## Contents

1. Important notes on Dymola .............................................................. 5
2. About this booklet ............................................................................. 5
3. Dymola 2018 ...................................................................................... 6
   3.1 Introduction .................................................................................. 6
   3.1.1 Additions and improvements in Dymola ....................................... 6
   3.1.2 New and updated libraries ......................................................... 6
3.2 Developing a model .......................................................................... 8
   3.2.1 Support for Modelica 2 in Dymola will be discontinued ............... 8
   3.2.2 Improved handling of current working directory and startup directory ........................................ 8
   3.2.3 Improved code completion in Modelica Text editor and parameter dialog ................................... 9
   3.2.4 Easier to create test-driven template models ................................ 9
   3.2.5 Improved dialog for working with annotations ........................... 13
   3.2.6 UTF-8 conformance checks of Modelica files ............................. 17
   3.2.7 Minor improvements ............................................................... 19
3.3 Simulating a model ........................................................................ 34
   3.3.1 Improved handling of the current working directory and the startup directory ..................... 34
   3.3.2 Sparse solver support for more solvers in Dymola ................. 37
   3.3.3 Support for Rosenbrock solvers for inline integration ................. 38
   3.3.4 Improvements in Variable Browser .............................................. 39
   3.3.5 Improved model debugging ....................................................... 43
   3.3.6 Plot window ........................................................................... 49
   3.3.7 Scripting ............................................................................... 50
   3.3.8 Minor improvements ............................................................... 52
3.4 Installation ....................................................................................... 60
   3.4.1 Dymola as a 32-bit application will be discontinued ..................... 60
   3.4.2 Installation on Windows ............................................................ 60
   3.4.3 Installation on Linux ................................................................. 60
3.5 Model Management ........................................................................ 60
3.6 Other Simulation Environments ...................................................... 61
   3.6.1 Dymola – Matlab interface ......................................................... 61
   3.6.2 Real-time simulation ............................................................... 62
   3.6.3 OPC Communication ............................................................... 63
   3.6.4 Java Interface for Dymola ......................................................... 64
   3.6.5 Python Interface for Dymola .................................................... 64
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 2018:

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 91 for more information.

Intel
Dymola 2018 has limited support for the Intel Parallel Studio XE compiler. For more information about this compiler, see section “Compilers” on page 91; the section about Intel compilers.

GCC
Dymola 2018 has limited support for the MinGW GCC compiler, 32-bit and 64-bit. For more information about GCC, see section “Compilers” on page 91; the section about GCC compilers.

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 93 for more information.

2 About this booklet
This booklet covers Dymola 2018. 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 2018

Note! Dymola 2018 is the last version that is available as a 32-bit application. Dymola 2018 FD01 and later versions will only be available as 64-bit applications. See section “Dymola as a 32-bit application will be discontinued” on page 60 for more information.

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2018. In particular, Dymola 2018 provides:

- A number of new libraries (list on page 6).
- Support for the new Modelica Language version 3.4 (page 69)
- Improved creation of test-driven template models (page 9)
- Improved dialog for working with annotations (page 13)
- UTF-8 conformance checks of Modelica files (page 17)
- Improved handling of the current working directory and the startup directory (page 34)
- Sparse solver support for more solvers in Dymola (page 37)
- Rosenbrock solvers for inline integration (page 38)
- Saving variable start values to a new model (page 39)
- Option to save only the variables in the currently active plot window (page 40)
- Improved model debugging
  - Enhanced function to plot variable dependencies (page 43)
  - Analyzing numeric integration (page 44)

3.1.2 New and updated libraries

New libraries

The following libraries are new in this Dymola version:

- Thermal Management Demos Library, version 1.0
- Thermal Systems Library, version 1.0.0
- Thermal Systems Mobile AC Library, version 1.0.0
- Vehicle Systems Modeling and Analysis (VeSyMA) Library, version 2017.1
- Vehicle Systems Modeling and Analysis (VeSyMA) – Powertrain Library, version 2017.1
• Vehicle Systems Modeling and Analysis (VeSyMA) – Suspensions Library, version 2017.1

For more information about the new libraries, please see the section “New libraries” starting on page 71.

**Updated libraries**

The following libraries have been updated:

• Air Conditioning Library, version 1.14
• Battery Library, version 1.4.1
• Brushless DC Drives Library, version 1.0.1
• Cooling Library, version 1.1
• Design Library, version 1.0.6
• Dymola Commands Library, version 1.3
• Electric Power Library, version 2.3
• Electrified Powertrains Library (ETPL), version 1.1
• Engine Dynamics Library, version 1.2.8
• Engines Library, see Vehicle Systems Modeling and Analysis (VeSyMA) – Engines Library
• Fuel Cell Library, version 1.5
• Heat Exchanger Library, version 1.6
• Heating, Ventilation, and Air Conditioning Library (HVAC Library), version 2.3.0
• Human Comfort Library, version 2.3.0
• Hydraulics Library, version 4.6
• Hydro Power Library, version 2.6.3
• Liquid Cooling Library, version 1.5.3
• Pneumatics Library, version 2.2
• Thermal Power Library, version 1.14
• Vapor Cycle Library, version 1.5
• Vehicle Dynamics Library, version 2.5
• Vehicle Systems Modeling and Analysis (VeSyMA) – Engines Library, version 2017.1
• Wind Power Library, version 1.0.1

For more information about the updated libraries, please see the section “Updated libraries” starting on page 76.
3.2 Developing a model

3.2.1 Support for Modelica 2 in Dymola will be discontinued

Modelica 3.0 was standardized in September 2007 and Modelica Standard Library 3.0 was introduced in March 1, 2008. Modelica 2 traces its origins back to 2004.

The support for Modelica 2 in Dymola will be discontinued from Dymola 2018 FD01. Neither simulation nor editing of Modelica 2 models will be supported. During a transition period conversion of models in Modelica 2 to Modelica 3 is supported.

<table>
<thead>
<tr>
<th>Dymola release</th>
<th>Support status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dymola 2018</td>
<td>No support for editing or simulating Modelica 2 models.</td>
</tr>
<tr>
<td></td>
<td>Automatic syntax conversion from Modelica 2 to Modelica 3 supported.</td>
</tr>
<tr>
<td></td>
<td>Modelica Standard Library (MSL) version 2.2.2 not installed, but available separately if needed, together with the Modelica 2-compatible libraries¹.</td>
</tr>
<tr>
<td>Dymola 2018 FD01</td>
<td>No conversion from Modelica 2.</td>
</tr>
<tr>
<td></td>
<td>Modelica Standard Library (MSL) 2.x not available.</td>
</tr>
</tbody>
</table>

There are several reasons for upgrading to a recent version of the Modelica language and its libraries, but the effort is not trivial and requires some planning. If support is needed, please contact support by [http://www.3ds.com/support](http://www.3ds.com/support).

3.2.2 Improved handling of current working directory and startup directory

More commands are now available for easier accessing and changing the current working directory and setting the startup directory. For setting the startup directory, there is also a new possibility in Edit > Options..., the Settings tab. Since the new commands are also available from a new toolbar button in the Commands window, the features are described in the next chapter, see section “Improved handling of the current working directory and the startup directory” starting on page 34.

---

¹ Modelica 2.2.2 is included in the file extra\CompatibilityLibraries MSL 2.2.2.exe that is included in the Dymola media. (Modelica 3.2.1 is in a similar way included in extra\CompatibilityLibraries MSL 3.2.1.exe.)
3.2.3 Improved code completion in Modelica Text editor and parameter dialog

The below improvements applies to the Modelica text editor and the parameter input lines in the parameter dialog. For corresponding improvement in the script editor, see “Improved code completion in the script editor” on page 52.

Improved selection of suggestions

In Dymola 2018, the code completion is not anymore based on general text completion, it is now contextual, that means using code completion in a class now only suggests:

- Modelica keywords (highlighted in blue).
- All variable names defined in the class.
- Names of open packages.
- Names of other classes available at the code completion level (models, functions etc.).

Improved navigation

As in previous versions, pressing Ctrl+Space displays a code completion list. In Dymola 2018, if you press Ctrl+Space again, or keeping Ctrl pressed and press Space again, the next alternative in the code completion list is selected.

3.2.4 Easier to create test-driven template models

The structured way of building template models (by finding a suitable base model, e.g. a base interface model, adding that base class, making it replaceable, making a subsystem, redefine and test the complete system) is well supported.

However, it is also possible to build a test-driven template model, by:

1. Testing a complete system.
2. Making a subsystem (using the context menu command Split Model).
3. Making sure the subsystem is based on a base class, e.g. some common interface base class, by
   a. Finding suggestions for base classes.
   b. Adding the selected base class.
4. Making the subsystem component replaceable, including defining the constraining clause of the template model component to be the selected base class.

(Note that also minor cleanup is needed, e.g. adding icons, moving connectors.)

The items 3a, 3b, and 4 in the list above have been improved according to the following sections.)
**Search for base class suggestions when creating a template model**

In Dymola 2018 a new command, and corresponding option, is available to suggest base classes for a selected class.

As an example, assume we want to create a template from the three first components in the Coupled clutches demo. To make this demo model editable, first duplicate it by the command **New > Duplicate Class**…. Then select the three first components. A submodel is created from this selection by right-clicking and using the **Split Model** command, naming the submodel **MyEngine**, then right-clicking that submodel in the package browser, selecting the new command **Suggest Base Class**…,
This command opens the following dialog, with the new option **Suggest base class** selected. When you have clicked **Search** the result is:

![Search dialog]

(It may be noted that the last found suggestion in the figure above creates a heavy penalty since the fixed is inherited and also exists locally.)

The feature corresponds to 3a in the list above.

