions, you could import result MAT-files issued by Dymola simulations (e.g. dsres.mat) and post-process them in Scilab using similar functions as “Matlab Dymtools”.

In Dymola 2025x Refresh 1, Dymola can also be run and driven from Scilab using its Java API wrapped into Scilab. You can use Dymola Java API functions (model parametrization, simulation, etc.).

3.8.2 Dymola – Matlab interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2020a (ver. 9.8) up to R2024b (ver. 24.2). 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.

3.8.3 Real-time simulation

Compatibility – dSPACE

Dymola 2025x Refresh 1 officially supports the DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2025x Refresh 1 generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases:

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

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

New utility functions – `dym_rti_build2` and `dym_rtimp_build2`

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

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

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

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

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

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

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

Compatibility – Simulink Real-Time

Compatibility with Simulink Real-Time has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R2020a (Simulink Real-Time ver. 6.12) to R2024b (Simulink Real-Time ver. 24.2). Only Microsoft Visual C compilers have been tested.

3.8.4 Java, Python, and JavaScript Interface for Dymola

Modernized JavaScript interface and report generator

The JavaScript interface and the report generator have been modernized. The web browsers Chrome, Firefox, and Edge are supported. The JavaScript interface and the report generator are supported on Linux as well.

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 15.

3.8.5 SSP Support in Dymola

Support of array components

Dymola 2025x Refresh 1 supports SSP 2.0, including support for arrays of connectors and array of parameters (a special kind of connector in SSP). Exporting a Modelica model that contains such arrays is supported.

As an extension to SSP 2.0, Dymola also supports arrays of components; this is an extension to the SSP 2.0 specification. When such an array is exported from Modelica to SSP, Dymola will issue a warning.

SSP Meta data to create model commands

To be able to create model commands, for example, to pre-define plotting or testing scripts, in SPP we need to store them in some convenient format, for example, in SRMD meta data.

The supported meta data keys are divided into two groups. The first group include one or more setup commands to define the **Commands** menu entry.

Key	Command Setup
dymola.command.title	Title of command, used as command name in the Dymola Commands menu.
dymola.command.group	For grouping commands in sub-groups [optional].
dymola.command.caption	Used as tooltip (explanation) in the command menu [optional].
dymola.command.auto	If true, plot is created automatically after simulation [optional].
dymola.command.translated	If true, model must be translated to execute command [optional].
dymola.command.simulated	If true, model must be simulated to execute command [optional].

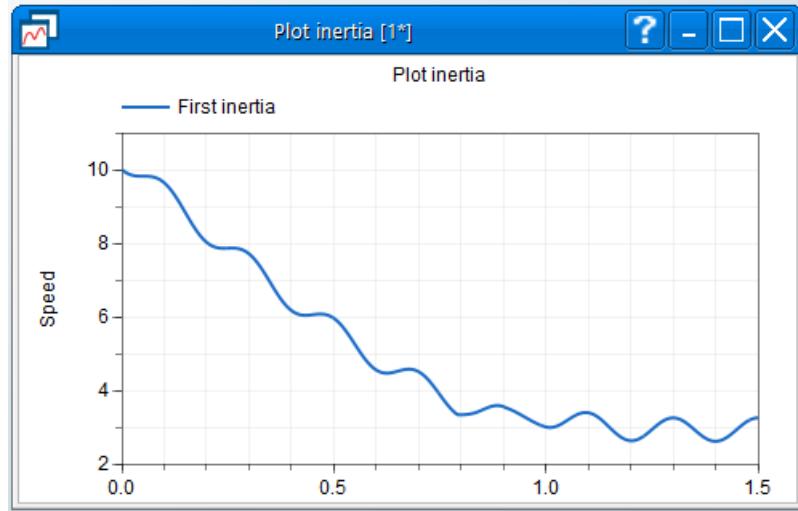
After the setup follows one of the below actions that defines what the command entry should do. This is the second group.

