

Dymola

Dynamic Modeling Laboratory

Dymola Release Notes

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

Document version: 1

© Copyright 1992-2014 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 Science Park
SE-223 70 Lund
Sweden

Support: <http://www.3ds.com/support>
URL: <http://www.Dymola.com>
Phone: +46 46 2862500

Contents

1	Important notes on Dymola.....	5
2	About this booklet	5
3	Dymola 2015.....	6
3.1	Introduction	6
3.1.1	Additions and improvements in Dymola	6
3.1.2	New and updated libraries	7
3.1.3	Limited Availability Features	7
3.2	Developing a model	8
3.2.1	Modelica Standard Library version 3.2 not included.....	8
3.2.2	Improved declaration of variables	8
3.2.3	The Split Model command enhanced and improved	10
3.2.4	Diagram layer zooming by spanning a rectangle.....	11
3.2.5	Selected commands (built-in functions) available as a library	12
3.2.6	Improved handling of tables	13
3.2.7	Print preview available	13
3.2.8	On-demand loading of models and packages (LA)	14
3.2.9	Minor improvements	15
3.3	Simulating a model	16
3.3.1	Print preview available	16
3.3.2	Plot window	16
3.3.3	Scripting	17
3.3.4	Minor improvements	18
3.4	Installation.....	21
3.4.1	Installation on Windows	21
3.4.2	Installation on Linux.....	21
3.5	Model Management	22
3.5.1	Extended encryption of libraries.....	22
3.6	Other Simulation Environments.....	22
3.6.1	Dymola – Matlab interface	22
3.6.2	Real-time simulation.....	23
3.6.3	New Java interface for Dymola on Windows.....	24
3.6.4	Python interface for Dymola on Windows available	24
3.6.5	FMI Support in Dymola	25
3.6.6	Code and Model Export.....	34

3.7	Advanced Modelica Support	34
3.7.1	Minor improvements	34
3.8	New libraries	34
3.8.1	Flight Dynamics Library	34
3.9	Updated libraries	36
3.9.1	Air Conditioning Library	36
3.9.2	DataFiles Library	36
3.9.3	Design Library	36
3.9.4	Electric Power Library	36
3.9.5	Engine Dynamics Library	36
3.9.6	Fuel Cell Library	36
3.9.7	Heat Exchanger Library	36
3.9.8	Hydraulics Library	37
3.9.9	Hydro Power Library	38
3.9.10	Liquid Cooling Library	38
3.9.11	Model Management Library	38
3.9.12	Plot 3D Library	38
3.9.13	Pneumatics Library	38
3.9.14	Thermal Power Library	38
3.9.15	User Interaction Library	38
3.9.16	Vapor Cycle Library	39
3.9.17	Vehicle Dynamics Library	39
3.10	Documentation	40
3.11	Appendix – Installation: Hardware and Software Requirements	41
3.11.1	Hardware requirements/recommendations	41
3.11.2	Software requirements	41

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.

Two types of compilers are supported on Windows in Dymola 2015:

Microsoft Visual Studio C++

This is the recommended compiler for professional users.

Note that **free** Microsoft compiler versions earlier than Microsoft Visual Studio Express 2008 are not supported (concerning **full** versions, some earlier versions are supported). Refer to section “Compilers” on page 41 for more information.

GCC

Dymola 2015 introduces limited support for the MinGW GCC compiler. For the list of current limitations; refer to section “GCC compiler supported” on page 21.

Installation on Linux

To translate models, Linux relies on a GCC compiler, which is usually part of the Linux distribution. Refer to section “Dymola versions on Linux and operating system versions, and compiler” on page 42 for more information.

2 About this booklet

This booklet covers Dymola 2015.

The disposition is similar to the one in Dymola User Manual Volume 1 and 2; the same main headings are being used (except for, e.g., Libraries and Documentation).

3 Dymola 2015

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2015. In particular, Dymola 2015 provides:

- Extended FMI support:
 - Support of FMI 2.0 Release Candidate 1 (page 25).
 - All Dymola solvers supported for FMU Co-Simulation export for FMI 1.0 on Windows (page 25).
 - Improved handling of external resources for FMU export on Windows (page 26)
 - String parameters are supported (page 28).
 - Online tunable parameters supported for FMI 2.0 RC1 (page 30).
 - Structured declaration of variables supported for FMI 2.0 RC1 (page 30).
 - Improved import of FMUs with many inputs/outputs (page 31).
- New interfaces to Dymola on Windows:
 - Java (page 24)
 - Python (page 24)
- Model creation by copying, using the Split Model command, is now supported (page 10).
- Simplified zooming in the diagram layer (page 11).
- Extended plot support:
 - User-defined time units (page 16).
 - Headers defined as text objects (font etc. can be changed) (page 16).
- New compiler on Windows: GCC (limited support) (page 21).
- Improved declaration of variables (page 8).
- Improved handling of tables (page 13).
- Improved robustness of license borrowing for Windows: FLEXnet license server upgraded (page 21).
- Selected commands (built-in functions) available as a library (page 8).
- Print preview available (page 13).
- Extended encryption of libraries to faster find the correct version of the libraries (page 22).
- On-demand loading of models and packages (Limited Availability) (page 14).

3.1.2 New and updated libraries

New libraries

The following libraries are new:

- Flight Dynamics Library, version 1.0.

For more information about the new libraries, please see section “New libraries” starting on page 34.

Updated libraries

The following libraries have been updated:

- Air Conditioning Library, version 1.8.7.
- DataFiles Library, version 1.0.1.
- Design Library, version 1.0.3.
- Electric Power Library, version 2.1.1.
- Engine Dynamics Library, version 1.2.1.
- Fuel Cell Library, version 1.2.1.
- Heat Exchanger Library, version 1.1.
- Hydraulics Library, version 4.0.
- Hydro Power Library, version 2.4.
- Liquid Cooling Library, version 1.2.1.
- Model Management Library, version 1.1.2.
- Plot 3D Library, version 1.0.3.
- Pneumatics Library, version 1.6.2.
- Thermal Power Library, version 1.8.
- User Interaction Library, version 0.63.
- Vapor Cycle Library, version 1.1.
- Vehicle Dynamics Library, version 1.9.

For more information about the updated libraries, please see the section “Updated libraries” starting on page 36.

3.1.3 Limited Availability Features

This version of Dymola contains certain features which are labeled “Limited Availability” (LA). It means that the implementation might be partial, diagnostics partial, the testing is not complete, and no support/maintenance and only limited documentation is available. However, we provide them to you as early as possible in order for you learn about these features, plan for later use, and be able to give us your feedback. You typically need to set some switch to enable them. These features are planned to become Generally Available (GA) in the Dymola 2015 FD01 release.

The LA features are briefly described below and are marked (LA). In this release the on-demand loading of models and packages is the only main LA feature.

3.2 Developing a model

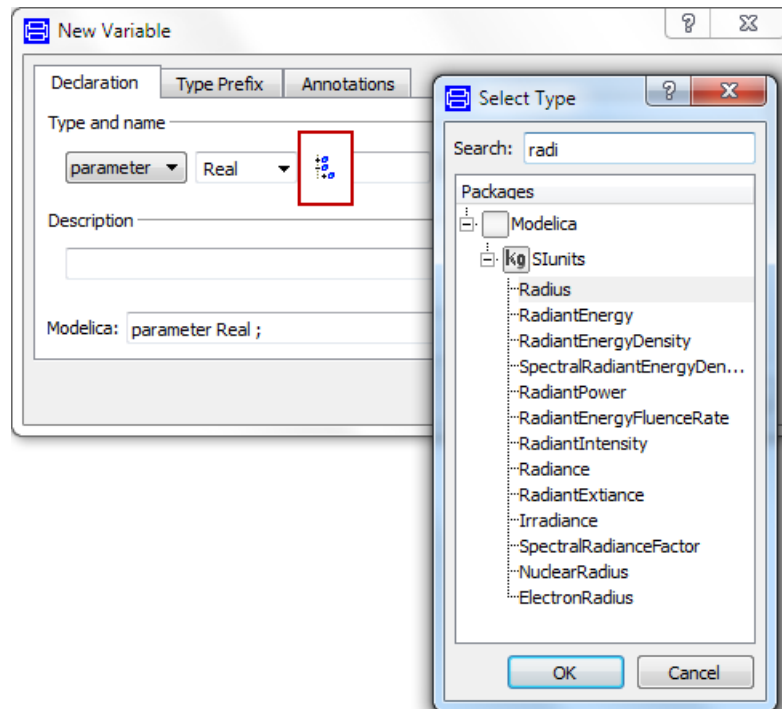
3.2.1 Modelica Standard Library version 3.2 not included

In Dymola 2015 only Modelica Standard Library version 3.2.1 is included in the distribution, and Modelica Standard Library 2.2.2 (for compatibility reasons).

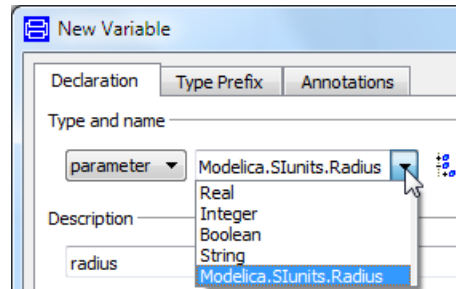
3.2.2 Improved declaration of variables

Type selection in variable declaration dialog improved

When declaring or editing a variable, it is now possible to select types other than Real, Integer, Boolean, and String; a new button **Select Type** in the dialog enables searching and inserting from all types available in the package browser (like the present **Insert Type** command).