**Adding a selected base class when creating a template model**

The results in the dialog above can be double-clicked to get possibility to adapt the template to the selected base class. Double-clicking any of the alternatives in the above list (we select the first one) gives:

![Advanced base class dialog]

This menu is new in Dymola 2018. The default selection here is to perform the actions needed to adapt the model that should be the template to this base class. The selection **Just add base-class** gives no additional actions; it corresponds to dragging a class from the
package browser to the component browser like in the previous versions when the new menu was not present.

This feature corresponds to 3b in the list above.

**Defining the constraining clause of the template model component to be the selected base class**

To make the template model component replaceable and define the constraining clause as the base class selected for the template model, right-click the template model component in the model where it was first defined (in our case a copy of the Coupled clutches demo) and select **Parameters**. In the Attributes tab, select **Replaceable**, and select the constraining clause to be the used base class:
The possibility to select the base class as constraining clause is new in Dymola 2018. This feature corresponds to item 4 in the list above.

### 3.2.5 Improved dialog for working with annotations

The dialog for working with annotations has been improved. The dialog can be used for top-level annotations and all annotations for all objects that are represented with graphics. (It cannot be used for variables and parameters that do not have any graphical representation.)

**Working with model annotations**

To open the annotations dialog for model annotations, right-click on the background (with nothing selected) and select **Annotation**. An example of dialog:

![Annotation dialog example](image)

In Dymola 2018, the following apply:

- All model annotations are shown.
- The part corresponding to the active layer when opening the dialog is shown expanded (in this example Diagram); annotations for the other graphical layers (in this example Icon and Documentation) are collapsed.
• You can create, edit, rename, change order (up/down), and delete annotations, including top-level ones, using either a context menu or the toolbar buttons in the bottom of the dialog:

For a detailed description on how to edit annotations, see “Details in working with annotations” below.

**Working with component annotations**

To work with component annotations, select or multi-select components, right-click and select **Annotation**. An example of a dialog from a multi-selection of connected components and some graphical objects:
In Dymola 2018, the following apply:

- All annotations for selected components, connections, and graphical primitives are shown.
- The root for connection annotations is the connection string; the root for component annotations is the component name.
- You can create, edit, rename, change order (up/down), and delete annotations, like for model annotations (see above). For a detailed description on how to edit annotations, see next section. There are some exceptions, in this dialog you cannot:
  - Change the root names of connection annotations and component annotations
  - Create top-level annotations
  - Change the order of the top-level annotations. However, you can change the order of the children of such annotations.
**Details in working with annotations**

**Adding top-level annotations**

You can add top-level annotations when you work with model annotations. In that dialog, click below the present annotations (you might have to make the window longer) and then click the Add toolbar button (or right-click and select Add Annotation).

You can now enter a name for the new node. An error message appears when you click Apply or OK if you try to apply a name that is not legal Modelica.

Nodes that are added on the top level only have a name. Nodes added further down also have a value.

**Adding children to an annotation**

To add children to an annotation without value, select the node and click Add (or right-click the node and select Add Annotation).

To be able to add children to an annotation with a value, first remove the value.

The first thing to do after adding an annotation (child) is to rename the new annotation.

If you add an annotation to a node that is a graphical primitive, component or connection, you will have a dropdown list to select from when you are to specify the name of the annotation:
Note. The connection Text has different attributes than the Text primitive.

Annotations not being top-level ones must have values. You get `<value>` when creating the annotation; you must yourself change this to a proper value. Presently there are no dropdown lists available for the values, only text boxes.

### 3.2.6 UTF-8 conformance checks of Modelica files

Starting with Dymola 2018, Modelica files are checked for valid UTF-8 encoding. In case of encoding errors, warnings are displayed in the Syntax Error tab of the Dymola Messages window.

For backwards compatibility, UTF-8 encoding errors are presently just warnings.

Encoding errors are by default handled by automatically transforming erroneous byte sequences to correct UTF-8 multi-bytes, if possible. This applies to outdated ISO/IEC 8859-1:1998 encoded files (commonly mislabeled as ISO-8859-1 or just Latin-1). Such bytes with values greater or equal to 0xA0 and followed by an ASCII character are by default automatically transformed to their respective UTF-8 multi-bytes. Byte sequences that cannot be auto-corrected are removed.
The alternative is to remove all erroneous byte sequences instead of trying to convert them. This can be done by setting the flag

```csharp
Advanced.ParserAutoFixesLatin1Errors = false;
```

(The flag is by default true.)

**Note!** Since saving a model containing encoding errors will change it according to the alternatives above, it is important to investigate the encoding errors first, before doing any further model changes.

A typical session with auto-corrected and non-correctable encoding errors is:

The same session with the flag `Advanced.ParserAutoFixesLatin1Errors` set to `false` is:
### 3.2.7 Minor improvements

**Top-level parameters can be displayed and edited**

In Dymola 2018 top-level parameters of a model can be displayed and (if the model is not read-only) edited. To display the parameter dialog for top-level parameters, open the top level by double-clicking the background, and right-click and select Parameters.

**Changing name when propagating a parameter already propagated with the same name**

In Dymola 2018 it is possible to change the name of a parameter when propagating it, even if that parameter is already propagated to the same name from another component instance.

This can be done by using, in the parameter input field, the context menu entry Propagate… which has been enhanced to allow this. For the previous behavior to propagate the parameter to the same parameter as the first one propagated, a new context menu entry Propagate <parameter name> is available.
In the above example, if the parameter $p$ has already been propagated to $p$ from `model2_1`, the new entry Propagate $p$ just propagates to $p$ like before (corresponding to the previous functionality of Propagate...). The entry Propagate... has in Dymola 2018 been improved to enable changing the name $p$ to a new name before propagating.

(The first time a parameter is propagated, both entries work the same.)
Closure types of circle sector/elliptic arc available

When creating or editing a circle sector/elliptic arc, the closure type can be set. When creating a circle sector/elliptic arc, the menu is, for example:

The selections None, Chord or Radial will create the following for a circle sector/elliptic arc 0 - 300:

The selection Default means that when the start angle is 0 and the end angle is 360, Chord is selected, Radial otherwise.

When right-clicking a present circle sector/elliptic arc, the command Elliptic Closure can be used to edit the closure type:
Updated color dialog for graphical objects

The color dialog for line color and fill color is updated:

- The command **Color > None** corresponds to the previous **No Color** command.
- Dymola default colors have been added in the dialog, they correspond to the default colors for curves in the Plot window (as seen in the entry **Color** in the context menu for a curve in that window).

Updated line style dialog for graphical objects

The line style dialog has been updated with the **Line Style > None** command.
Improved font selection for text primitives

It is now possible to select serif, sans-serif, and monospace when selecting text fonts:

![Font selection dialog box]

The fonts used for Windows and Linux are:

<table>
<thead>
<tr>
<th>Font Type</th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serif</td>
<td>Times New Roman</td>
<td>Times</td>
</tr>
<tr>
<td>Sans-serif</td>
<td>Arial</td>
<td>Helvetica</td>
</tr>
<tr>
<td>Monospace</td>
<td>Courier New</td>
<td>Courier</td>
</tr>
</tbody>
</table>

Enhanced formats of tokens expanded in text strings

Previously the format when entering tokens to be expanded in text strings was %, for example %name to display the name of the enclosing class.
This format is still supported, but now also curly brackets are supported in specifying such tokens, for example \%{name}. This increases clarity, and also makes it possible to avoid blanks, for example, to use \%{MyRecord.a}m to display the record component a with unit m.

**Option when to display dialog when rearranging classes in the package browser**

There is now an option when to display the Move dialog or the Copy dialog when you rearrange classes in the package browser, either by context commands or by drag and drop. The option is available by the command Edit > Options…, in the General tab:
The alternatives are:

- **Never**: Never display the Move or the Copy dialog
- **Only when different top level package**: Display the Move dialog or the Copy dialog when moving or copying a class to another top level package. This is the default setting.
- **Always**: Always display the Move dialog or the Copy dialog.

The alternatives correspond to setting the flag `Advanced.ShowRearrangeDialog` to 0, 1, or 2, respectively.

The setting is saved between sessions.

An example of the Move dialog is:
Option to include non-graphical objects in the component browser moved to Edit > Options

The option if to include non-graphical objects in the component browser is moved to the General tab of the Edit > Options… command:

The default value is as previously to not include them.

The setting is now saved between sessions.

Added alternatives when double-clicking a component in the diagram layer

In Dymola 2018 you can select to open the class of a component when you double-click it in the diagram layer. You can open the class in the same tab or in a new tab. The option is available by the command Edit > Options…, in the Graphical Editor tab:
The setting is saved between sessions.

**Variable declaration dialog when double-clicking variables in the component browser**

If variables are displayed in the component browser (see section “Option to include non-graphical objects in the component browser moved to Edit > Options” on page 27 for the setting to display variables in the component browser), double-clicking on such a variable displays the variable declaration dialog for that variable.

**More alternatives what to display when using Find Connection or Create Connection**

When using the **Find Connection** command (as **Edit > Find Connection** or a context command for a connection) or when using the **Create Connection** dialog when for example connecting to expandable connectors, there is now a new dropdown list selection **Show items**. The example below is a **Find Connection** dialog, for an example of **Create Connection** dialog, see next section.
The selections are:

- **All** displays all items, this selection corresponds to what was displayed in previous versions.
- **In highlighted group** displays only the variables in the group where the hit is. This is the default selection.
- **Only highlighted** display only the hits and the path to them
- **Filter instead of highlight** is similar to the above one, but filters instead of highlights.
Adding variable in Create Connection dialog now possible to have on context command

In previous versions, `<Add Variable>` to add a variable to, for example, an expandable connector was always a visible alternative. This is still the default behavior, but you can change this to instead use a context command of the relevant component/subcomponent to display the `<Add Variable>` alternative before you can use it.