Key	Action
dymola.command.figure	A Modelica figure annotation specifying the plot contents. All figure commands imply “simulated”. For example: <pre>{Plot(curves={Curve(x=time, y=J1.w, Legend="First inertia")}, y=Axis(label="Speed"))}</pre>
dymola.command.script	URI of a script file. For example, <pre>modelica://Modelica/Resources/Scripts/Dymola/Mechanics/Rotational/CoupledClutches.mos</pre>
dymola.command.execute	Execute arbitrary command. For example, <pre>cd()</pre>
dymola.command.open	Open a Modelica model in a new tab, for example, a test model, <pre>Modelica.Electrical.Analog.Examples.CaurLowPassAnalog</pre>

As example, a meta data file `Dymola Commands.smrd`, used in a Coupled Clutches model, with the following content:

Key	Description
dymola.command.title	Plot inertia
dymola.command.caption	Plot first inertia velocity
dymola.command.auto	true
dymola.command.figure	<pre>{Plot(curves={Curve(x=time, y=J1.w, legend="First inertia")}, Y=Axis(label="Speed"))}</pre>

gives the plot:



Option to add suffix for imported SSP or SSD file names

When an SSP or SSD file is imported, Dymola creates a new package. By default, this package is created at the top level of the existing package hierarchy, which sometimes causes naming conflicts. For that reason, you can now set a suffix that is added to all created top-level packages. The variable is called `Advanced.SSP.TopLevelPackageSuffix`.

Example:

```
Advanced.SSP.TopLevelPackageSuffix = "_SSD";
```

The value of the flag is stored between sessions. The default value is an empty string (no suffix).

Unit conversion when importing an SSP, SSD, or SSV file

When importing an SSP, SSD or SSV file, unit conversions defined in the file are applied because Modelica assumes values are always expressed in the underlying SI unit. To disable unit conversions, for example if they are inconsistently used, set the flag:

```
Advanced.SSP.ApplyUnitConversion = false
```

(The flag is by default `true`.)

MAP-COORD support when importing CSV files by the ImportSSV built-in function

The built-in function `importSSV` supports importing CSV files according to the proposed Modelica Association standard, see also Modelica Association MAP-COORD. Imported CSV files may contain a header row. CSV files without header row are also supported.

3.8.6 FMI Support in Dymola

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

FMI Export: FMU input interpolation for FMI 3 Co-simulation FMUs

Introduction

Dymola 2025x Refresh 1 supports optional FMU input interpolation for Co-simulation FMUs. By smoothing any continuous-time Real input to the FMU, the integrator is allowed to continue the simulation without any reset. The result is potentially improved efficiency. To further help the integrator when smoothing inputs, you can also activate predictor compensation.

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

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

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

The features are also available for FMU source code export.

Important:

- For FMI 3, if using intermediate update mode, an FMU may enter intermediate update mode (in order to exchange input and output values) between communication points. For this reason, it is not possible to set inputs with intermediate update mode when the input smoother is enabled. (If tried, the input won't be set and the input smoother will work like usual.) However, retrieving outputs is still allowed.
- These features are already available for FMI version 2, from previous version.

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

Input smoother

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

With the flag activated, exported Co-simulation FMUs with any of the solvers specified in the item list in the introduction above will include the input smoother. This means that any continuous-time Real input will be smoothed. Discrete inputs like Integers, Booleans, and discrete-time Real inputs are not affected by the input smoother, but are handled normally.

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

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

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

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

Predictor compensation

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

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

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

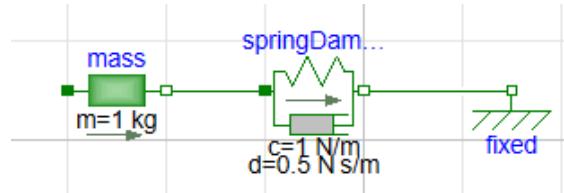
To alleviate the problem, the integrator predictor can be modified using predictor compensation. The improved predictor results in better error estimates and helps in solving for the next integrator step. In general, we may see even larger step-sizes, even fewer model evaluations, and even fewer Jacobian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