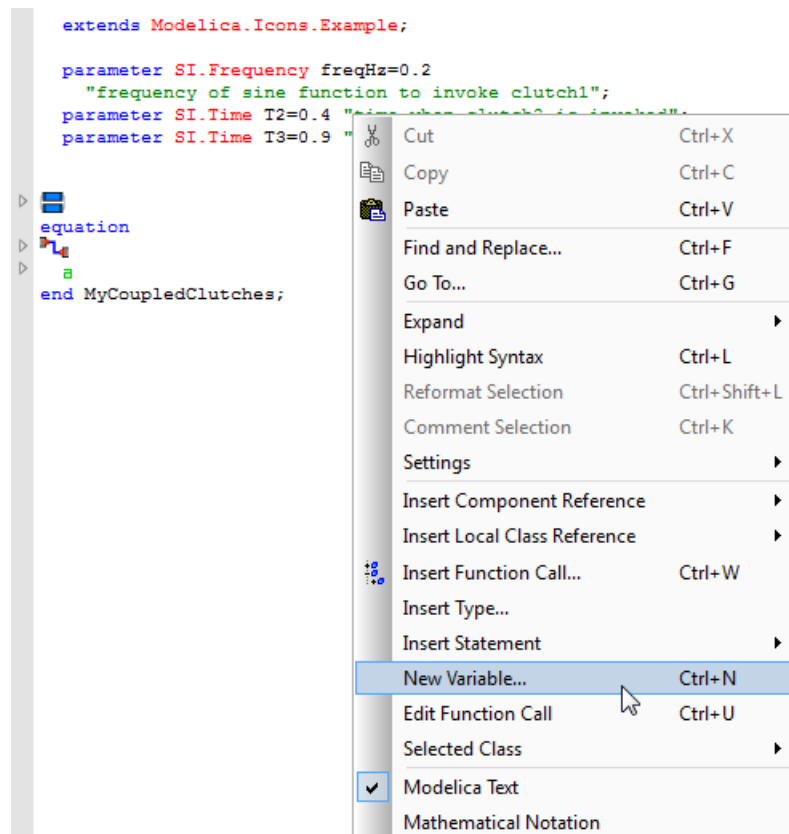
The selected type will be added to the dropdown type list as selected:



To see the whole type without using the dropdown menu, make the window wider.

Variable declaration added to the Modelica Text layer context menu

The context menu in the Modelica Text layer has been extended to allow defining new variables:



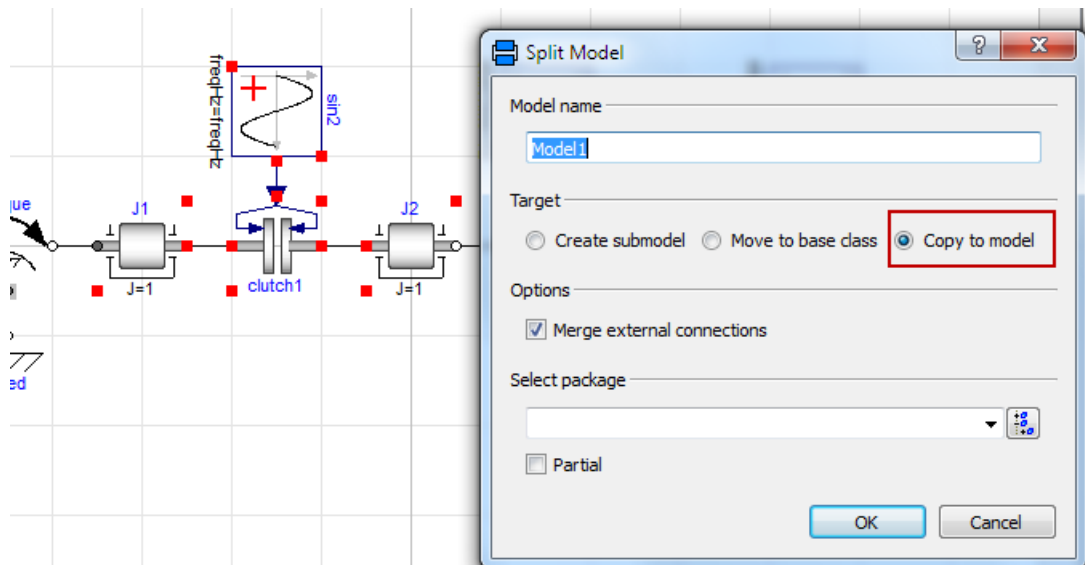
Selecting this command displays the variable declaration dialog. The new variable is added at the end of the declaration part of the Modelica Text layer. (The command **Edit > Insert Variable** works the same.)

A new variable declaration can only be added if there are no syntax errors in the Modelica text.

3.2.3 The Split Model command enhanced and improved

Model creation by copying, using the Split Model command, is now supported

A new selection **Copy to model** is available when using the split model command. (The command is reached by right-clicking on selected objects and selecting **Split Model...**, or by **Edit > Split Model...**)



The selected objects will be copied to a new model.

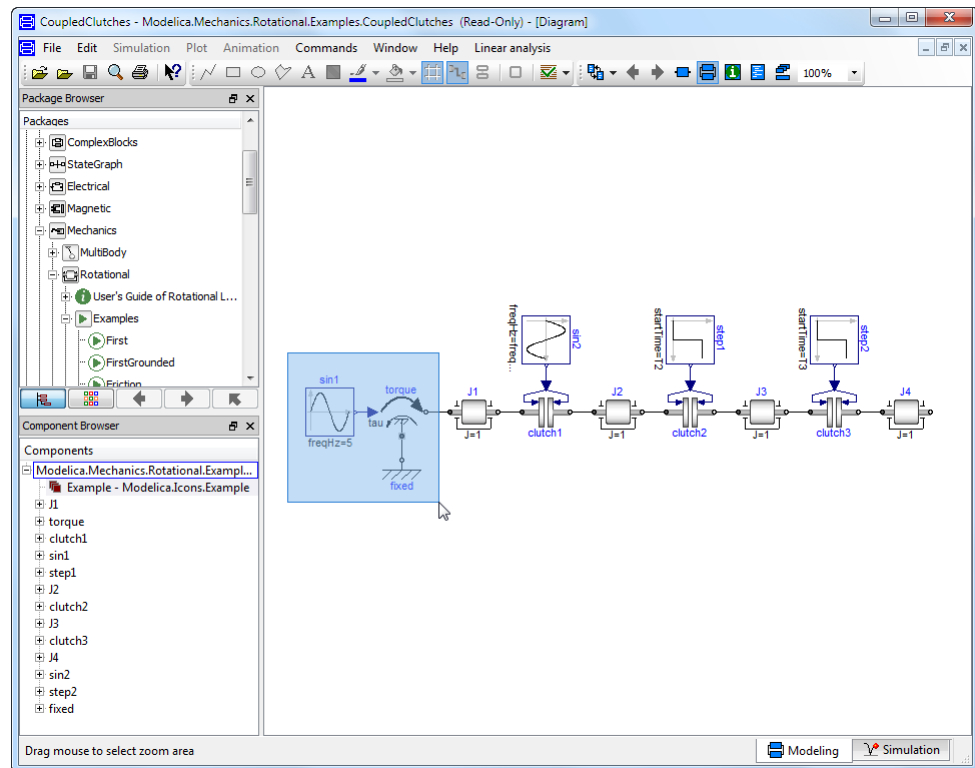
General improvements

In addition to the above, the Split Model command has been improved in general:

- For all options (**Create submodel**, **Move to base class**, and **Copy to model**), parameters used in the components are copied to the sub-system for more cases. Previously they were only copied when the parameters or components of them were used (" $=p$ " and " $=p.x$ "). Now this has been extended; parameters are now also copied when used in arrays/matrices/simple arithmetic expressions.
- For all options (**Create submodel**, **Move to base class**, and **Copy to model**), inner/outer components are now also copied to the sub-system. This is similar to parameter handling. However, the alternative **Create submodel** if necessary changes inner/outer components to outer components, without any modifiers, in order to create an appropriate submodel.

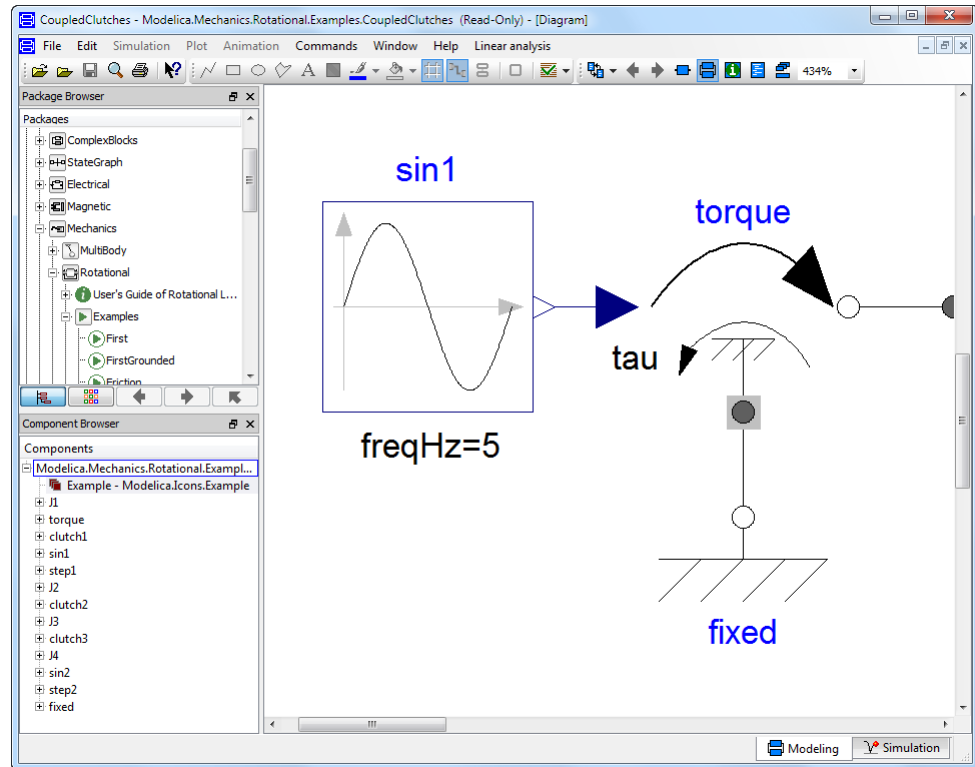
3.2.4 Diagram layer zooming by spanning a rectangle

The diagram layer (and the icon layer) can now be zoomed by pressing **Alt** and spanning a rectangle¹.

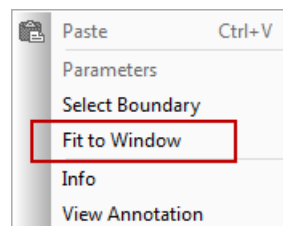


Releasing the mouse, the result is:

¹ On Linux, **Ctrl+Shift** have to be used instead of **Alt**.