The old behavior, the default one:

![Old behavior diagram]

The new behavior, if activated:

![New behavior diagram]
To activate the new behavior, set the flag

```
Advanced.AddVariableRequireContext = true;
```

(The flag is by default false.)
Ordering available for connection lines

In the context menu of a graphical connection Order commands are added. The commands are the same as for components:

Easier to cascade or tile windows

The commands for cascading or tiling windows now have shortcuts (see image), and they are now also available by activating a new Window toolbar by Window > Tools > Window. The image below shows this toolbar activated. The toolbar can also be activated in Simulation mode.
Shortcuts for model navigation

The **Back** and **Forward** command was earlier only accessible as command buttons, now the mouse back and forward buttons can be used also.

Changed handling of Scientific Data Format (SDF) files

Due to Microsoft already having reserved the .sdf file extension for SQL Server Compact Edition Database File, the automatic association for .sdf files with the SDF Editor is not done anymore.

You have to do this association manually if you want, for example, that the file should be opened in the SDF Editor when you double-click the file in the browser. To do the association, right-click the file in a file browser, and select **Properties**. In the General tab, click **Change** and select **SDF_Editor.exe** if available. If not available, browse to `...Program Files (x86)\bin\SDF_Editor.exe` and select that one.

Note that if you have used previous versions of Dymola, .sdf files are still automatically associated with the SDF Editor. If you get problems because you are to use SQL Server Compact Edition Database files, you must change the file association manually (like above) to associate them with the SQL software.
3.3 Simulating a model

3.3.1 Improved handling of the current working directory and the startup directory

New commands

More commands are now available for easier accessing and changing the current working directory, and for defining the startup directory. These commands are available both from File > Working Directory and from a new command button in the Commands window:
Clicking the arrow on the **Working Directory** button in the Commands window gives:

The **File > Working Directory** commands are available in Modeling mode as well.

The new commands are:

- **Use as Startup Directory** defines the current working directory as the startup directory, that is, the working directory to be used when starting Dymola the next time. This command is related to a setting in **Edit > Options...**, the **Settings** tab. See next section.
- **Copy Path** copies the path of the current working directory to the clipboard.
- **Open in File Browser** opens the current working directory in the file browser, for example, in Windows:

  ![File Browser](image)

- **Open in Command Prompt** opens the current working directory in a command prompt:
New setting to define the startup directory

The settings reached by the command Edit > Options..., the Settings tab, have been improved to better handle the startup directory, that is, the working directory to be used when starting Dymola the next time:
The two first radio buttons were available in previous version as well, as a checkbox **Startup directory** in the Save between sessions group. The last button, with input field and Browse button, is new in Dymola 2018.

The above selection corresponds to selecting **File > Working Directory > Use Current Directory as Startup** in the first image in the above section. (The current working directory in that image is E:\Experiment\Test.)

The default setting is **Save last working directory**.

### 3.3.2 Sparse solver support for more solvers in Dymola

Dymola 2018 supports sparse solver for the following solvers in Dymola:
- Cvode (supported already in Dymola 2017 FD01)
- RadauIa
- Esdirk23a, 34a, 45a
- Sdirk34hw

The sparse solver uses a multithreaded (OpenMP) variant of SuperLU.

The sparse solvers can only be activated if the model is sparse and large enough. You will automatically get a message in the translation log of a model if a model is suitable for sparse solvers:

As said in the message, you can activate the sparse solvers with the flag:

```plaintext
Advanced.SparseActivate = true;
```

The flag is by default false.
The flags defining if a model is suitable or not for sparse solvers are already tuned to good values, we don’t recommend changing them. The flags are (the last one in percentage):

Advanced.SparseMinimumStates = 90;
Advanced.SparseMaximumDensity = 5;

Note. You can store the selected algorithm in the model, but you cannot store the value of the flag in the model.

### 3.3.3 Support for Rosenbrock solvers for inline integration

The Rosenbrock methods are now supported as inline integration methods. Rosenbrock has been added as a selection when you select the inline method in the simulation setup. You can reach this setup by the command **Simulation > Setup…**, the **Realtime** tab:
The order can be set to 1 – 4.

The use of Rosenbrock corresponds to the flag `Advanced.InlineMethod = 7;`.

### 3.3.4 Improvements in Variable Browser

**Extending to a new model when saving start values to model**

In Dymola 2018 it is possible to extend to a new model when saving start values to model from the context menu in the variable browser. By default the start values are saved to the current model (as in earlier versions), but a dialog appears with possibility to extend to a new model and save the changed value to that model. Below this option has been ticked, and the name of the model to save to has been changed to MyCoupledClutches:

The new extended model is created when clicking **OK**.
This feature makes it possible to, for example, stop a simulation on the fly, and then store the present start values of e.g. state variables to a new model, and then start a new simulation from that model. (This was previously also possible using the Simulation > Continue > Import Initial… command.)

**Exporting only variables in currently active plot window**

In Dymola 2018 you can export only the variables in the currently active plot window by the top level nodes context command Export Result > Only Plot Window… in the variable browser:

![Variable Browser and Export Result Context Menu](image)

Giving this command gives the usual save result dialog:
The result file can be saved as .mat, .csv, .txt, and .sdf.

(Concerning the commands **Export Result > All...** and **Export Result > Only Current Diagram...**, they correspond to previously available commands that now have been somewhat restructured. See section “Export result commands restructured in the variable browser top-level nodes context menu” on page 56.)
Improved indication of vectors

Vectors are no more indicated by brackets [ ] in the variable browser, they are indicated by the icon 🔄. An example from the Furuta pendulum demo:

<table>
<thead>
<tr>
<th>Variable Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Furuta</td>
</tr>
<tr>
<td>world</td>
</tr>
<tr>
<td>R1</td>
</tr>
<tr>
<td>B1</td>
</tr>
<tr>
<td>frame_a</td>
</tr>
<tr>
<td>frame_b</td>
</tr>
<tr>
<td>animation</td>
</tr>
<tr>
<td>r</td>
</tr>
<tr>
<td>r_shape</td>
</tr>
<tr>
<td>lengthDirection</td>
</tr>
<tr>
<td>widthDirection</td>
</tr>
<tr>
<td>length</td>
</tr>
<tr>
<td>width</td>
</tr>
<tr>
<td>height</td>
</tr>
<tr>
<td>innerWidth</td>
</tr>
<tr>
<td>innerHeight</td>
</tr>
<tr>
<td>density</td>
</tr>
<tr>
<td>color</td>
</tr>
<tr>
<td>specularCoefficient</td>
</tr>
<tr>
<td>r_0</td>
</tr>
<tr>
<td>der(r_0)</td>
</tr>
<tr>
<td>y_0</td>
</tr>
<tr>
<td>v_0</td>
</tr>
<tr>
<td>der(v_0)</td>
</tr>
<tr>
<td>a_0</td>
</tr>
<tr>
<td>angles fixed</td>
</tr>
<tr>
<td>angles_start</td>
</tr>
<tr>
<td>[1]</td>
</tr>
<tr>
<td>[2]</td>
</tr>
<tr>
<td>[3]</td>
</tr>
<tr>
<td>sequence_start</td>
</tr>
</tbody>
</table>
3.3.5 Improved model debugging

Model debugging has been improved by enhancing the function to plot variable dependencies, and introducing a tool to inspect the statistics from the numerical integration.

Enhanced function to plot variable dependencies

The function to plot variable dependencies has been enhanced. Instead of plotting all dependencies immediately (potentially creating a number of plot windows), a browser showing the dependency expression and all associated variables is shown. The variables are grouped by top-level component.

As an example, simulating the Coupled clutches demo using the command Commands > Simulate and Plot, and right-clicking the variable J4.a in the variable browser, then selecting Plot Dependencies now gives:

For Type/Unit, common long names are for cosmetic reasons abbreviated by removing any occurrence of Modelica.SIunits, so, for example, Modelica.SIunits.Torque becomes just Torque.
Right-clicking a variable in the dependency browser, for example w_rel, gives:

![Dependency Browser Context Menu](image)

The commands available in this context menu are:

- **Open** displays the plot dependencies for the selected variable. Pressing the **Back** button returns to the previous view.
- **Plot** plots the selected variable in the currently active plot window.
- **Plot New Window** creates a new plot window and plots the variable in this window. The dependencies of that variable is now displayed in the dependency browser.

**Analyze Numeric Integration**

**Note:** To use this function, you must enable the setting **Which state that dominate error** in the simulation setup, reached by the command **Simulation > Setup…**, the **Debug** tab.

Statistics from the numerical integration is available, facilitating inspection of state variables that most likely slow down the simulation of large models. It is available by right-clicking on the simulation result in the variable browser and selecting **Analyze Numerics** from the context menu:
The window that opens shows the following information (it can be sorted in ascending or descending order by clicking on any of the column headings).

- The state variable name.
- The number of times the variable **limits the step size** of the numerical integrator. Counts greater than zero are highlighted.
- The number of times the variable **dominates the error**, i.e. has an integration error larger than any other state variable. Counts greater than zero are highlighted.
- The number of times the variable contributes to **more than 10 %** of the integrator error.
Right-click on a variable displays a context menu with two commands:
Plot Error creates a new plot window and displays the time-plot for the variable, as well as the information regarding limits step size, dominates error, and more than 10 % of error.

However, in the plot a marker is plotted at each time point where such an event occurs, and the value on the right vertical axis is the contribution to the numeric error in percent. This plot facilitates a better understanding of the variable’s effect on the numeric integrator at various stages of the simulation.