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

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

Test case

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



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

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

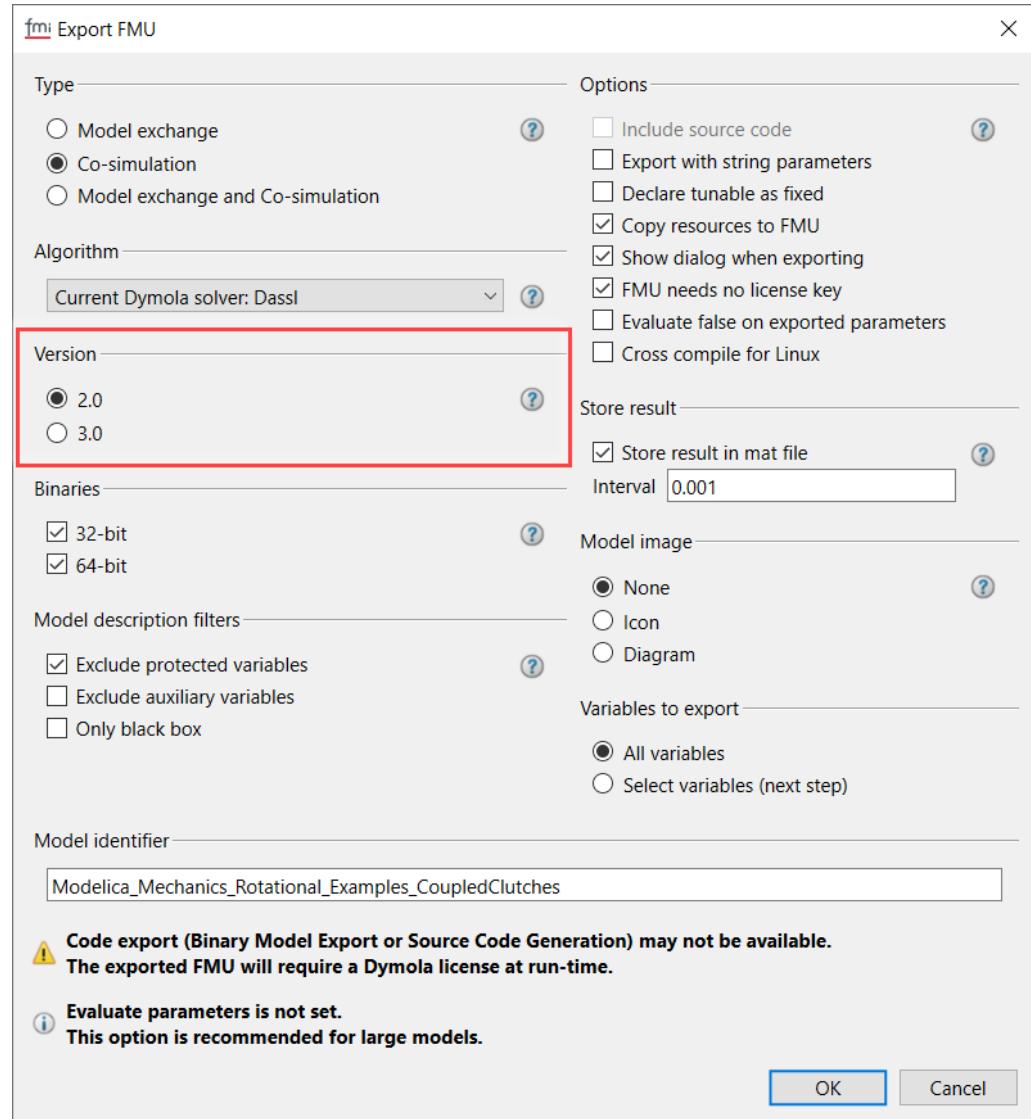
Feature	Number of f-function evaluations	Number of Jacobian-evaluations	Predictor compensation f-function evaluations
Normal Co-simulation	5471	499	-
Input smoother	2708	40	-
Input smoother and predictor compensation	853	14	998