To go back (to default), set the zoom to 100% in the dropdown menu in the upper toolbar, or use the new command **Fit to Window** in the context menu of the diagram layer or icon layer.

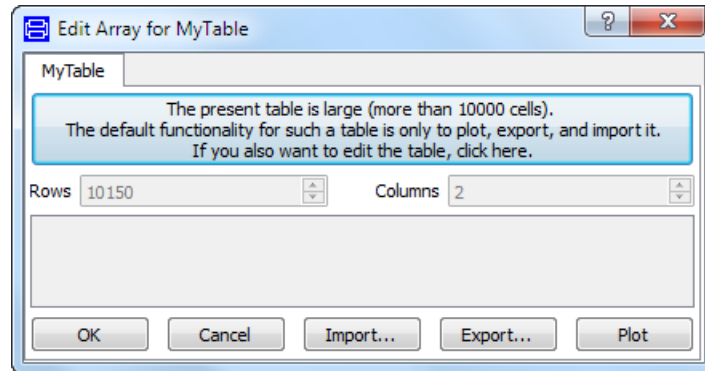


3.2.5 Selected commands (built-in functions) available as a library

Previous version of Dymola introduced a package with selected commands (built-in functions). This package is now available as a library in the library menu. The library can be opened by the command **File > Libraries > DymolaCommands**.

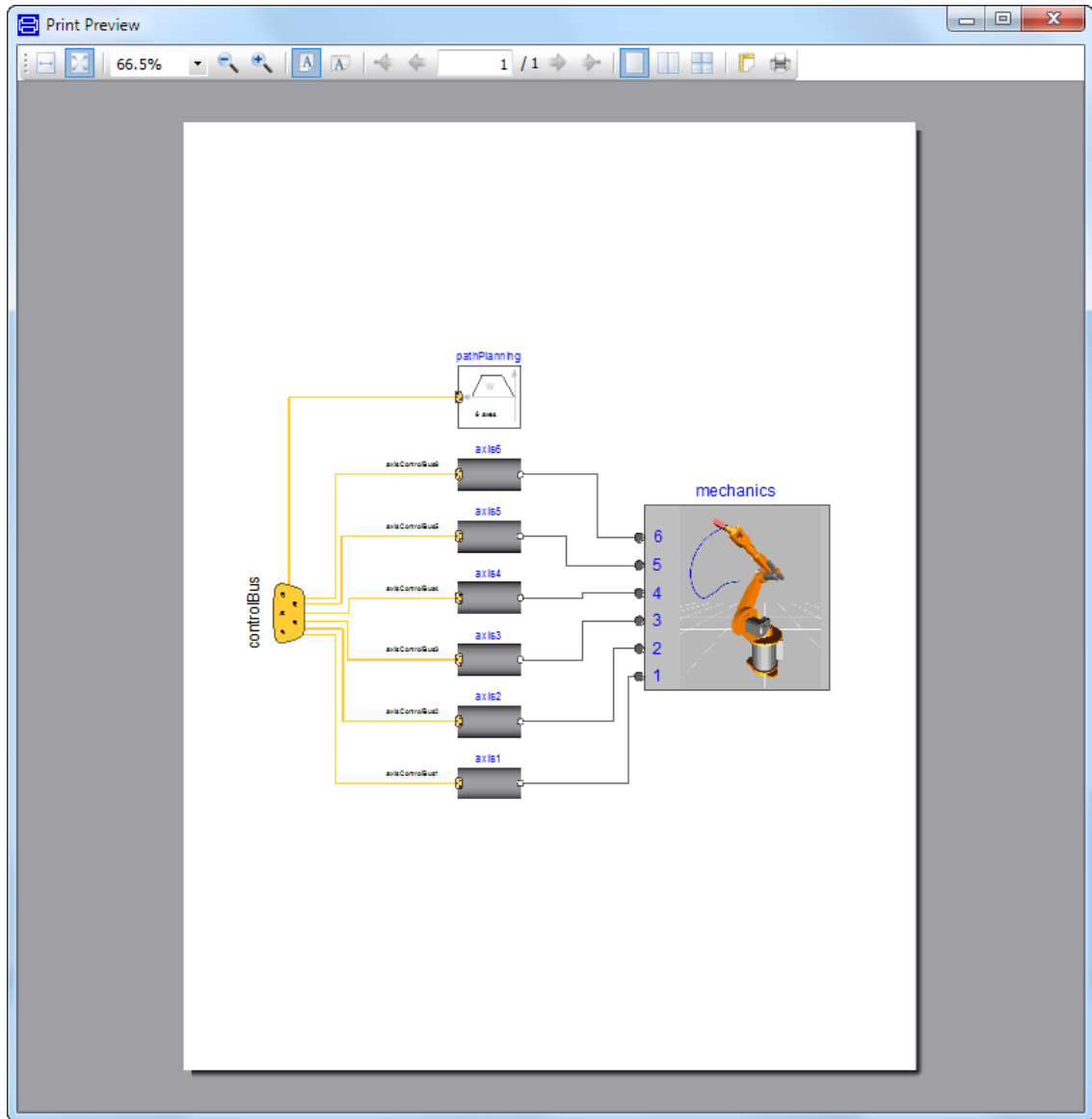
3.2.6 Improved handling of tables

When tables are large (more than 10000 cells), the default functionality in the matrix editor is now only to import/export and plot them, not to edit them. If editing is to be done, a command must be given; a warning that this will be a time-consuming operation follows. This improves handling of large tables, since many users are content with import/export and plot.



3.2.7 Print preview available

In Dymola 2015, the command **File > Print Preview** displays a print preview, where for example the effect of selecting portrait or landscape printing can be displayed. The print setup can be changed, and printing can be done from the window.



3.2.8 On-demand loading of models and packages (LA)

It is possible to select to avoid loading classes using the flag `Advanced.DemandLoadClasses`. The following alternatives are available:

```
Advanced.DemandLoadClasses = 0;
```

This is the default value; entire libraries are loaded in Dymola when needed.

```
Advanced.DemandLoadClasses = 1;
```

Classes in libraries are loaded when needed. Classes are needed when they are used for components/base-classes in the GUI, or used during translation. Note that Parameter dialogs with class-selectors will usually load complete libraries when using this setting. The end result of this setting is the same as the default setting, but the loading of classes rather than whole libraries speeds up the start-up time of opening a large model in Dymola.

```
Advanced.DemandLoadClasses = 2;
```

Classes in libraries are only loaded when needed, even if that reduces the choices in the GUI; e.g. the class-selectors that include the matching classes will not load new classes. This setting further speeds up the start-up time of opening a large model in Dymola.

There are some restrictions for on-demand loading:

- Older libraries without the file `package.order` in each directory will be read completely.
- Encrypted libraries are read completely.
- In the Demo version of Dymola the libraries included with Dymola will be read completely.

3.2.9 Minor improvements

Improved handling of array elements

In Dymola 2015 all array variable elements can be set individually (value, start, quantity, min, max, nominal, stateSelect, fixed, unit, displayUnit). Previously only value, start and quantity were supported to set individually.

Code completion in the Modelica Text editor for incomplete models

The code completion in the Modelica Text editor now also works for Modelica texts that have syntactic errors. Thus you can add a declaration of a new component and the code completion will give you its subcomponents, even if the declaration of the component is not yet syntactically correct.

Improved handling of adjustable maximum line length in the Modelica Text editor

The dotted vertical line in a Modelica Text editor that indicates the maximum line length is now local for each Modelica Text editor window. (The line can be dragged to change the maximum line length, the user can then adapt to the new value by right-clicking and selecting the command **Highlight Syntax**, or by giving the corresponding short command **Ctrl+L**).

Improved support for checking of interactive functions

If a function calls a built-in function that is not defined in the Modelica Language Specification (such as `translateModel`) a check will no longer state that

translateModel is an unknown function – a warning will instead recommend using the following annotation to turn off the warning:

```
annotation(__Dymola_interactive=true);
```

Both with and without that annotation the check will be less thorough than for normal functions; but performs some checks, e.g. the spelling of function names.

Displaying of constants in the Used Classes layer now controllable

Right-clicking in the Used Classes layer of the Edit window, and selecting **Settings**, a new entry **Include Constants** is available. Constants are by default displayed.

3.3 Simulating a model

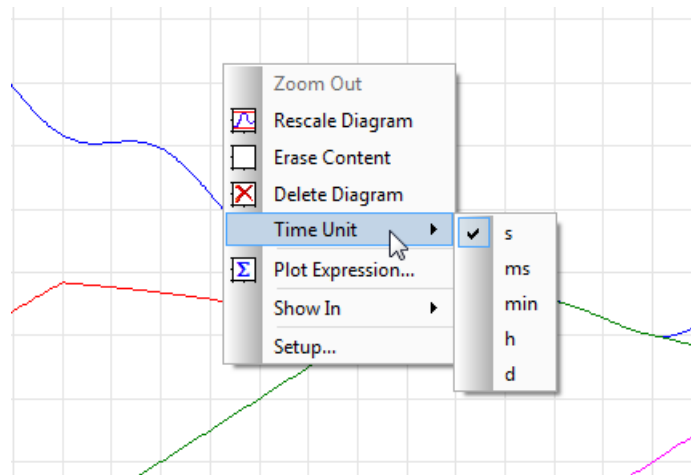
3.3.1 Print preview available

In Dymola 2015, the command **File > Print Preview** displays a print preview. See page 13.