In the following example, we can see that the variable \( J1.w \) dominates the error around \( t=0.8 \), but not at \( t=0.9-1.0 \) although the error is still > 60 %. The variable \( J1.w \) limits the step size near \( t=0.65 \) and \( t=1.15 \).

![Plot Error Example](image)

Plot Dependency displays the dependencies of the variables in a new window, as described in the previous section. The menu choice is disabled if dependencies have not been used.

In addition the behavior of the numeric integrator can be studied by pressing the Plot Integrator Step Size button, which opens a new plot window showing the internal integrator step size and the integration order. Note in particular that rejected steps (and subsequent restart with a smaller step) are indicated.
3.3.6 Plot window

Curve tooltip now optional

In Dymola 2018 you can select if curve tooltips should not be displayed or not. This can be done using the setting `Show curve tooltip`. This setting is available in the plot setup, accessed by the command `Plot > Setup…`, the `Options` tab.

The setting is default active. The setting corresponds to the new flag

```plaintext
Advanced.ShowPlotTooltip = false;
```
3.3.7 Scripting

The built-in function experiment included in Dymola Commands library

The built-in function experiment has been included in Dymola Commands library, in the Simulator API group. The function is used to define the default simulation setup.

experiment

experiment(StartTime=0.0, StopTime=1.0, NumberOfIntervals=0, OutputInterval=0.0, Algorithm="", Tolerance=0.0001, FixedStepSize=0.0)

Algorithm is a string with the name of the integration algorithm; the names correspond to the ones found in the popup-menu and the string is case insensitive. FixedStepSize is only used if the algorithm Euler is selected.

The entire command corresponds to the simulation setup reached by the command Simulation > Setup..., the General tab, except the Experiment group in that menu.

The function was available in earlier versions as well, but not included in the Dymola Commands library.

Note. The function has been made available also in the Java, Python, and JavaScript interfaces for Dymola.

The built-in functions simulateExtendedModel, simulateMultiExtendedModel, and simulateMultiResultsModel improved

The built-in function simulateExtendedModel, simulateMultiExtendedModel, and simulateMultiResultsModel have been improved:

- Default values have now been added for initialNames, initialValues, and finalNames, to be compatible with the built-in function simulateModel.
- The description of the functions is improved with a note that for the argument initialValues, Integer and Boolean variables (Boolean coded as 0 or 1) are also supported, although that argument is stated as Real.

The built-in function createPlot improved

Support for time unit

The built-in function createPlot now contains an argument to define time unit:

input String timeUnit = "s" "Time unit"

Support for display units

The built-in function createPlot now contains an argument to define display units:

input String displayUnits[size(y, 1)] "Display units. Empty string means use the default display units."
Selection of displayed curves

The built-in function `createPlot` now contains two new arguments to control what curves should be displayed:

```plaintext
input Boolean showOriginal = true "When enabled, original curves are shown"
input Boolean showDifference = false "When enabled, the difference between curves is shown"
```

These arguments correspond to the selections Original, Difference, and Original and Difference in the More section of the variable browser:

![Variable browser screenshot](image)

The built-in functions plotArray and plotArrays improved

The built-in functions `plotArray` and `plotArrays` have been improved to include unit handling; a new argument has been added:

```plaintext
Input String unit = "" "Unit"
```

This argument allows the user to specify units. An example is:

```plaintext
plotArray(1:10, sin(1:10), unit="s")
```

Since the resulting curve has a unit, the Display Unit command is available in the context menu of the curve.

The unit string can be enhanced to also specify the display unit by using the format “unit|displayUnit”. An example could be `unit="rad/s|rpm"`.

The built-in function updateModelicaAnnotations has been improved

The built-in function `updateModelicaAnnotations` has been improved to handle more annotations, in particular removing deprecated annotations for text-primitives. The function is now defined as:

```plaintext
function updateModelicaAnnotations "update Modelica annotations to follow the standard and preserve behavior"
input String className;
input Boolean changeLinePattern := true "LinePattern.None->LinePattern.Solid";
input Boolean orderBitmapExtent := true "Order Bitmap extent";
input Boolean removeTextDeprecated := true
```

---

Dymola 2018 Release Notes 51
"Remove deprecated textannotations";
input Boolean renameTextColor := false
   "Text.lineColor->Text.textColor";
output Boolean ok;
end updateModelicaAnnotations;

The inputs changeLinePattern and orderBitmapExtent was previously present in the function, but with fixed values (true). orderBitmapExtent is used to allow images to be created as flipped.

The inputs removeTextDeprecated and renameTextColor are new in Dymola 2018. The input removeTextDeprecated removes the following annotations for text-primitives: fillColor, pattern, fillPattern, and lineThickness. (The input renameTextColor is mainly intended for future use and presently has the default value false.)

As an example, annotations can be updated (with default settings) for a library MyLibrary by executing updateModelicaAnnotations("MyLibrary");. The library must be loaded before giving this command.

**Improved code completion in the script editor**

The below improvements applies to the script editor. For corresponding improvements in the Modelica text editor and the parameter input lines in the parameter dialog, see section “Improved code completion in Modelica Text editor and parameter dialog”.

**Improved selection of suggestions**

In Dymola 2018, the code completion is not anymore based on general text completion, it is now contextual, that means using code completion in a class now only suggests:

- Modelica keywords (highlighted in blue).
- Names of built-in functions available in DymolaCommands library.
- Names of open packages.
- Names of models available at the code completion level.

**Improved navigation**

As in previous versions, pressing Ctrl+Space displays a code completion list. In Dymola 2018, if you press Ctrl+Space again, or keeping Ctrl pressed and press Space again, the next alternative in the code completion list is selected.

### 3.3.8 Minor improvements

**Improved display unit handling**

Display unit for fuel consumption

Dymola defines three new display units for fuel consumption, specifically aimed at automotive applications. They are:
To use these display units the variable should be given the unit "$\text{cl/km}$" (centiliter/kilometer). The reason is that we do not want existing quantities with the SI unit $m^2$ to acquire surprising display units.

**Defining reciprocal display units**

Reciprocal unit conversions can now be handled through display units, such as the relationship between $l/100km$ and $mpg$ (miles per gallon). Using this new feature you can handle other reciprocal conversions like, for example, between frequency and period. The feature is in particular useful for library developers.

The following code in the file displayunit.mos shows the usage:

```mos
// Fuel consumption
defineUnitConversion("cl/km", "l/100km", 1, 0)
defineUnitConversion("cl/km", "mpg", 0, 454.609/1.609344)
```

Setting the input argument `scale` to 0 in the `defineUnitConversion` function call, defines the following argument `offset` to be used not as an offset, but instead as scale for an reciprocal conversion.

**Improved support of display units in some built-in functions**

The built-in functions `createPlot`, `plotArray`, and `plotArrays` now have support for display units. See the sections “The built-in function `createPlot` improved” and “The built-in functions `plotArray` and `plotArrays` improved” starting on page 50.

**Evaluation of top-level parameters supported**

In previous versions, an option *Evaluate parameters to reduce models (improves simulation speed)* was available. This setting is still present, and affects all parameters except top-level parameters; the idea is that those could be manipulated from e.g. Simulink.

In Dymola 2018 also the top-level parameters can be evaluated by using the setting *Also evaluate top-level parameters*. By default the option is not activated. The corresponding flag is `Advanced.EvaluateAlsoTop = false;`.

Both settings are available in the simulation setup. The settings can be reached by, for example, the command *Simulation > Setup…*, the **Translation** tab:
Improved logging of evaluated parameters

Already in previous versions the flag Advanced.LogStructuredEvaluation could be set to true to log all parameters that are evaluated during translation, and present the result in the simulation log, displayed in the Translation tab of the Dymola Messages window. (The flag is by default false.)

In Dymola 2018, the logging controlled by this flag has been improved to include links to the parameters, and displaying “annotation(Evaluate=true)” to indicate parameters that the modeler recommended to evaluate.
Support for checking of top-level of a model containing components of partial classes at top-level

The top-level of a model containing components of partial classes at top-level can be checked by setting the flag Advanced.StructuralCheckOfTopLevel = true. (The flag is by default false.) The check assumes that sub-models are correct and can be used to:

- Check a template-model (using partial models for replaceable components at top-level).
- Quickly find some common errors at the top level for any other model. In this case the check does not add any new information, but changes the order of the check; the top-level is checked first instead of last.

Note the limitation of this flag that “input connectors” of media are not handled correctly when checking Modelica.Media.Examples and Modelica.Fluid (primarily Sources).

This new flag was actually available already in Dymola 2017 FD01.

GUI support for order 1 for explicit and implicit Runge Kutta solver for inline integration

You can now select order 1-4 for explicit and implicit Runge Kutta in the Realtime tab of the simulation setup. Previously you could only select 2-4 here. Selecting order 1 means selecting explicit/implicit Euler. (This was also previously supported in code, but not in GUI.)

The execution setup is reached by the command Simulation > Setup....
Export result commands restructured in the variable browser top-level nodes context menu

In previous versions, either all variables, or the variables plotted in the active plot diagram could be saved, in a number of file formats. This was all done by the top-level nodes context command Export Result As… in the variable browser.

Introducing the new option to save variables present in the currently active plot window (variables in all diagrams of that window included), there was a reason to restructure the commands. The commands are now:

- **Export Result > All…** exports the whole result file. It corresponds to the previous command Export Result As… when saving the following file types (when selecting Save as type):
  - Result Files (*.mat)
  - Scientific Data Format (*.sdf)
  - Text Files (*.txt) **Note 1!** This item has been renamed to Dymosim input file.