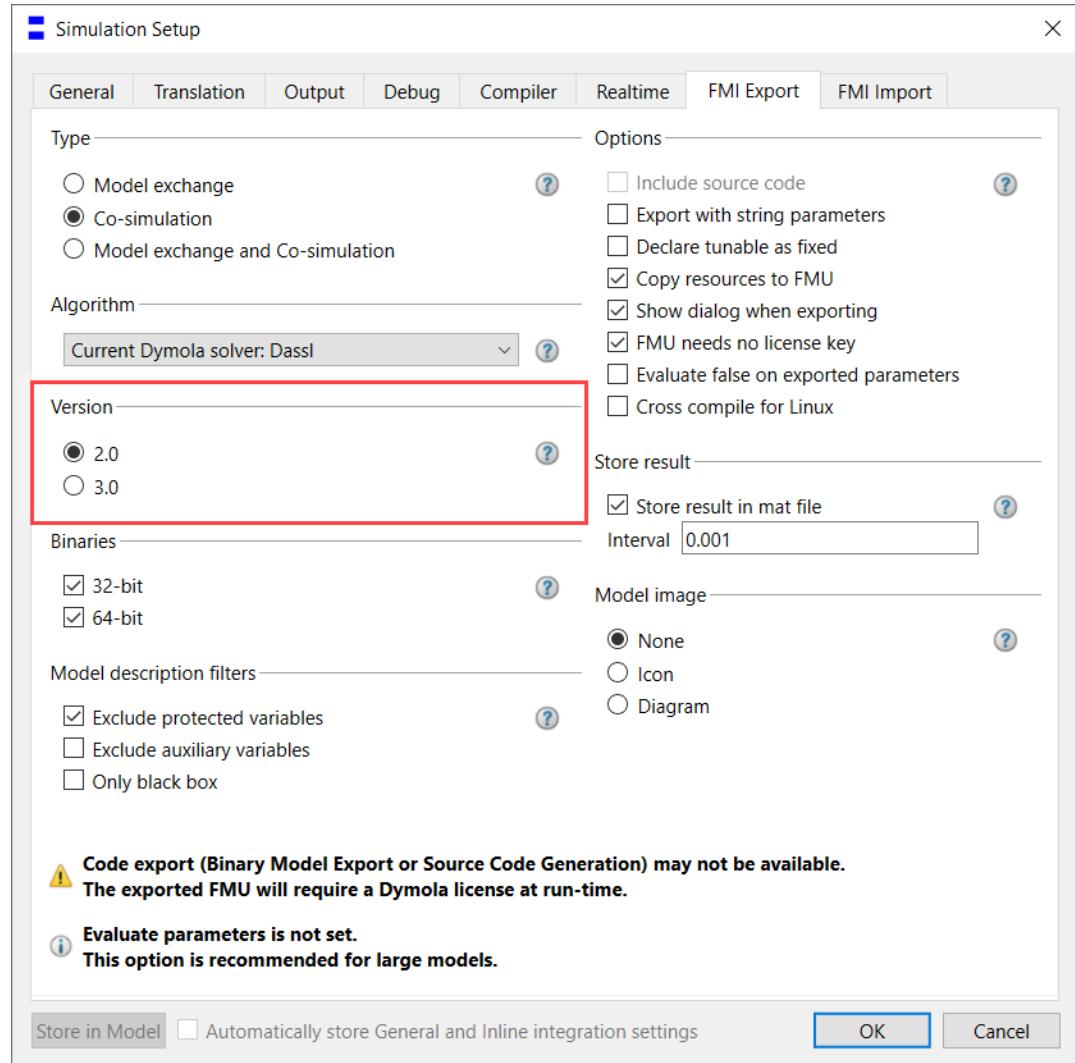
FMI 1.0 export alternative removed from menus

Due to technical limitations with FMI version 1, the support for FMI version 1 will be fully discontinued in a future release. Users are recommended to use FMI version 2 or 3 instead.

In Dymola 2025x Refresh 1, the status is:

- FMI 1.0 has been removed from the menu when exporting an FMU, and from the general simulation setup FMI Export tab:





- Exporting in FMI 1.0 format is still supported using scripting, by the built-in function `translateModelFMU`, but that support will also be removed in a future release of Dymola.

FMU Export: By default use unique naming when generating source code FMUs

To enable import of multiple source code FMUs, the extra source code file that includes all other C files of an FMU must have unique names for these FMUs. To accomplish that, when exporting an FMU with source code, by default that source code file is now generated with a unique name.

That is, the default value of the flag `Advanced.FMI.FMUSourceCodeUniqueNaming` has been changed, the default value is now `true`.

FMU Import: Source code FMU import in Dymola

In previous versions of Dymola, you could export FMUs with source code, from Dymola 2025x Refresh 1 you can also import FMUs with source code included.