3.3.2 Plot window

User-defined time units supported

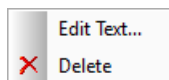
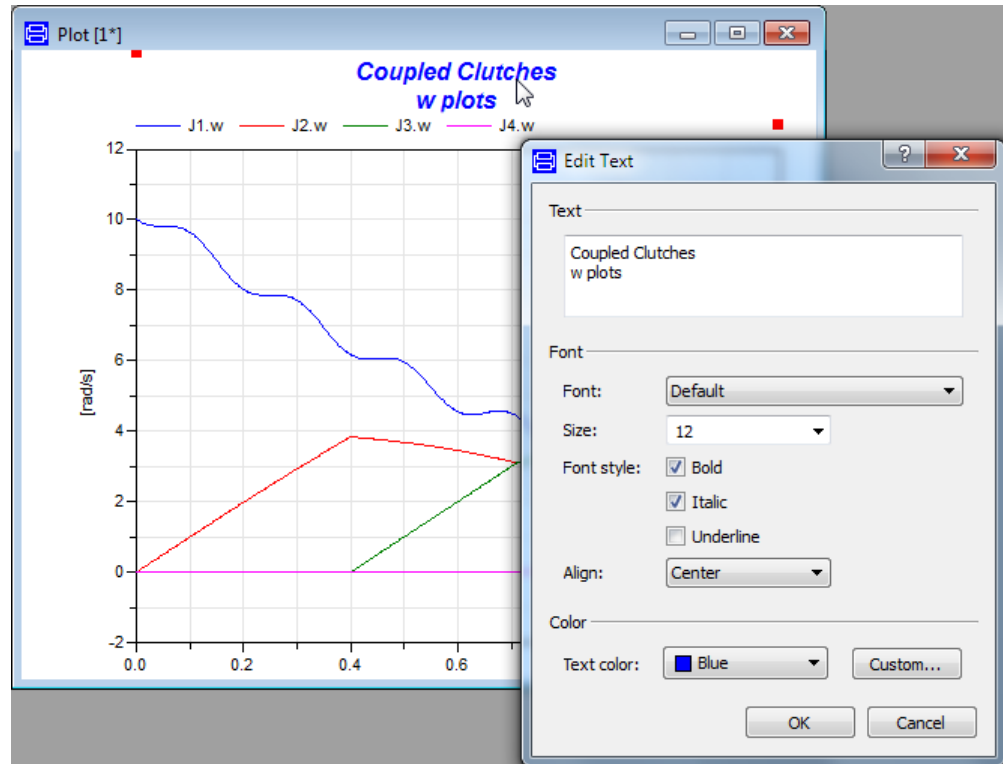
The selections available for the time on the horizontal axis are now the same as the user-defined time units, except s (seconds) that is always shown as default time unit.



(User-defined time units are the display units for time.)

Plot headings treated as text objects

In Dymola 2015 plot headings are treated as text objects; font size, alignment, color, etc. can be changed by the menu that appears when double-clicking the header or right-clicking the header and selecting **Edit Text...**



The header can be deleted using the context menu of the header.

The functionality is also supported by scripting, using the new built-in function `plotHeading`. The header above can be created, using this function, as:

```
plotHeading(textString="Coupled Clutches\nw plots",  
fontSize=12, lineColor={0,0,255}, textStyle={TextStyle.Bold,  
TextStyle.Italic});
```

3.3.3 Scripting

New command for clearing all flags

A new built-in function `clearFlags` has been added; executing this function resets all flags and integer constants to their default values.

New command for multi-simulation

A new built-in function has been added: `simulateMultiResultsModel`. The function is similar to the existing built-in function `simulateMultiExtendedModel`; the difference is that the result of using the existing one gives the endpoints, while the result of the new function is the whole trajectories.

An example of call is:

```
simulateMultiResultsModel(  
  "Modelica.Mechanics.Rotational.Examples.CoupledClutches",  
  stopTime=1.2, numberOfIntervals=10, resultFile="CoupleClutches",  
  initialNames={"freqHz"}, initialValues=[0.1;0.2;0.3;0.4],  
  resultNames={"J1.w", "J3.w"});
```

A comparison between the now available built-in functions for simulation:

Function	Additional input	Output
<code>simulateModel</code>		Trajectories for one simulation.
<code>simulateExtendedModel</code>	Parameter values and start values (for one simulation).	Endpoints for one simulation.
<code>simulateMultiExtendedModel</code>	As <code>simulateExtendedModel</code> , but for several simulations.	Endpoints for several simulations.
<code>simulateMultiResultsModel</code>	As <code>simulateExtendedModel</code> , but for several simulations.	Trajectories for several simulations.

3.3.4 Minor improvements

Improving the code efficiency when using the Visual Studio 2012 compiler

The Visual Studio 2012 compiler is fully supported. However, this compiler by default generates a bit less efficient code than previous versions of the compiler, with the selected optimization settings. As a temporary work-around you can set the flag

```
Advanced.Define.GlobalOptimizations = 2;
```

before generating code, to activate global optimization in the compiler. (The default value of the flag is 0.)

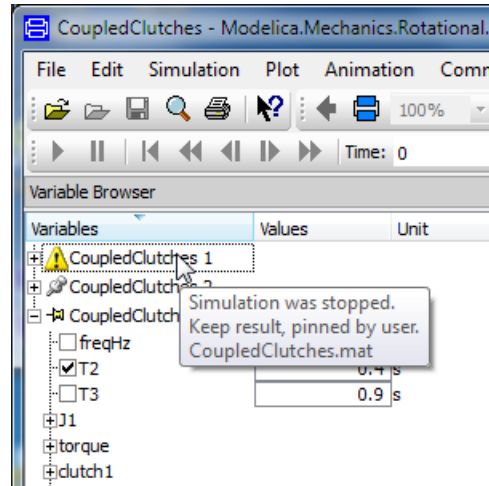
This flag works the same for all Visual Studio compilers, but the effect on compilers of previous versions is small. For the Visual Studio 2012 compiler, however, the simulation performance is restored, but the compilation of the code might take substantially longer for large models.

The setting above corresponds to the compiler command `/Og`.

Simulation status indicated in Variable browser

The simulation status (Running, Failed, and Stopped) of the simulation result files is indicated in the Variable browser. Failed and Stopped are indicated by a warning icon ⚠ in front of the result file name, while Running is indicated by the text (Running) after the result file name. No text means that the simulation is finished with success.

Result files of failed and stopped simulations can also be kept, but that is not indicated by any pin symbol. To see the complete status for a result file, use the tooltip.



Changed default value of the parameter Erase in plotArray and plotParametricCurve

The default value of the parameter erase is now true for the built-in functions plotArray and plotParametricCurve. This means that old plots are erased when new plots are created, if this default value is used.

Changed type for some input parameters in some built-in plot functions

Some input parameters in some built-in plot functions have changed type; from Integer to Enumeration.

Built-in function	Input parameters with type change to enumeration
createPlot	patterns markers
plot	patterns markers
plotArrays	patterns markers
printPlot	patterns markers

<code>printPlotArrays</code>	<code>patterns</code> <code>markers</code>
<code>plotSignalOperator</code>	<code>signalOperator</code>
<code>plotSignalOperatorHarmonic</code>	<code>signalOperator</code> <code>window</code>
<code>plotText</code>	<code>lineStyle</code> <code>horizontalAlignment</code>

Session setting for erasing plots when loading new file after simulation

A new flag `Advanced.DefaultAutoErase` can be used to set, for the session, if plots should be erased or not when loading a new file after simulation. The default value is `true`, meaning that the plots are erased when loading a new file after simulation.

The value of the flag only influences plot windows created by GUI commands. Plot windows created by the built-in function `createPlot` follow the input argument `autoerase` of the function.

The setting **Erase plot when loading new file after simulation** in the plot window now only influences the current plot window. This setting is available in the **Options** tab of the plot setup, reached by the command **Plot > Setup...** (or by right-clicking in the plot, then selecting **Setup...**).

Alphabetical order output and wildcards support in some built-in functions

The output from the built-in functions `list()` and `variables()` are now in alphabetical order, and grouped.

The wildcards `*` and `?` are supported when using them, for example:

- `list(variables={"A*"})` – lists all items starting with A.
- `list(variables={"Advanced.*"})` – lists all items starting with `Advanced.` – that is, list all Advanced flags settings.
- `list(variables={"*Output*"})` – lists all items containing `Output` in the text.

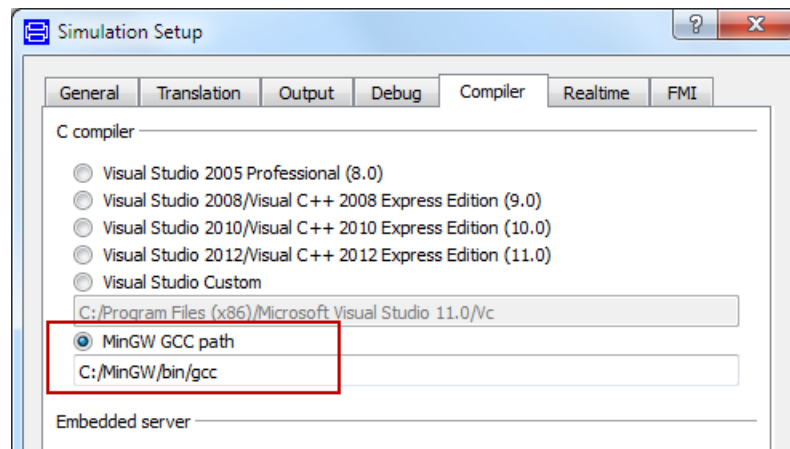
3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 41.

3.4.1 Installation on Windows

GCC compiler supported

Dymola 2015 introduces limited support for the MinGW GCC compiler. The compiler is selected by using the simulation setup, reached for example by the command **Simulation > Setup...**, the **Compiler** tab.



In order to use the GCC compiler, the user must download and install MinGW, with a GCC version compatible with 4.8.

Limitations

- Only ordinary simulation is supported (no DLL, FMU Export, DDE or OPC servers).
- Only 32-bit simulation is supported.
- Commercial libraries: Only limited testing, no support for external library resources.
- No support for run-time license.

Upgraded FLEXnet license server

The Dymola program and the vendor daemon have been upgraded to version 11.11 of FLEXnet on Windows.

3.4.2 Installation on Linux

Dymola on Linux now runs on SUSE Linux (Release 11), 32-bit and 64-bit, with gcc version 4.3.4, and compatible systems. In addition to gcc, the model C code generated by

Dymola can also be compiled by clang. To change compiler, change the variable CC in /opt/Dymola/insert/dsbuild.sh.

64-bit support on Linux

For information related to the 64-bit Linux version of Dymola 2015 (including 64-bit FMU export), please consult the web page www.Dymola.com/Linux.

FLEXnet license server

The Dymola program and the vendor daemon currently supports version 11.9 of FLEXnet on Linux. For upgrade to version 11.11, see www.Dymola.com/Linux.