**Note 2!** A new format has been added to the selections: Comma Separated Values (*.csv or *.txt)

So the selections in Dymola 2018 for this command are:
• **Export > Only Plot Diagram**... is the new command to save all variables in all plot diagrams of the currently active plot window. See “Exporting only variables in currently active plot window” on page 40.

• **Export > Only Current Diagram**... corresponds to the previous command **Export Result As**... when saving the following file types (when selecting **Save as type**):
  - Comma Separated Values – Only Plotted (*.csv or *.txt)
  - Result Files – Only Plotted (*.mat)
  - Scientific Data Format – Only Plotted (*.sdf)

**Changed handling of Scientific Data Format (SDF) files**

Due to Microsoft already having reserved the .sdf file extension for SQL Server Compact Edition Database File, the automatic association for .sdf files with the SDF Editor is not done anymore.

You have to do this association manually if you want, for example, that the file should be opened in the SDF Editor when you double-click the file in the browser. To do the association, right-click the file in a file browser, and select **Properties**. In the General tab, click **Change** and select **SDF_Editor.exe** if available. If not available, browse to \Program Files (x86)\bin\SDF_Editor.exe and select that one.

Note that if you have used previous versions of Dymola, .sdf files are still automatically associated with the SDF Editor. If you get problems because you are to use SQL Server Compact Edition Database files, you must change the file association manually (like above) to associate them with the SQL software.

**A simulation result file with default name is not deleted if a new simulation is stored to another result file name**

In previous version, the result file with default name (dsres.mat) was always deleted when a new simulation was performed, even if that simulation was stored to another result file name. This is not the case anymore; the old file with default file name is kept in such a case. (Files with other names than the default name was kept also in previous Dymola versions.)
Context menus in diagram layer in Simulation mode improved

In Dymola 2018 the context menus in the diagram layer displayed in Simulation mode has been improved:

- Context menu items that could never be accessed are disabled (removed) instead of being dimmed.
- Context menus of connections and graphical objects are disabled for the same reason.

A typical context menu for a component now looks like:
Minor restructuring of More section of the variable browser

The previous checkboxes *Original* and *Difference* have been restructured into a dropdown list.

Clicking the marked button in the above image gives:

The *Compare Results* checkbox has been turned into a corresponding toggle button (to the right of the red frame above).
3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 91.

3.4.1 Dymola as a 32-bit application will be discontinued

Dymola 2018 will be the last version to support Dymola as a 32 bit application. Dymola 2018 FD01 and later versions will only be available as 64-bit applications, on Windows and Linux.

Note however that only the Dymola 32-bit application will be discontinued, the executable dymosim.exe, for example, will still be available as a 32-bit executable.

3.4.2 Installation on Windows

License server: Upgraded version of FLEXnet

The Dymola license server for Windows is now upgraded to use FLEXnet Publisher version 11.14. This version is part of the Dymola distribution.

3.4.3 Installation on Linux

License server: Upgraded version of FLEXnet

The Dymola license server for Linux is now upgraded to use FLEXnet Publisher version 11.14. This version is part of the Dymola distribution.

3.5 Model Management

Reminder on enabling of Model Management

In Dymola 2017 FD01, the Model Management product (MOC) was integrated into the Dymola standard configuration, meaning that Model Management is from that version available without any licensing.

Because of this, the checkbox to control if Model Management should be disabled was removed, in the menu Help > License… > Details. The default value of this checkbox was that Model Management was disabled. That setting was stored in the setup file.

As a general rule, we never override customer settings, which meant that the stored setting may then block opening the Model Management library in Dymola 2017 FD01 and later versions.
To correct this, check what value the flag `Advanced.EnableModelManagement` has by executing it in the command input line of the command window. If the value is false, execute `Advanced.EnableModelManagement=true;` to re-enable the feature. This has to be done only once. The setting is saved between sessions.

3.6 Other Simulation Environments

3.6.1 Dymola – Matlab interface

Upgraded DymolaBlock GUI

A new text field has been added to the DymolaBlock GUI to allow the user to specify the name of the generated result file. If the checkbox Generate result is marked, the generated result file will get the name specified in the Result name text field (see picture below).

As default the text field is empty and if not specified the result name will be populated with the same name as the generated S-function MEX file after compilation. This is an automatically generated name (on the form `<modelname>_dy<tag>`) that ensures uniqueness between DymolaBlocks (and result files) also when generated from the same Modelica model.

The handling of the simulation input file has also been approved to better handle multiple DymolaBlocks in the same model. During compilation, the dsin.txt file is copied to a .txt-file with the same name as the generated Simulink MEX file (`<modelname>_dy<tag>.txt`). This will allow resetting of parameters in multiple DymolaBlocks where before there was a risk of using an incorrect dsin file.
The loading of pre-compiled binary S-function in import mode has also been improved to now use the new model-specific input file if it exists. The previous requirement to have both dsin.txt and dsin.mat together with the MEX file is now removed. It is sufficient for binary loading to have the following two files `<modelname>_dy<tag>.<mexext>` and `<modelname>_dy<tag>.txt`.

Finally, the DymolaBlock GUI has also been approved to allow proper rescaling of its graphical components when the window is resized.

**Compatibility**

The Dymola – Simulink interface now supports Matlab releases from R2012a (ver. 7.14) up to R2016b (ver. 9.1). 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

**GUI change due to integration of the Realtime Simulation product**

In Dymola 2017 FD01, the Realtime Simulation product (RHS) was integrated into the Dymola standard configuration, meaning that Real-time simulation is from that version available without any licensing.

Because of this, the checkbox in the menu Help > License… > Details that was used to control if Realtime Simulation should be disabled is now removed. The default value of this checkbox was to disable Realtime Simulation. This setting is stored in the setup file.

As a general rule, we never override customer settings, which means that a previously stored setting may block Realtime Simulation in Dymola 2017 FD01 and later versions.

To correct this, check what value the flag `Advanced.EnableRealtimeSim` has by executing it in the command input line of the command window. If the value is false, execute `Advanced.EnableRealtimeSim=true;` to re-enable the feature. This has to be done only once. The setting is saved between sessions.

(The reason the checkbox in the GUI was not removed in Dymola 2017 FD01 was done as precaution, because of the fact that Realtime Simulation is related to the Source Code Generation product.)

**Compatibility – dSPACE**

Dymola 2018 officially supports the DS1005, DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2018 generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases:

- dSPACE Release 7.3 with Matlab R2012a
- dSPACE Release 7.4 with Matlab R2012b
- dSPACE Release 2013-A with Matlab R2013a
- dSPACE Release 2013-B with Matlab R2013b
- dSPACE Release 2014-A with Matlab R2014a
• dSPACE Release 2014-B with Matlab R2014b
• dSPACE Release 2015-A with Matlab R2015a
• dSPACE Release 2015-B with Matlab R2015b
• dSPACE Release 2016-A with Matlab R2015b and R2016a
• dSPACE Release 2016-B with Matlab R2015b, R2016a, and R2016b

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 – Simulink Real-Time (formerly xPC Target)**

Compatibility with Simulink Real-Time has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R2012a (xPC Target ver. 5.2) to R2016b (Simulink Real-Time ver. 6.5). Only Microsoft Visual C compilers have been tested.

### 3.6.3 OPC Communication

**Logging of events to GUI and file supported (M)**

In Dymola 2018 events are by default logged in the GUI provided by the OPC server and optionally to a file. To alter the behavior, two environment variables can be set:

**DYMOSIM_OPC_LOG_LEVEL**

<table>
<thead>
<tr>
<th>Value</th>
<th>What is logged</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Warnings and errors</td>
</tr>
<tr>
<td>2</td>
<td>Errors</td>
</tr>
<tr>
<td>3</td>
<td>Nothing</td>
</tr>
</tbody>
</table>

**DYMOSIM_OPC_LOG_TO_FILE**

<table>
<thead>
<tr>
<th>Value</th>
<th>File written</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>opclog.txt</td>
</tr>
</tbody>
</table>

The file opclog.txt is created in the current simulation directory.
3.6.4 Java Interface for Dymola

**Improved string handling**

The Java interface for Dymola has been updated to handle string parameters natively. Setting the string parameter \( s \) to “foo” in the model `MyTestModel` when simulating it with the function call `dymola.simulateModel` can simply be written as:

```java
    dymola.simulateModel("MyTestModel(s="foo")")
```

**New built-in function available**

The built-in function `experiment` has been included in the Java interface for Dymola. The function is used to define the default simulation setup, for more information see section “experiment” on page 50.

**The built-in function createPlot improved**

The built-in function `createPlot` has been enhanced with arguments that support time unit, display units, and selection of displayed curves. See section “The built-in function createPlot improved” on page 50 for more information.

**The built-in functions plotArray and plotArrays improved**

The built-in functions `plotArray` and `plotArrays` have been improved to include unit handling via a new argument. Note that display units can also be specified by an enhancement of that argument. See section “The built-in functions plotArray and plotArrays improved” on page 51 for more information.

**Added default values for a number of built-in functions**

The built-in functions `simulateExtendedModel`, `simulateMultiExtendedModel`, and `simulateMultiResultsModel` now have default values for all input parameters.

3.6.5 Python Interface for Dymola

**Improved string handling**

The Python interface for Dymola has been updated to handle string parameters natively. Setting the string parameter \( s \) to “foo” in the model `MyTestModel` when simulating it with the function call `dymola.simulateModel` can simply be written as:

```python
    dymola.simulateModel("MyTestModel(s="foo")")
```

**New built-in function available**

The built-in function `experiment` has been included in the Python interface for Dymola. The function is used to define the default simulation setup, for more information see section “experiment” on page 50.
The built-in function `createPlot` improved

The built-in function `createPlot` has been enhanced with arguments that support time unit, display units, and selection of displayed curves. See section “The built-in function `createPlot` improved” on page 50 for more information.