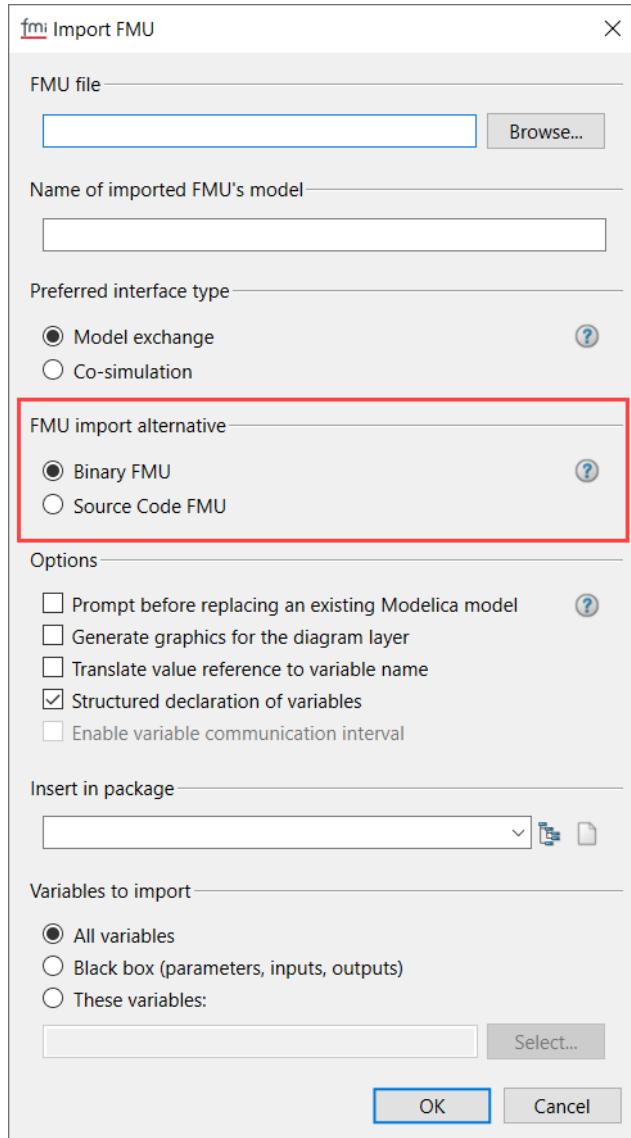
Important!

- This functionality is not supported for FMUs of FMI version 1.
- For source code FMUs generated with Dymola, source code import of such FMUs generated by versions earlier than Dymola 2025x Refresh 1 is not supported.
- Import of multiple source code FMUs is supported.

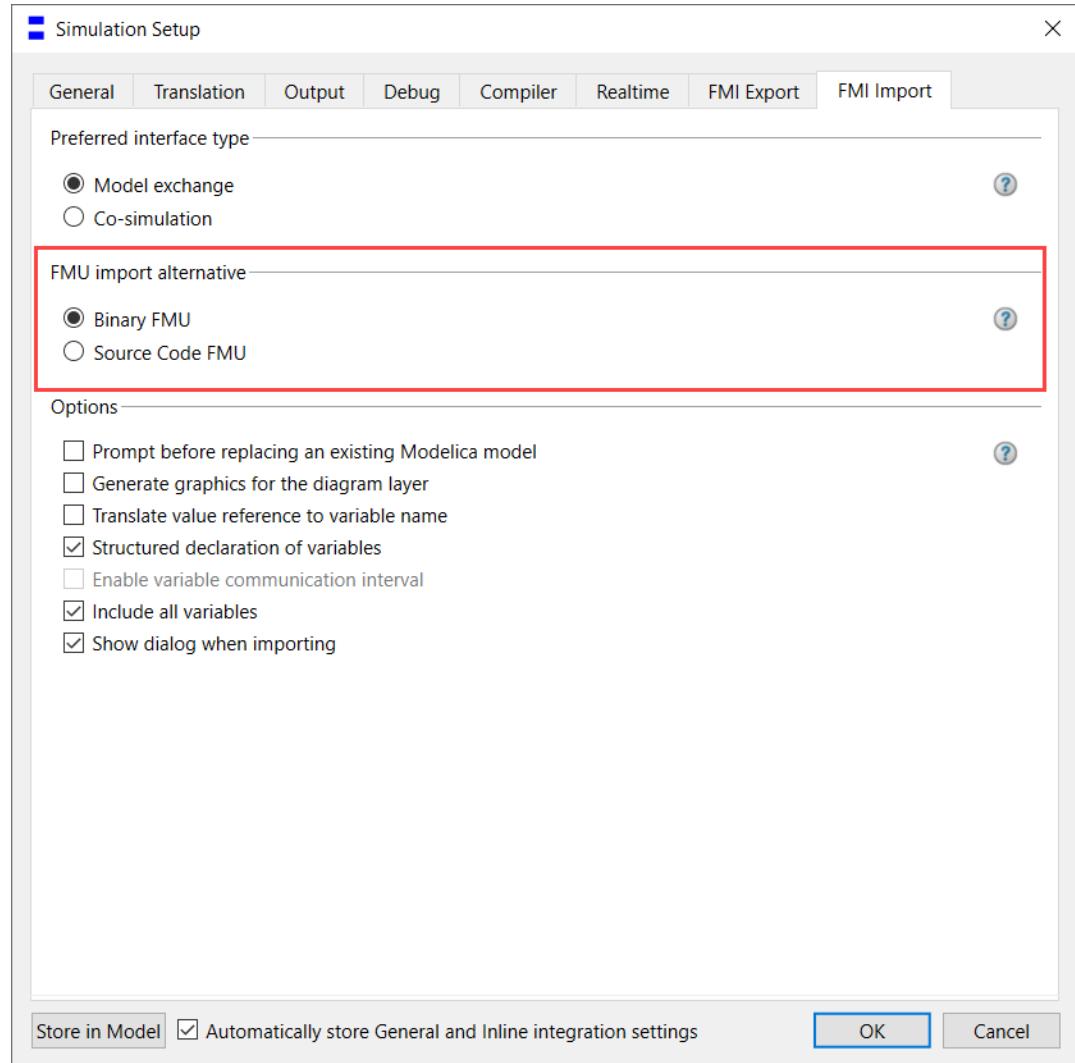
Note the difference between exporting an FMU with source code and importing a source code FMU. For the exporting, to include source code is an option, this means that such an FMU can be imported either as binary FMU or as source code FMU, both alternatives are generated by such an export. In addition, source code export demands a specific license. When importing, you must select to import *either* as binary FMU *or* as source code FMU. No specific license is needed for the import.