3.5 Model Management

3.5.1 Extended encryption of libraries

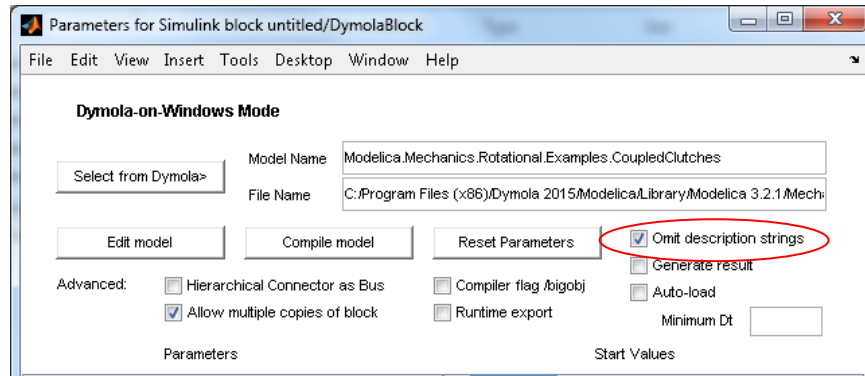
The encryption of libraries has been extended to be able to more quickly find the correct version of encrypted libraries. The new encryption can also be read in older Dymola versions; we recommend you to re-encrypt your encrypted libraries. Note: there is no change for “Encrypt Total Model” or libraries without version number.

3.6 Other Simulation Environments

3.6.1 Dymola – Matlab interface

Option to omit description strings in DymolaBlock GUI

An option to omit description strings for parameters and start values has been added to the DymolaBlock GUI. This option should be used for models with big input file (dsin) to avoid problems with Matlab memory limitations (especially for 32-bit installations).



Updated dymtools utility (LA)

An improved and extended version of the dymtools utility is available in the location

Program Files\Dymola 2015\Mfiles\dymtools2

Replace the standard dymtools path with this path in order to use the new version. This version works with Matlab R2009b and later.

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2009a (ver. 7.8) up to R2013b (ver. 8.2). Only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. The LCC compiler is not supported.

3.6.2 Real-time simulation

Improving real-time simulation performance when using the Visual Studio 2012 compiler

For real-time simulation using the Visual Studio 2012 compiler you can speed up the simulation by activating the flag

`Advanced.Define.GlobalOptimizations = 2;`

before generating code. This will improve the simulation speed, but note that the compilation may take substantially longer time for large models.

For more information about this flag, see section “Improving the code efficiency when using the Visual Studio 2012 compiler” on page 18.

Compatibility – dSPACE

Dymola 2015 generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases.

dSPACE DS1005 and DS1006 platforms

- dSPACE Release 6.4 with Matlab R2009a

- dSPACE Release 6.6 with Matlab R2010a
- dSPACE Release 7.0 with Matlab R2009bSP1 and R2010bSP1
- dSPACE Release 7.1 with Matlab R2011a
- dSPACE Release 7.2 with Matlab R2011b
- dSPACE Release 7.3 with Matlab R2012a
- dSPACE Release 7.4 with Matlab R2012b
- dSPACE Release 2013-A with Matlab R2012b, and R2013a
- dSPACE Release 2013-B with Matlab R2012b, R2013a, and R2013b

SCALEXIO

- dSPACE Release 7.4 with Matlab R2012b
- dSPACE Release 2013-A with Matlab R2012b, and R2013a
- dSPACE Release 2013-B with Matlab R2012b, R2013a, and R2013b

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.

Compatibility – xPC Target

Compatibility with Matlab xPC Target has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R2009a (xPC Target ver. 4.1) to R2013b (xPC Target ver. 5.5). Only Microsoft Visual C compilers have been tested.

3.6.3 New Java interface for Dymola on Windows

A new Java interface for Dymola is available in Dymola 2015, containing a number of functions to perform operations such as simulating, setting variables, plotting, and exporting data. For more information about this feature, please see “Dymola User Manual Volume 2”, chapter “Other Simulation Environments”, section “Java Interface for Dymola”.

The old version of Java interface is still available. This interface can still be used if wanting to call Java functions from Modelica, or calling Modelica functions from Java functions. When wanting to access Dymola remotely, the new interface should be used.

The Java interface is currently only supported on Windows.

3.6.4 Python interface for Dymola on Windows available

An interface for Python is available in Dymola 2015. For more information about this feature, please see “Dymola User Manual Volume 2”, chapter “Other Simulation Environments”, section “Python Interface for Dymola”.

The Python interface is currently only supported on Windows.

3.6.5 FMI Support in Dymola

Unless otherwise stated, features are available both for FMI version 1.0 and version 2.0 RC1.

Support for FMI 2.0 Release Candidate 1

Dymola 2015 supports FMI 2.0 RC1 (Release Candidate 1) that was published on October 18, 2013.

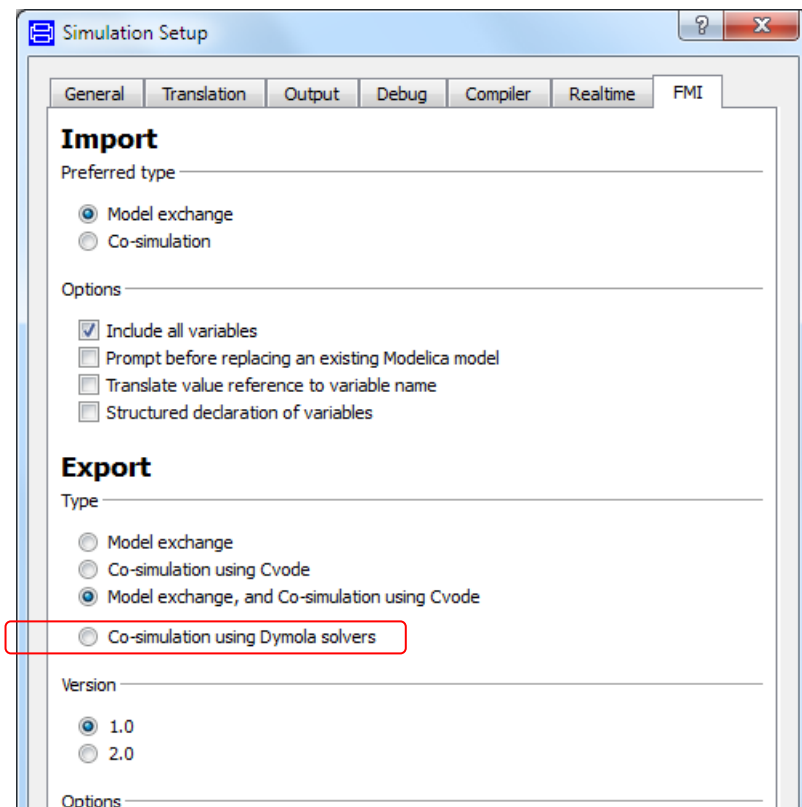
Note that the non-compatible FMI 2.0 Beta 4 specification is no longer supported.

For information about limitations and the latest status concerning supported features of FMI, please see www.Dymola.com/FMI.

All Dymola solvers supported for FMU Co-Simulation export for FMI 1.0 on Windows

Ticking the new option **Co-simulation using Dymola solvers**, the solver and tolerance that is selected in Dymola will be also used by the exported Co-simulation FMU. Source code generation FMU is not supported by this option.

This option is reached by the command **Simulation > Setup...**, the **FMI** tab:

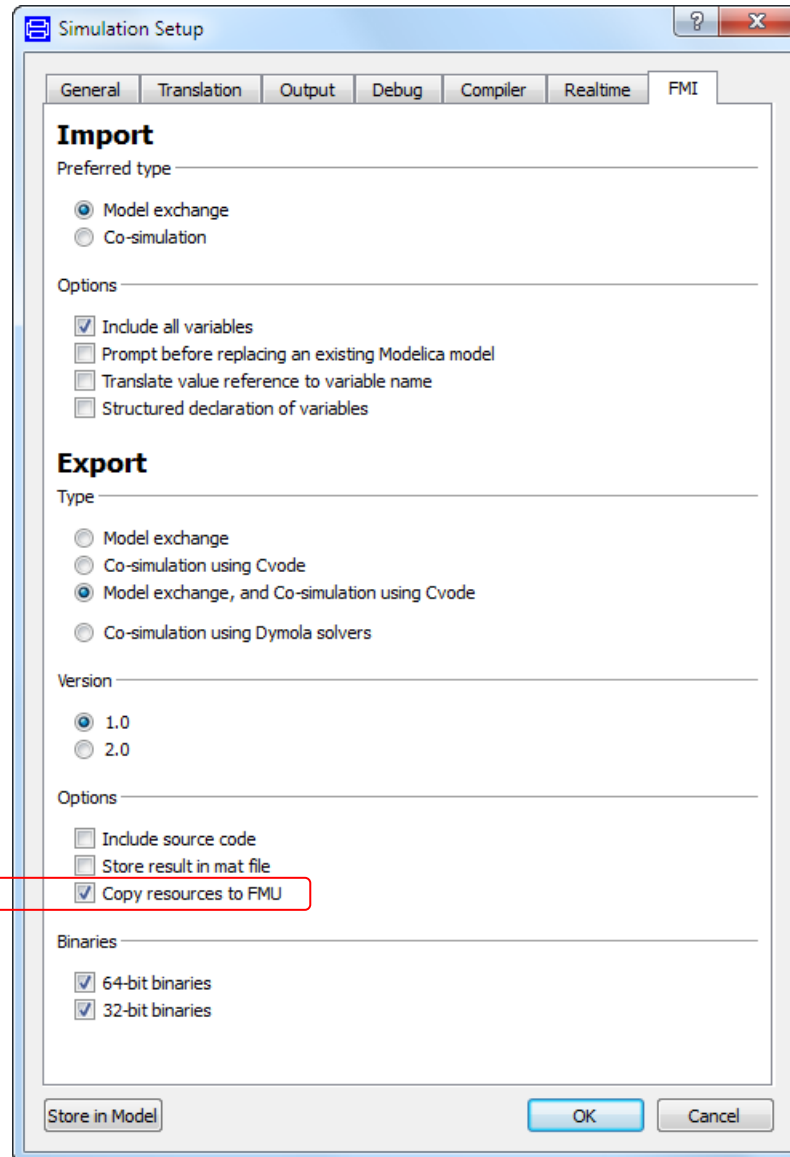


This option is also available in the built-in function `translateModelFMU`, by setting the input parameter `fmiType="csSolver"`.