The built-in functions `plotArray` and `plotArrays` improved

The built-in functions `plotArray` and `plotArrays` have been improved to include unit handling via a new argument. Note that display units can also be specified by an enhancement of that argument. See section “The built-in functions `plotArray` and `plotArrays` improved” on page 51 for more information.

Added default values for a number of built-in functions

The built-in functions `simulateExtendedModel`, `simulateMultiExtendedModel`, and `simulateMultiResultsModel` now have default values for all input parameters.

3.6.6 JavaScript Interface for Dymola

Improved string handling

The JavaScript interface for Dymola has been updated to handle string parameters natively. Setting the string parameter `s` to “foo” in the model `MyTestModel` when simulating it with the function call `dymola.simulateModel` can simply be written as:

```javascript
  dymola.simulateModel("MyTestModel(s="foo\")")
```

New built-in function available

The built-in function `experiment` has been included in the JavaScript interface for Dymola. The function is used to define the default simulation setup, for more information see section “experiment” on page 50.

The built-in function `createPlot` improved

The built-in function `createPlot` has been enhanced with arguments that support time unit, display units, and selection of displayed curves. See section “The built-in function `createPlot` improved” on page 50 for more information.

The built-in functions `plotArray` and `plotArrays` improved

The built-in functions `plotArray` and `plotArrays` have been improved to include unit handling via a new argument. Note that display units can also be specified by an enhancement of that argument. See section “The built-in functions `plotArray` and `plotArrays` improved” on page 51 for more information.
Added default values for a number of built-in functions

The built-in functions `simulateExtendedModel`, `simulateMultiExtendedModel`, and `simulateMultiResultsModel` now have default values for all input parameters.

3.6.7 FMI Support in Dymola

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

**FMU Export**

Extended support for string parameters in FMU export

String parameters in Dymola are now also fully supported in co-simulation export for any FMU.

This means that the use of string parameters is supported for all types of FMU export.

To have string parameters available you can use the new setting in the dialog reached by the command *Simulation > Setup…*, the FMI tab:
The setting is also available each time you export an FMU (if the setting *Show dialog when exporting* is set):
The setting corresponds to the flag `Advanced.AllowStringParameters=true`. This flag was available already in previous version.

**Note!**

- The default value of the setting (and the flag) is unticked (`false`), since you would normally not want the regular simulation to use string parameters as this makes the code slightly less efficient.
- String variables are not supported in FMU export.

**Extended support for optional FMI Export options in FMI 2.0**

- `fmi2GetDirectionalDerivatives` is now also supported for when selecting Co-simulation using Dymola solvers, except for the solvers Lsodar, Dassl, Euler, Rkfix2, Rkfix3, and Rkfix4.
- `fmi2GetFMUstate`, `fmi2SetFMUstate`, and `fmi2FreeFMUstate` are now also supported when selecting Co-simulation using Dymola solvers, except for the solvers Lsodar, Dassl, Cvode, Euler, Rkfix2, Rkfix3, and Rkfix4. Note that you can select Co-simulation using Cvode to get Cvode support including these options.
**FMI Kit for Simulink**

A new minor version (2.4.0) of Dassault Systèmes tools for export and import of FMUs in Simulink is released and is included with Dymola 2018. The tool includes support for Matlab R2016b.

---

**3.7 Advanced Modelica Support**

**3.7.1 Support for Modelica Language version 3.4**

Dymola 2018 is compliant with the new Modelica Language Specification version 3.4. Support for the following new features is added:

- Explicitly casting a model record, provided you set the flag `Advanced.RecordModelConstructor=true;` (The default value of the flag is `false`.)
- Differentiation of functions handles records mixing Real and no-Reals.
- Ellipse segments can set whether to also draw arc, chord, or nothing.

Note that the other major features added in Modelica 3.4 were already supported in previous releases of Dymola – e.g. automatic conversion of models.

**3.7.2 Deprecation warnings**

In Dymola 2018 the following language constructs generate warnings indicating future deprecation:

- The usage of `extends` to constrain replaceable components – `constrainedby` should be used instead.
- The usage of `operator` as an identifier name; since Modelica 3, `operator` is a keyword.
- The usage of the `flow` and `stream` prefix within short class definitions.

Note that Modelica 2 contains the above; using Modelica 2 will generate warnings when used, due to this.

The warnings are displayed in the messages window, in the **Syntax Error** tab.
3.8 Modelica Standard Library and Modelica Language Specification

The current version of the Modelica Standard Library is version 3.2.2. The current version of the Modelica Language Specification is 3.4.

Note that the Modelica Standard Library is compliant with the Modelica Language Specification 3.4. (So this version of the Modelica Language Specification should be used for reference, and not the Modelica Language Specification 3.2, Revision 2.)
3.9 New libraries

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

3.9.1 Thermal Management Demos Library

The Thermal Management Demos library contains examples that demonstrate how to combine the commercial Cooling Library with other commercial libraries to design thermal management solutions.

The examples include a battery pack under cyclic load cooled by a cold plate, and an integrated coolant in an electric machine.

The demo library can be opened by the command File > Demos > Thermal Management Demos Library.

The number of examples available depends on what libraries you have license for. To be able to use the battery pack example, you need to have license for the Cooling library and the Battery library, to be able to use the integrated coolant you need to have license for the Cooling library and the Electric Powertrain library.

3.9.2 Thermal Systems Library

The Thermal Systems library is intended for the stationary and transient simulation of freely configurable thermodynamic systems. Thanks to the included substance property library, system simulation with various media can be performed.

The library consists of extremely versatile models. On the one hand, individual components can be modeled and calculated in detail. On the other hand, the models are ideally suited for the design and optimization of large and complex systems. In this respect, this library is characterized not only by its precision, but also by extreme speed.

The library bases on a shallow structure, which allows expanding the library easily to the users requests by resorting to pre-defined models, structures and numeric approaches.

This library may be used, among others, for the modeling of the following systems:

- Refrigeration cycles, including refrigeration mixtures
- Heat pump systems e.g. with ejectors
- Hydraulic networks
- Clausius-Rankine cycles
- Heating, ventilation and air-conditioning systems
- Ab- and adsorption systems
- Fuel cell systems

The graphical user interface that this library provides, in addition to the source code view, enables the composition and parameterization of very different thermal systems. In Modelica code, model details can be viewed and components can be added. The library follows the object-oriented structure of the program and the principle of inheritance.
3.9.3 Thermal Systems Mobile AC Library

The Thermal Systems Mobile AC library is based on the Thermal Systems library and focuses on mobile air conditioning systems. It provides additional models for: A car cabin, a detailed MPET heat exchanger, an internal heat exchanger, and common example systems for mobile AC cycles with different refrigerants.

The Cabin model enables the energy balancing of a car’s single air volume and its surrounding components like walls, windows and built-in components. Furthermore, inner and outer thermal effects, the air conveyance separated in fresh and circulated air and the moisture ratio, influenced by the passenger’s water production are considered. Besides that, the human energy balance and the comfort level (PMV/PPD) will be calculated.

The detailed MPET model enables the discretization of each pass along the refrigerant flow direction and orthogonal to the flow direction. The model can have multiple layers, and the flow pattern through the passes is configured by the user. The detailed MPET model provides ports on the air side which can be used for inhomogeneous air temperature and mass flow rates. The optional receiver model makes this heat exchanger applicable to evaporator and condenser geometries. The internal heat exchanger is a TubeAndTube model which provides a simple interface to enter the IHX geometry.

3.9.4 Vehicle Systems Modeling and Analysis (VeSyMA) Library

VeSyMA (Vehicle Systems Modeling and Analysis) is a suite of commercial Modelica libraries for the modeling of vehicles.

The core functionality is provided in the VeSyMA library and several extensions to this foundation provide more detailed and sophisticated models for those domains. This initial release focuses on Automotive applications.

VeSyMA consists of a suite of Modelica libraries that are built on the Modelica Standard Library and VehicleInterfaces Library from the Modelica Association. The VehicleInterfaces Library is an open standard that defines the basic architecture of vehicle subsystems without limiting how each subsystem can be defined and how they can be combined together.

The VeSyMA suite of libraries covers the simulation of vehicles as well as the integration and deployment of the models for use in other environments. The core VeSyMA library provides the foundation for all the other Modelica libraries within the solution and is designed to enable the prediction of performance, fuel economy and energy analysis of vehicles. In this initial release, it provides the architecture that is used across the other automotive orientated application libraries, including VeSyMA - Engines, VeSyMA - Powertrain and VeSyMA - Suspensions. The use of the templates and standardized architecture makes it easy to vary the level of detail in each subsystem to suit your current needs.

This library replaces the Automotive Library and your models can be automatically migrated using a conversion script that has to be manually run before opening your own models. See the library documentation for more information.
The whole suite of libraries is built using an open standard model architecture and standard Modelica connectors. This means that the capabilities can be easily extended through the integration of 3rd party Modelica libraries. This is especially easy if the libraries are compatible with the VehicleInterfaces Library as examples are provided showing how this can be done (see the VeSyMA2EPTL Library\(^2\)).

The example shown below is a full vehicle using the architecture defined in this library, with suspension and wheel models from VeSyMA - Suspensions, multibody gearbox and driveline from the VeSyMA - Powertrain and engine model from the VeSyMA - Engines.

The core Vehicle Systems Modeling and Analysis (VeSyMA) Library provides complete vehicle systems models to enable performance, drive cycle and gradeability analysis looking at every aspect of the vehicle operation. It builds on the VehicleInterfaces Library which provides the basic model architecture to enable the analysis of conventional, hybrid, electric, and novel automotive vehicles.