Source code FMU import by command

The dialog when importing an FMU with the command **File > Open > Import FMU...** has been extended to allow source code FMU import:



In addition, the simulation setup has been enhanced with this alternative. (The simulation setup is reached by the command **Simulation > Setup**, the **FMI Import** tab.)



The alternative **Binary FMU** is the default; it corresponds to what was also previously available in Dymola.

There is a corresponding flag `Advanced.FMI.SourceCodeImport` available, the default value is `false`, corresponding to the **Binary FMU** alternative.

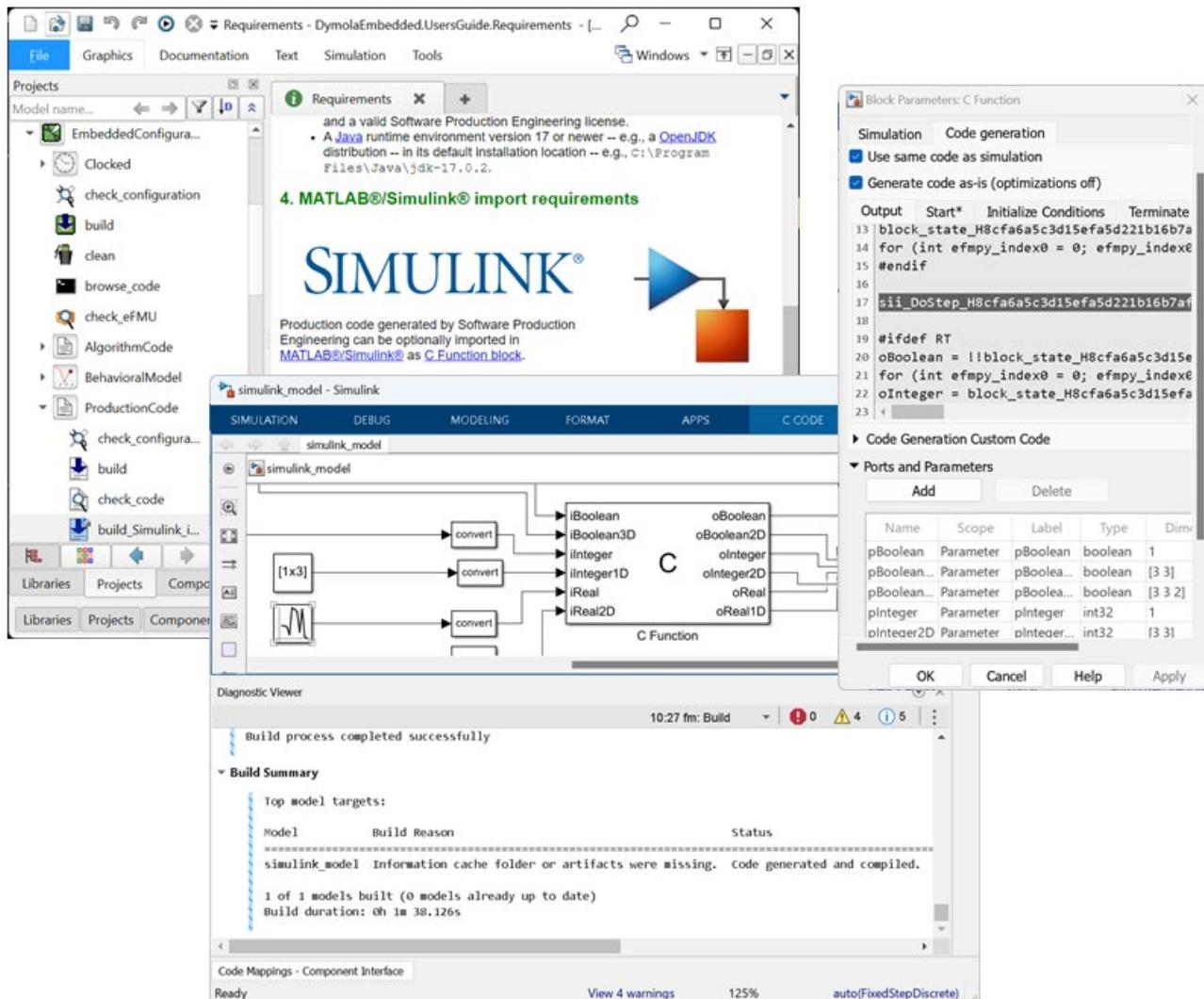
Source code FMU import by scripting

The built-in function `importFMU` now has a new Boolean input argument `sourceCodeImport`. The default value is `false`, corresponding to the binary import also previously available.

If you set `sourceCodeImport` to `true`, the code generates a wrapper that compiles the FMU source files and use them to call the FMU when you execute the built-in function.

3.8.7 eFMI Support in Dymola

The main new feature in Dymola 2025x Refresh 1 is the added support to generate MATLAB® scripts that import an eFMU Production Code container generated by Software Production Engineering as a Simulink® C Function block (`.DymolaEmbedded.EmbeddedConfiguration.ProductionCode.build_Simulink_import()`). For details, please consult the requirements documentation in `DymolaEmbedded.UsersGuide.Requirements`.



Other notable improvements are:

- Support to expose the independent parameters of record instances on any nesting level as tunable parameters of the GALEC block-interface. To do so, the record type either, has to extend `DymolaEmbedded.SupportModels.eBlockParameterization`, or have the new annotation `__Dymola_eFMI_ExposeTunableParameters = true`.