Note – this option requires Binary Model Export license, and is currently only supported on Windows, for FMI version 1.0.

Improved handling of external resources for FMU export on Windows

External resources using the functions `ModelicaServices.ExternalReferences.loadResource` or `Modelica.Utilities.Files.loadResource` are now by default copied to the FMU. This corresponds to the default setting in the FMI tab of the simulation setup:



This menu is reached by the command **Simulation > Setup...**, the **FMI** tab.

The resulting FMU will be larger due to this. If this is not wanted, de-selecting the setting will not copy the resources to the FMU, but resource-paths using Windows-shares will be changed to UNC-paths when possible. This makes the FMU usable within a company – without increasing its size.

For an example, see the extended example of the string parameter support example below.

Note – the resource handling is currently only supported on Windows.

String parameters supported

In Dymola 2015 string parameters are supported in FMUs. For the FMU export to support string parameters, the following flag must be set:

```
Advanced.AllowStringParameters=true
```

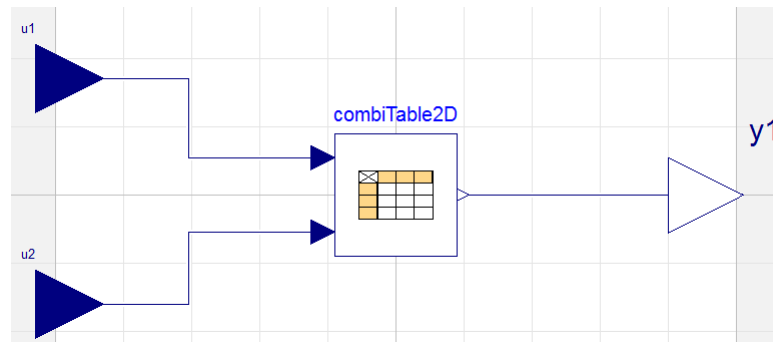
The flag is by default false.

(String variables are however presently not supported.)

Example

String parameter support can be illustrated by a simple example of changing tables for an FMU; consider creating a simple model for linearization.

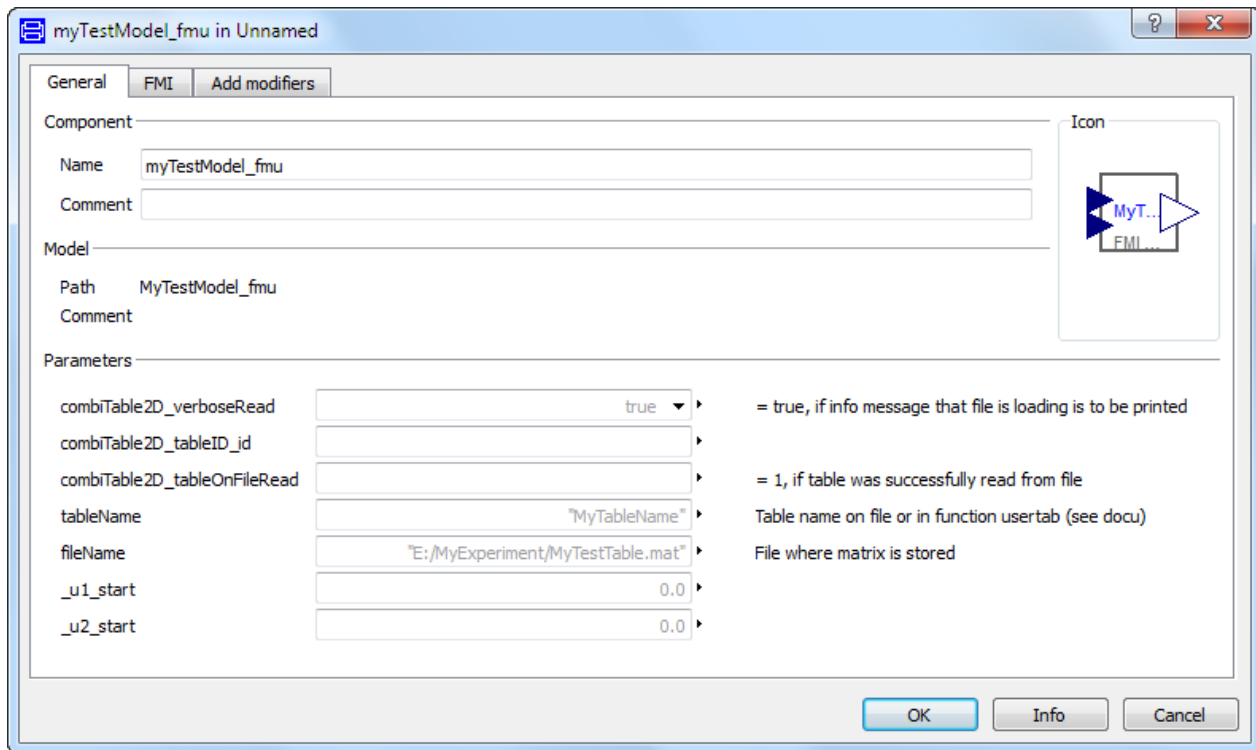
Create a model; drag an instance of `Modelica.Blocks.Tables.CombiTable2D` into the model. Connect the two inputs and the output and create the corresponding connectors. The result is:



In the parameter dialog of `combiTable2D`, select **tableOnFile** to true, and propagate **tableName** and **fileName**. Give relevant default values for them. As an example, looking at the resulting Modelica code when having specified a table name and file name as default value, we find:

```
model MyTestModel
  parameter String tableName="MyTableName"
    "Table name on file or in function usertab (see docu)";
  parameter String fileName="E:/MyExperiment/MyTestTable.mat"
    "File where matrix is stored";
equation
  ...
end MyTestModel;
```

Saving the model, and then generating an FMU from it (do not forget to set the flag above), we can import this FMU and look at the resulting parameter dialog of an instance of that FMU:



This FMU supports changing the table name and file name as string parameters.

Extended example (resource handling on Windows)


If the FMU should contain the table as a resource, the following can be done, in Windows:

Rename the parameter **fileName** to **includeFileInFMU** (really not needed, but for clarity). Use, in the variable definition dialog of **includeFileInFMU**, in the default value input field, the context command **Insert Function Call...** to access `Modelica.Utilities.Files.loadResource`, and specify the file name. The resulting code is (given a new model `MyTestModel2` is created):

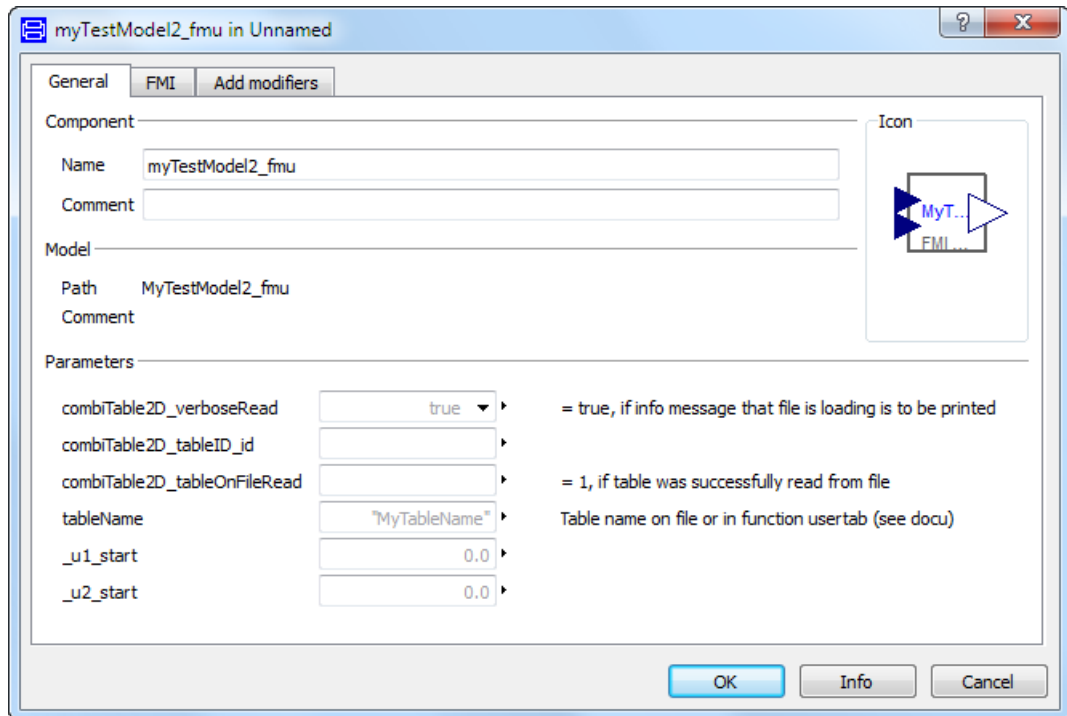
```
model MyTestModel2
>
  parameter String tableName="MyTableName"
    "Table name on file or in function usertab (see docu)";

  parameter String includeFileInFMU=Modelica.Utilities.Files.loadResource("E:/MyExperiment/MyTestTable.mat")
    "File where matrix is stored";
equation
>
>
end MyTestModel2;
```

Save the model. Before generating the FMU, check:

- that `Advanced.AllowStringParameters=true`.
- that **Copy resources to FMU** is ticked in the **FMI** tab of the simulation setup. (The simulation setup can be reached in Simulation mode by the command **Simulation > Setup...** or by the command button .)

We can import the generated FMU and look at the resulting parameter dialog of an instance of that FMU:



The **includeFileInFMU** parameter is not displayed, it is evaluated, and the corresponding file has been copied to the Resources directory of the FMU.

Online tunable parameters supported for FMI 2.0 RC1

Online tunable parameters are now supported for FMI 2.0 RC1. (Such parameters are not defined in FMI version 1.0.)

Structured declaration of variables supported for FMI 2.0 RC1

In Dymola 2015 structured declaration of variables is supported also for FMI 2.0 RC1; it was supported for FMI 1.0 already in the previous version.