Key features include:
- Drive cycle simulation
- Performance prediction
- Fuel consumption prediction

\(^2\) The VeSyMA2EPTL library provides examples of how to couple the VeSyMA and the Electrified Powertrains libraries. The models in the library will only appear correctly if you have licenses for both libraries.
- Energy flow analysis
- Powertrain integration analysis
- Extendible model fidelity through integration with rest of the VeSyMA suite and 3rd part libraries.

The core VeSyMA library provides the foundation for other libraries that extend from, and develop on, the templates that are set out here, with interfaces of components and vehicle models extended from the same root allowing any available and applicable components to be interchanged to suit the area of investigation. While each further library has its own individual area of development there are some basic models contained here that allow a basic level of full vehicle simulation such as a linear 3 degree of freedom drive cycle test.

3.9.5 Vehicle Systems Modeling and Analysis (VeSyMA) - Powertrain Library

The Vehicle Systems Modeling and Analysis (VeSyMA) - Powertrain Library is part of the Vehicle Systems Modeling and Analysis solution which is a set of commercial Modelica libraries for vehicle simulation. This part of the platform focuses on the transmissions, gearboxes and driveline models.

The VeSyMA Library provides the foundation for the platform which means it includes the interfaces, templates and 1D components that allow for representative evaluation of vehicles with reduced complexity models. The Claytex library provides common components that can be used in many different applications. The use of this interfacing allows users to create custom engine models, either using the templates or from the base interface to allow easy application within the vehicle models.

The VeSyMA - Powertrain Library provides a convenient modeling methodology and deliver efficient simulation of these complex systems. The Modelica name of the library is PTDynamics. It includes models for shafts, bearings, gear meshes, flexible joints, as well as complex assemblies such as epicyclic and differential models including the associated mounting systems. It has been developed to support the modeling of any drive system, including automotive and marine gearboxes and drive systems. Animation is included in all the parts to aid understanding of the system dynamics; the image below is of a complete vehicle driveline created using this library (suspension from VeSyMA - Suspensions).

This library was previously known as the Powertrain Dynamics Library.

3.9.6 Vehicle Systems Modeling and Analysis (VeSyMA) - Suspensions Library

The Vehicle Systems Modeling and Analysis (VeSyMA) - Suspensions library is part of the Vehicle Systems Modeling and Analysis solution which is a set of commercial Modelica libraries for vehicle simulation. This part of the platform focuses on the suspension systems and provides the ability to evaluate the dynamic performance of automotive vehicles in static, rig and full vehicle manoeuvre experiments as shown below with the example of the Double Lane Change experiment.
The VeSyMA Library provides the foundation for the platform which means it includes the interfaces, templates and 1D components that allow for representative evaluation of vehicles with reduced complexity models. The Claytex library provides common components that can be used in many different applications. The use of this interfacing allows users to create custom suspension systems, either using the linkage templates or from the base interface to allow easy application within the vehicle models. With many different common and less common ideal, compliant or bushed example linkages available with associated experiments.

The VeSyMA - Suspensions Library provides parametric and MultiBody suspension models for a wide variety of suspension linkages. The MultiBody suspension models can use ideal joints or include bushes and flexible bodies. The driver and road models provided enable a wide range of dynamic driving events to be simulated as well as many different test rigs. Real-time capable models are included and the model architecture includes support for the multi-threading features available in Dymola to partition the model into multiple threads.

As all the components are based on VeSyMA it allows easy interfacing with other libraries that provide detailed models of both Engines and Powertrain.
3.10 Updated libraries

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

3.10.1 Air Conditioning Library

A new version 1.14 has been released.

Examples of improvements and new features

- A new flag was introduced in the air-refrigerant heat exchangers, which allows to easily switch between inlet/outlet headers only and a uniform distribution of the volume across the entire refrigerant flow path including the internal header volumes, which do not store mass. It is recommended to use inlet/outlet headers only, if the dynamic behavior is the main interest and the switch distributeToInternalHeaders, if the correct charge is of more importance.

- The receiver summary Boolean variable full has been removed, since it could cause unnecessary events and chattering.

- The friction model in pipe components in the experiment models under Templates are no longer parameterized using initial parameters, because component friction losses should not depend on init value choices.

Conversion of user libraries

Automatic conversion of user libraries from previous version is supported using the included conversion script. Note! In order to correct a previous bug when converting, some manual actions might need to be performed. See the library documentation for more information.

3.10.2 Battery Library

A new version 1.4.1 has been released.

Examples of improvements and new features

- New parameter fitting to generate a table based cell model from an electrochemical cell model using python. The generated cell model includes data for the open circuit voltage, the serial resistor and three RC-circuits. The python file for the fitting is located at Battery/Resources/Scripts and the required Dymola models are included in the ParameterFitting package.

- Voltage limits added and assertion activated for
  
  - Battery.Electrochemical.Cells.Variants.LCO_3Ah
  - Battery.Electrochemical.Cells.Variants.LMO_2Ah
3.10.3  **Brushless DC Drives Library**

A new version 1.0.1 has been released. This is a maintenance release.

**Examples of improvements and new features**

- Weakened encryption, now all models can be duplicated
- Added identifier in data selection pulldown menu to show where a data set is coming from/is located
- Averaged machine model has new parameters (useL, useLt) to switch off inductances to further improve the performance

3.10.4  **Cooling Library**

A new version 1.1 has been released.

**Examples of improvements and new features**

- Heat exchanger models (static) for liquid-liquid and liquid-air application
  - Geometry based flat tube louvered fin (liquid to air) and chevron plate (liquid to liquid) heat exchanger models
  - Generic heat exchanger models for cross- counter- and parallel flow configuration
- Optional dynamic thermal wall model in pipe components
- Bend components for liquid and air domain
- Control valve with compressibility effect for air domain
- Weakened encryption enables more efficient development (you can now duplicate most models)
- Re-package flow split and join components in dedicated SplitsJoins package
- Simplified parameter dialog in liquid domain pipe models
- Revised package structure for shared friction and heat transfer models
- Pipe models now also with rectangular cross section
- New pressure loss models using Modelica.Fluid.Dissipation
- Revised performance characteristic model for liquid pump models
- Revised model structure of the three-way-valve model

3.10.5  **Design Library**

A new version 1.0.6 has been released. The previous use of the now obsolete Modelica_LinearSystems library is now updated to use the present Modelica_LinearSystems2 library.

For the function `sweepTwoParameters` in `Design.Experimentation.Analysis` it is now possible to output the result to a CSV file.
3.10.6 Dymola Commands Library

A new version 1.3 has been released. The update reflects changes of built-in functions described in, for example, section “Scripting” starting on page 50.

3.10.7 Electric Power Library

A new version 2.3 has been released.

Electric Power library 2.3 has seen a large update focusing on improving the user experience. This new version comes with new dynamic visualizers of key signals for DC and AC for all transforms which makes it easier to analyze and understand large systems. The library has seen a large update on model documentation and user parameter dialogs, which make it easier to get started with the modeling. Version 2.3 comes with the Modelon Base Library containing a large set of thermofluid models that may be used for more detailed investigation of temperature and cooling effects. Beyond this the library comes with an improved Modelica compliance.

Examples of improvements and new features

- Improved Modelica compliance - The full library now checks pedantically.
- Improved documentation - Expanded documentation for the AC3ph_dqo package and corrected/clarified many other cases.
- Restructured examples package - Low level component examples are now located within the individual component packages rather than all in the top level Examples package. This makes it easier to find the individual test cases for component models.
- Removed deprecated packages
- Added dependency to Modelon Base Library (MBL) - This avoided duplicating various icons that already existing in the MBL.
- Renamed 'Partials' packages to 'Interfaces' and 'Templates' across the library - This improves consistency across other Modelon libraries.
- Added Literature package - This new package includes references to literature related to electric power systems.
- New display components created - New display components were created for each of the representations (DC, AC1ph, AC3ph_abc, AC3ph_dqo, and AC3ph_dq). The new display sensor extends from each of the respective PVImeter components in the Sensors packages and can be used in place of these meters. They include discrete cells for simultaneous display of 5 variables that that are independently selectable by the user. Examples using the new meters are included in the top level Examples package (Islanding (abc),Islanding (dq0),Islanding (dq)) and also in each component example package (DC, AC 1ph, AC 3ph abc, AC 3ph dq0, and AC 3ph dq).
- Improved dialog box for the System component - This reduces the number of button clicks needed by the user to choose specific options.
- Improved frequency specification in the System component - Previously the user was restricted to specifying the frequency in radians unless they had custom unit conversions defined between hertz (Hz) and radians (rad). Also included typical train system frequencies in the drop-down list of frequencies.
Conversion of user libraries

Automatic conversion of user libraries from version 2.2.4 is supported using the included conversion script.

3.10.8 Electrified Powertrains Library

A new version 1.1 has been released.

Examples of improvements and new features

- Integration of the electrically excited synchronous machine (ESM)
  - Based on d/q field oriented control including decoupling and field weakening (by torque reduction)
  - Automatic controller parameter computation based on machine parameters for d/q and excitation current controllers
  - Torque setting by optimal current computation (based on minimal copper loss)
  - Speed control based on magnitude optimum
  - Two ready-made datasets of machines
  - Feature-parity with PSM/AIM including map generation, drive integration, generic and split-losses energy-based models

- User Convenience
  - Weakened encryption enables more efficient development (basically you can now duplicate all models)
  - Added warning/error message for exceeded temperature in Lossy Inverter

- Other functionality items
  - Simplified Energy Analysis models in ElectricMachines
  - Simplified and unified power electronics losses in Conduction and Switching losses
  - Removed "firstOrder" block in within IGBT and Diode losses due to changes in the above models
  - Changed delay time in IGBT and Diode losses to turnOn or turnOff time respectively
  - Changed initialization in BatterySupplied Example to better reflect physical behavior
  - Reworked interfaces for EnergyAnalysis machines to enable the electrically excited machine

3.10.9 Engine Dynamics Library

A new version 1.2.8 has been released. It is a maintenance release.