- eFMU co-simulation stubs now support multi-dimensional tunable Boolean parameters.
- Added further Modelica implementations for the built-in functions of the eFMI GALEC language (`ln`, `ln1D`, `ln2D`, `lg`, `lg1D`, `lg2D`, `tan`, `tan1D`, `tan2D`, `asin`, `asin1D`, `asin2D`, `acos`, `acos1D`, `acos2D`, `atan`, `atan1D`, `atan2D`, `sinh`, `sinh1D`, `sinh2D`, `cosh`, `cosh1D`, `cosh2D`, `tanh`, `tanh1D`, `tanh2D`).
- Experiment-packages now use the new maximum of absolute and relative tolerance style of the `Testing` library instead of the old sum approach (cf. `Testing.Utilities.Types.ToleranceStyle`). This is required by the eFMI® Standard 1.0.0 Beta 1.
- Experiment-packages now support to configure tolerances for Integer GALEC block-interface outputs.
- Manifests of generated Behavioral Model containers are now according to the eFMI® Standard 1.0.0 Beta 1, which finalized the Behavioral Model standardization and completely changed manifest structure compared to the previous candidate-draft Alpha 4.
- Reference trajectories files of generated Behavioral Model containers now always have the reference time grid as first column and floating-point numbers in canonical form as normalized in GALEC and required by the eFMI® Standard 1.0.0 Beta 1.
- Updated check code scripts to work with latest Cppcheck version (open source 2.17.1, premium 25.3.0.2).
- Added new function to find the best suited available Java Runtime Environment for executing Software Production Engineering, considering environment variables (`DYMOLA_JAVA_HOME` and `JAVA_HOME`), the available Java versions in default installation directories and the minimal Java version required (cf. `DymolaEmbedded.BuildUtilities.ExternalResources.resolve_JAVA_HOME()`).