64-bit export of FMUs on Linux

Consult www.Dymola.com/Linux for information about 64-bit Linux support.

Improved import of FMUs with many inputs/outputs

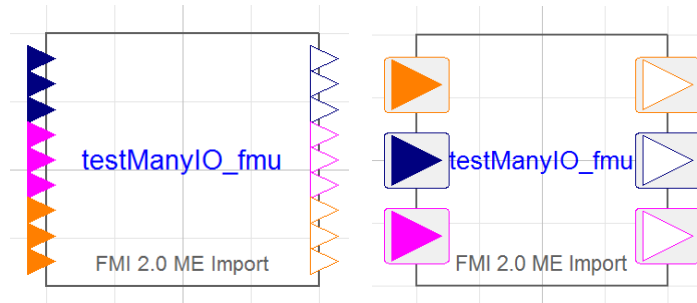
When importing FMUs with many inputs/outputs, the import is improved by setting the flag

```
Advanced.FMI.OverlappingIO=true
```

The flag is by default `false`.

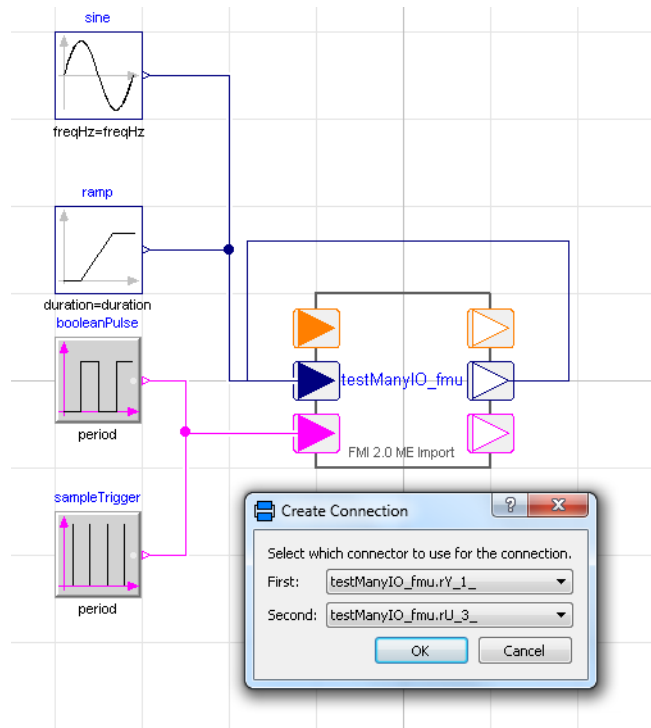
Setting this flag, the following two features are enabled:

- The input and output connectors of the imported FMU are stacked at the same location, one location for each type (Integer, Real, and Boolean) of connectors (the image to the right below).



- Dragging a connection from/to a stacked connector displays a dialog to conveniently select what connectors to connect.

The result can be illustrated as:



Note! The feature “Smart Connect” is not supported when using this feature, so when setting the flag above, also the corresponding Smart Connect flag should be set to the status:

```
Advanced.SmartConnect=false
```

(This flag is by default true.)

The above figure shows the stacking in groups of the connectors of the imported FMU, and also the dialog that is shown when dragging a connection from (“First”) the Real output group of the FMI to (“Second”) the Real input group of the FMI, and having selected (by the drop-down list) to connect the first connector in the Real output group to the third connector in the Real input group. The connection is done when clicking **OK**.

This connection dialog for overlapping connectors is also supported for other components than FMUs.

Minor improvements

Hidden and protected variables by default not generated in FMU export

In Dymola 2015 hidden and protected variables are by default not generated in modelDescription.xml when exporting an FMU. If they should be generated, the following flag setting must be applied:

```
Advanced.FMI.xmlIgnoreProtected = false;
```


The flag is true by default.

Improved handling of start values for initialization

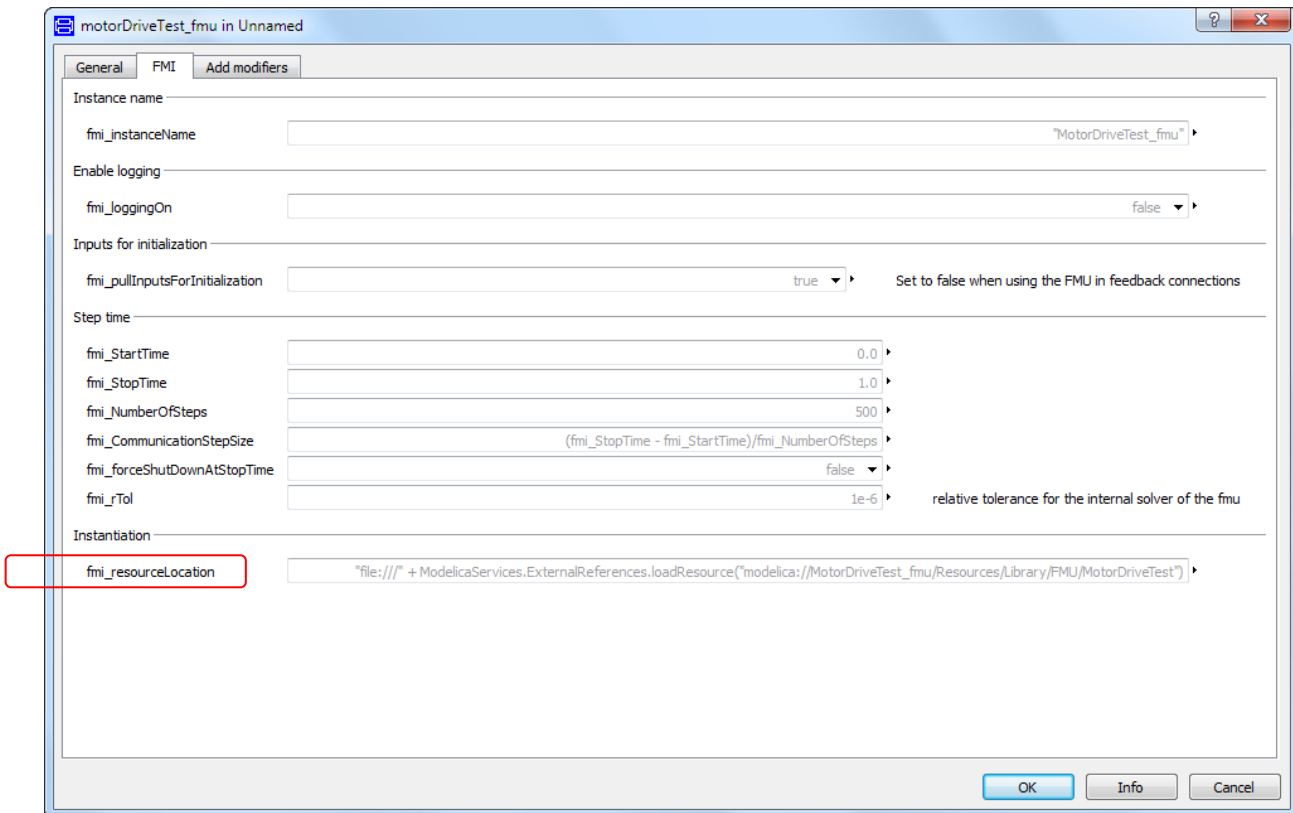
In Dymola 2015 start values can be set for FMI Model Exchange, to set input start values before initialization. This can be useful when wanting to avoid e.g. division by zero when initializing.

Improved error messages for FMU import

The error messages for FMU import have been improved; more information is given.

Location of external resources available as parameter

The location of external resources (dlls, tables etc.) is available as a parameter in the FMI tab of the parameter dialog of the FMU:



In most cases, a user does not need this information (in particular with the new resource handling introduced in this version), but it can be valuable if FMUs are moved etc.

(The FMI tab contains different parameters depending on e.g. if a Model Exchange or a Co-Simulation FMU is selected.)

3.6.6 Code and Model Export

Binary Model Export

Generating a dymosim DLL

It is possible today to generate a dynamic link library (dymosim.dll) from a model, if the Binary Model Export (or Source Code Generation) option is available. This is still supported; however, any new development will be to use FMI instead since FMI now supports Co-simulation using Dymola solvers (see page 25). This also means that the possibility to generate a dymosim DLL in the way it is done today will be removed in some future version of Dymola.

3.7 Advanced Modelica Support

3.7.1 Minor improvements

getInstanceName implemented

The function `getInstanceName` is implemented according to Modelica Language Specification, version 3.3. The function returns the name of the active instance, and can be used in e.g. error messages.

spatialDistribution implemented

The function `spatialDistribution` is implemented according to Modelica Language Specification, version 3.3. The function gives a transport delay where you can specify the initial condition, and the function supports reversible flow-direction.

3.8 New libraries

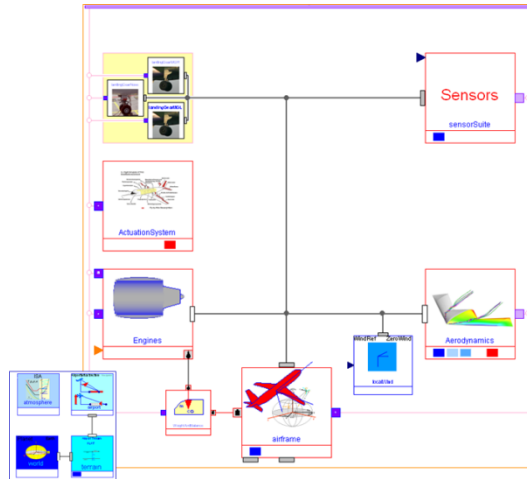
Below is a short description of new libraries. For a full description, please refer to the libraries documentation.

3.8.1 Flight Dynamics Library

The Flight Dynamics library allows for construction of multi-disciplinary flight dynamics models of flight vehicles, like transport and military aircraft, Unmanned Aerial Vehicles, airships, etc.

The environment models provide functionalities to cover on-ground operations up to flight at high speeds and high altitudes. The resulting models may for example be used for design

analysis in various fields and stages of the aircraft development, for flight control law design, as well as for real-time flight simulation.



Typical use cases:

- Aircraft and flight control design.
- Mission simulation and optimization.
- Real-time simulation.

Key features:

- Full six degrees-of-freedom simulation including trim calculation.
- Full compatibility; develop and interconnect airframe and systems models using components from the Modelica Standard Library.
- Detailed environment models and a wide range of compatibilities.

Benefits:

- True multi-physics integrated modeling, control and simulation tool.
- Easily scaled model complexity, tailored to your analyses.
- Reduced modeling costs and early feedback to the design process.
- Improved maintainability by using a single tool for modeling, control and simulation.

3.9 Updated libraries

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

3.9.1 Air Conditioning Library

A minor version 1.8.7 has been released. Some features and improvements:

- Air split and junction models added.
- Improved Modelica compliance.
- Steady-state initialization of heat exchanger models with moisture condensation improved by applying the homotopy operator.

The included conversion script is only needed if the model `ThermoFluidPro.Components.Walls.HXDynamicWall.3Dim` is used in custom models.

3.9.2 DataFiles Library

A minor version 1.0.1 has been released.

3.9.3 Design Library

A minor version 1.0.3 has been released.

3.9.4 Electric Power Library

Version 2.1.1 is a minor version with focus on making the library compliant with the Modelica Language Specification. This included work with enumeration types, consistent type declaration, restructuring of conditional components in sensor models, and use of URIs.

3.9.5 Engine Dynamics Library

Version 1.2.1 is a minor update of the Engine Dynamics Library.

3.9.6 Fuel Cell Library

Version 1.2.1 is a minor version of the Fuel Cell Library with improved documentation and improved stability in the examples. The improved documentation on how to use the library for real-time purposes is based on results from recent projects. Furthermore, a new tutorial section is added, showing the library workflow when configuring a fuel cell stack.

3.9.7 Heat Exchanger Library

Heat Exchanger Library version 1.1 is a major release with new features and other improvements.

Examples of new features:

- Flat tube type air – gas components have been added.
- Flat tube type air – two phase components have been added.
- A new type of fin geometry has been added.
- It is now possible to discretize the flat type internal flow orthogonal to the flow direction.
- Heat exchanger test benches are now included.

In order to support all different heat exchanger types using common base classes, there have been some changes to the internal structure of the model with this release. Some examples:

- A common template `PartialFlatTubeHX` for all flat tube type heat exchangers has been created.
- A separate template for each of air – gas, air – liquid, and air – two phases has been created.
- The geometry records have been structured in a hierarchical way.

The included conversion script supports automatic conversion of user models from versions 1.0 and 1.0.1.

3.9.8 Hydraulics Library

The version 4.0 is a major release, with a number of enhancements and improvements; the following are some examples.

With “Thermo Hydraulics” it is now possible to enable thermo dynamic equations, as well as to disregard the temperature effects. The models of the working fluids are based on polynomials and accurately model fluid properties dependent on temperature and pressure. A large variety of pre-defined thermo-dynamically sound oil models are included. A descriptive guide/tutorial is added on how to get started with Thermo Hydraulics.

New fluid property models of jet fuels have been added, as have new mass flow rate- and temperature sensors.

In the Elements package, a number of modifications to the spool valve component have been made.

The library is significantly improved in terms of ease of use, quality and graphical appearance, some examples are:

- Pressure dependent color on cylinders and chamber, for animation purposes.
- More consistent use of the circle on ports (for visualizing where pressure might be a state).
- Restructured Fluids package; the oils are now easier to find.
- The previous pumps package is now divided into three subpackages: Pumps, Motors, and Sources.
- The friction model in Cylinders is now replaceable.
- `OilVolume`, `MultiPortOilVolume`, and `Chamber` now extend from the same template (`VolumeDynamics`), where all common parameters and equations are located.

Conversion from release 3.3.1 and 3.3.3 to 4.0 is embedded. Note however that some additional actions might be required due to changed port names. Refer to the library documentation for more information.

3.9.9 Hydro Power Library

A major version 2.4 has been released. The release contains two major improvements; a new surge tank model with support for more complex geometries, and a reduction of the simulation speed of approximately 50% due to a more efficient media implementation.

Models that are using Hydro Power Library version 2.3 will automatically be converted to version 2.4.

3.9.10 Liquid Cooling Library

Version 1.2.1 of the Liquid Cooling Library is a minor release.

3.9.11 Model Management Library

A minor version 1.1.2 has been released.

3.9.12 Plot 3D Library

A minor version 1.0.3 has been released.

3.9.13 Pneumatics Library

A minor version 1.6.2 has been released.

3.9.14 Thermal Power Library

The version 1.8 is a major version, with a number of enhancements and improvements. Some examples:

- New discretized gas pipe model ThermalPower.Gas.FlowChannels.Pipe. It can handle reversing flow.
- New wall model with a more userfriendly generic structure.
- New gas and two-phase heat exchanger Generic_gas2ph. It's parametrized by basic geometry parameters.
- New radiation model.
- New generic gas friction loss model. The user can choose from a large variety of pressure loss correlations; including quadratic, linear, loss coefficient based, etc.

Models that are using Thermal Power Library version 1.7 will automatically be converted to Thermal Power 1.8.

3.9.15 User Interaction Library

A minor version 0.63 has been released.

3.9.16 Vapor Cycle Library

Version 1.1 is a major release of the Vapor Cycle Library.

In particular, new components have been added, some examples:

- A pump model with several different options for defining flow and power characteristics.
- A turbine model with flow rate determined according to the Stodola law and a given nominal operating point. Constant isentropic and mechanical efficiency are assumed.
- Splits and junctions with different options to define the flow resistance.
- A working fluid flow source which may be used to remove or add charge in a closed cycle according to a given setpoint.
- A multi-port volume, the number of ports is automatically adapted to the number of connections to this component.
- An interface to the NIST REFPROP data base.

Existing models can be converted to the current version using the supplied conversion script.

3.9.17 Vehicle Dynamics Library

Version 1.9 is a major maintenance release with many changes to the Wheels interface and templates. Some changes are described below.

Support for TYDEX Tyre Interface Tire Models

The Standard Tyre Interface (STI) was defined by the TYDEX working group. The existing built-in tire models in Vehicle Dynamics Library have separate blocks for calculating forces and contact (tire/ground interaction). For STI-compatible tire models, the forces and contact calculations are combined. Further, the Vehicle Dynamics Library tire models calculate the tire forces at the contact patch, and the hub model is responsible for transforming those forces for application at the hub. STI-compatible tire models calculate the forces at the hub.

New Wheel Interface and Templates

The fundamental difference between the Vehicle Dynamics Library tire models and STI-compatible tire models mean that the existing templates for wheels, forces, hubs, and contact calculations in the Vehicle Dynamics Library could not be used. The Vehicle Dynamics Library has now been extended to support STI-compatible tire models that calculate the forces at the hub by adding new wheel, forces, and hub interfaces and templates.

Support for the Delft-Tyre family of tire models

The Vehicle Dynamics Library has been extended to include support for the TNO Delft-Tyre product line which includes MF-Tyre and MF-Swift tire force models. The Delft-Tyre wheel model fully supports Vehicle Dynamics Library built-in ground models, TNO road property files, and OpenCRG road descriptions.

Conversion

A conversion script is needed to accommodate the changes described above. This will convert models from version 1.8 to the current version.

3.10 Documentation

In the software distribution of Dymola 2015 Dymola User Manuals of version “March 2014” will be present; these manuals include all relevant features/improvements of Dymola 2015 presented in the Release notes. Limited Availability (LA) features are not included.

A new paper by Hilding Elmqvist: “Modelica Evolution – From My Perspective” is available by the command **Help > Documentation**. The paper describes the history of Modelica and Dymola from the author’s perspective.

3.11 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2015 are listed.

3.11.1 Hardware requirements/recommendations

Hardware requirements

- At least 1 GB RAM
- At least 400 MB disc space

Hardware recommendations

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

A dual processor will be enough; the simulation itself uses only one execution thread so there is no need for a “quad” processor.

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 2-4 GB of RAM for 32-bit architecture and 3-6 GB of RAM for 64-bit architecture.

3.11.2 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions

Dymola 2015 is supported, as 32- and 64-bit application, on Microsoft Windows XP, Windows Vista and Windows 7. 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 2015 on Windows:

Microsoft C/C++ compilers, free editions:

- Visual Studio 2008 Express Edition (9.0)
- Visual C++ 2010 Express (10.0)
- Visual Studio 2012 Express Edition (11.0)

Microsoft C/C++ compilers, professional editions:

- Visual Studio 2005 (8.0)
- Visual Studio 2008 (9.0)
- Visual Studio 2010 (10.0)
- Visual Studio 2012 (11.0)

GCC compilers

Dymola 2015 has limited support for the MinGW GCC compiler, with a GCC version compatible with 4.8. For the list of current limitations; refer to section “Limitations” on page 21.

Dymola license server

For a Dymola license server on Windows, all files needed to set up and run a Dymola license server on Windows, except the license file, are available in the Dymola distribution. (This includes also the license daemon, where Dymola presently supports FLEXnet Publisher version 11.11. This version is part of the Dymola distribution.)

Linux

Dymola versions on Linux and operating system versions, and compiler

Dymola 2015 runs on SUSE Linux (Release 11), 32-bit and 64-bit, with gcc version 4.3.4, and compatible systems. In addition to gcc, the model C code generated by Dymola can also be compiled by clang. To change compiler, change the variable `CC` in `/opt/dymola/insert/dsbuild.sh`.

Dymola 2015 is supported as a 32-bit application on Linux. For information related to the 64-bit Linux version of Dymola 2015 (including 64-bit FMU export), please consult the web page www.Dymola.com/Linux.

Note on Optimization library

Please note that you have to use the Optimization library version 2.x or higher to use multi-criteria design optimization on Linux; the older Design.Optimization package does not support multi-criteria design optimization on Linux.

Dymola license server

For a Dymola license server on Linux, all files needed to set up and run a Dymola license server on Linux, except the license file, are available in the Dymola distribution. (This also includes the license daemon, where Dymola presently supports FLEXnet Publisher version 11.9. This version is part of the Dymola distribution. For upgrade to version 11.11, see www.Dymola.com/Linux.)