3.9 Advanced Modelica Support

Option to include values in assert-messages

You can now include values in assert-messages by using the flag `Advanced.Translation.Assert.IncludeValue`. The possible values of the flag are:

- 0 – skip including values
- 1 – add if seems to make sense (default value)
- 2 – always add values
- 3 – use the full name

3.10 Modelica Standard Library and Modelica Language Specification

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

Dymola 2025x Refresh 1 supports the future Modelica Standard Library 4.1.0 version:

- Dymola 2025x R1 has been tested with development versions of MSL 4.1.0.
- The flag `Advanced.PlaceDymolaSourceFirst=1` now works automatically for development versions of 4.1.0; (so that you don't have to set `Advanced.PlaceDymolaSourceFirst=2` for MSL 4.1.0 and then set it back to 1 for MSL 4.0.0).
- As in previous releases a Dymola specific `ModelicaServices` 4.1.0 is included in the Dymola distribution; it shall be used instead of the one included with MSL 4.1.0.

3.11 Documentation

General

In the software, distribution of Dymola 2025x Refresh 1 Dymola User Manuals of version “March 2025” will be present; these manuals include all relevant features/improvements of Dymola 2025x Refresh 1 presented in the Release Notes, except the “under development” ones (if present).

Major update of chapter “Design Optimization” in the manuals

The chapter “Design Optimization” previously only contained a reference to other documentation, now it contains the integrated Optimization GUI, as described here in “Integrated GUI for Model Optimization” starting on page 40. In addition, it contains a more elaborated example of the integrated GUI.

3.12 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2025x Refresh 1 are listed.

3.12.1 Hardware requirements/recommendations

Hardware requirements

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

Hardware recommendations

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

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

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

3.12.2 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions

Dymola 2025x 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 2025x 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 as code generator for this compiler.
- Visual Studio 2019 Build Tools **Notes:**
 - The recommended selection to run Dymola is the workload “C++ build tools” + the option “C++/CLI Support...”
 - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
 - This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2019 alternative: **Visual Studio 2019/Visual C++ 2019 (16)**.
 - For more information about installing and testing this compiler with Dymola, see the document [Installing Visual Studio Build Tools for Dymola.pdf](#).
 - You can select to use Clang as code generator for this compiler.
- Visual Studio 2022 Community (17). Note that you can select to use Clang as code generator for this compiler.
- Visual Studio 2022 Build Tools **Notes:**
 - The recommend selection to run Dymola is the workload “Desktop development with C++” + the option “C++/CLI Support...”
 - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
 - This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2019 alternative: **Visual Studio 2022/Visual C++ 2022 (17)**.

- For more information about installing and testing this compiler with Dymola, see the document [Installing Visual Studio Build Tools for Dymola.pdf](#).
- You can select to use Clang as code generator for this compiler.

Microsoft C/C++ compilers, professional editions:

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

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

Clang compiler

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

Intel compilers

Note!

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

MinGW GCC compiler

Dymola 2025x 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.

WSL GCC compiler (Linux cross-compiler)

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

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

Dymola license server

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

As an alternative to FLEXnet, Dassault Systèmes License Server (DSLS) can be used. Dymola 2025x Refresh 1 supports DSLS R2025x. Earlier DSLS versions cannot be used.

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

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

Linux

Supported Linux versions and compilers

Dymola 2025x 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 2025x Refresh 1 it is Qt 6.8.1.
- Select Supported platforms)

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

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

Dymola 2025x Refresh 1 is supported as a 64-bit application on Linux. Corresponding support for 64-bit export and import of FMUs is included.

Notes:

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

Note on libraries

- The following libraries are currently not supported on Linux:
 - Process Modeling Library
 - Thermodynamics Connector Library
 - UserInteraction Library

Dymola license server

For a Dymola license server on Linux, all files needed to set up and run a Dymola license server on Linux, except the license file, are available in the Dymola distribution. (This also

includes the license daemon, where Dymola presently supports FLEXnet Publisher 11.16.2.1.)

As an alternative to FLEXnet, Dassault Systèmes License Server (DSLS) can be used. Dymola 2025x Refresh 1 supports DSLS R2025x. Earlier DSLS versions cannot be used.

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

"