No conversion of user libraries is needed.
3.10.10 Engines Library

The library has been renamed. Please see section “Vehicle Systems Modeling and Analysis (VeSyMA) - Engines Library” starting on page 86.

3.10.11 Fuel Cell Library

A new version 1.5 has been released.

Version 1.5 of the Fuel Cell library has focused on improving the user-friendliness. The new release has extended the capabilities of the composition visualizers to enable the option to show flow per species including both mass and mole based units. With this new functionality, it is easy to get an overview of the flow streams in the system. This is especially useful when visualizing diffusion flow through a membrane. There has also been a big improvement on the examples where a large part of the examples visualize system performance indicators such as fuel utilization, heat release and temperature and concentration profiles. This makes it easy to quickly get an overview of the system characteristics.

Examples of improvements and new features

- Flow substance visualizers to show the flow composition in terms of mass and mole fractions. A user can conveniently switch between flow or fraction sensor types where both mass and mole based units are supported.
- Predefined preferential oxidation reactor with a new output variable for the CO concentration in ppm. This makes it easy to directly compare against CO limits that are specified in ppm.
- Example models that visualize important system indicators, to quickly get an overview of the system characteristics
- Top-level propagation of composition design parameters in DesignProx model.

Conversion of user libraries

No conversion is needed.

3.10.12 Heat Exchanger Library

A new version 1.6 has been released.

Examples of improvements and new features

- In test benches for cross-flow flattube heat exchangers different sets of boundary conditions can be selected via a new initType parameter. Different choices with groups of three from in- and outlet pressures, inlet specific enthalpy and temperature, mass flow rate and outlet superheat/subcooling are available. Steady-state initialization must be selected for all choices other than the combination of outlet pressure, inlet specific enthalpy and inlet mass flow rate.
- Configuration of heat exchanger stacks in the air flow has been made more convenient and flexible with the new templates. Different fluids may be chosen freely as well as
additional components such as fans and obstacles. See also the updated tutorial for the supported workflow. The powertrain cooling examples illustrate the combined performance of heat exchangers from high and low temperature cooling circuits, refrigeration system and charge air path together with fans and flow obstacles. The old templates are moved to a Deprecated package.

- An opening component as an additional choice for an obstacle in the air stream can be integrated in stacks.
- The parameterization of example heat exchangers was moved from test benches to separate heat exchanger classes, in order to illustrate the recommended workflow. Pre-configured heat exchanger models are then used in experiment models via redeclare.
- An incorrect computation of air-side pressure drop in Kim-Bullard and Chang-Hsu correlations is now fixed. A deviation from results with previous versions can be expected for heat exchangers with more than one layer only. The pressure drop was previously underestimated. The previous bug-containing versions are kept in the Deprecated section to conserve the behavior of calibrated components.

Conversion of user libraries
Automatic conversion of user libraries from is supported using the included conversion script.

3.10.13 Human Comfort Library
A new version 2.3.0 has been released. This release is a maintenance release.

3.10.14 Heating, Ventilation, and Air Conditioning Library (HVAC Library)
A new version 2.3.0 has been released.

Note! When checking or translating models with unmatched sinks/sources, error messages appear. You have to manually ensure that the sinks/sources are matched. Assuming sources and sinks are matched, setting the flag

\[
\text{Advanced.StructuralCheckOfTopLevel=true;}
\]

generates correct diagnostics. For more about this setting, please see “Support for checking of top-level of a model containing components of partial classes at top-level” at page 55.

Examples of improvements and new features
- HVACLib.Components.SolarPanels.SolarPanel_PV: The ambient temperature is newly provided by an input which can be defined manually or connected with the weather adapter. The former ambient model is not longer used.
3.10.15  Hydraulics Library

A new version 4.6 has been released.

The new version contains new fitting components based on the 3-K (Darby) method added. This includes loss coefficients for 33 different bends (for different angles), valves and tees. In addition to this, a simpler 1-K tee is also available.

No conversion is needed for user libraries of version 4.5.

3.10.16  Hydro Power Library

A new version 4.6 has been released.

Examples of improvements

- Added enableDroop parameter in powergrid model to enable or disable droop control. This makes it possible to model production units which are not affected by a deviation in frequency.
- Improved documentation of recovery factors.

Conversion of user libraries

No conversion is needed from version 2.6 is needed.

3.10.17  Liquid Cooling Library

A new version 1.5.3 has been released.

The valve logic has been restructured and extended. It has also been moved to Modelon base library.

Automatic conversion of user libraries is supported using the included conversion script.

3.10.18  Pneumatics Library

A new version 2.2 has been released.

Examples of improvements and new features

- Example package involving air suspension systems is added. The air suspension components as well as system models involving developed components are made available to the user.
- New fitting components based on the 3-K (Darby) method. This includes loss coefficients for 33 different bends (for different angles), valves and tees. In addition to this, a simpler 1-K tee is also available.
- Reservoir will no longer have volume-volume connection inside
- MultiPositionCylinder example no longer have volume-volume connections
• The gases documentation is made to clarify the assumptions about inflowing enthalpy to volumes. The stop_h_outflow parameter in Constant gas model was also removed from GUI to avoid confusion, it is now always true.

• Enhanced parameterization option for Bellows model. Effective area and volume can be specified by table. Added bellows material damping as well.

Conversion of user libraries

Automatic conversion of user libraries from version 2.1 is supported using the included conversion script.

3.10.19 Thermal Power Library

A new version 1.14 has been released.

This release comes with a large set of models targeting simulation and optimization of district heating networks. With these additions, engineers in the field can:

• Learn the limits of performance of the power plant and the network
• Improve the economy by minimizing production and operation costs, as well as heat losses

Using the physics based modeling of Modelica, it is possible to optimize heating networks without simplifications and set constraints on any temperature, flow or pressure in the system. This is something that is not possible with existing standard approaches based on considerable model simplifications using linear models. To demonstrate this, examples aimed at district heating production planning using dynamic optimization are included in the library. These demonstrate how to find the most profitable scheduling of production units in a heating network with a clear objective and well defined constraints such as limits in heating and pumping capacity.

Examples of improvements and new features

• District heating package with simulation and optimization examples. Pipe, consumer, producer and optimization models including optimization tutorial with scripts.
• Time varying system ambient temperature. This is useful for simulations that reach several hours in duration where the ambient temperature may vary. An example of this is district heating simulation.
• A fast simulating, incompressible water medium. This can be useful in water applications where simulation speed is prioritized.
• Variable correction factor CF_length in pipe and heat exchanger models and a corresponding CF_area in wall models. They can be used to dynamically change the length and heat transfer area during simulation in order to solve design problems.
• Improved parameter dialog for fixed zeta flow resistance.
• Improved Solid media interface.
• Replaced the thermoRoot flow regularization function with regRoot from the Modelon Base Library.
Conversion of user libraries
Automatic conversion of user libraries from version 1.13 is supported using the included conversion script.

3.10.20   Vapor Cycle Library
A new version 1.5 has been released.

Examples of improvements and new features
- The superheat and subcooling sensors contain an option, which, if activated, returns negative values in the two-phase region. This is useful, if superheat or subcooling are controlled close to zero in a system.
- New expansion valve models were added to the library: An electronic expansion valve and a check valve.
- The orifice tube model now contains additional options in case the used refrigerant is R744: A bypass for inlet pressures larger than a user defined threshold and the possibility to calibrate the tube with available experimental data in case no geometry information is present.
- The Modelica source code of external media example packages can be accessed and copied in order to easily create own packages.
- A compressor test bench experiment template was added.

Conversion of user libraries
No conversion is needed for this version.

3.10.21   Vehicle Dynamics Library
A new version 2.5 has been released.

Examples of improvements and new features

Slalom maneuver
A new ISO Slalom maneuver has been introduced. There's a RoadBuilder function (Grounds.RoadBuilder.slalom) to create the road file and an example experiment (Examples.Manuevers.Slalom) that uses it.
New functionality in suspension test rigs

The suspension test rigs can now actuate suspension roll angle while maintaining a predefined total axle load. This is a common load case when evaluating a suspension.

When running with contact patch height as input to the test rig, there is now the option to specify a switch time when the rig switches from controlling wheel center height to controlling contact patch height. This allows the zero height for contact patch height to be defined where wheel center height is zero. This gives more consistent results for varying tire radius and stiffness.

Updated interfaces for heavy vehicles

Chassis interfaces for heavy vehicles have been updated. There is now a single interface with vectorized hubs (Vehicles.Chassis.Interfaces.Truck) that is used for all truck and tractor models. Similarly, new interfaces for trailers have been added (Vehicles.Chassis.Interfaces.SemiTrailer and Vehicles.Chassis.Interfaces.FullTrailer). SemiTrailer has one group of axles while FullTrailer has one front and one rear group. Note that existing truck and trailer templates have been updated to extend from these new interfaces.

Having a single interface for all truck models allows us to create more experiment templates which are valid for all trucks regardless of the axle configuration. Experiment templates for trucks are now available in Vehicles.Chassis.Experiments.Templates.Trucks.
The **Change Class** menu can now be used to redeclare to any truck example in VDL and any custom model that extends the single Truck interface model.

**Conversion of user libraries**

Automatic conversion of user libraries from version 2.4 is supported by using the included conversion script.

### 3.10.22 Vehicle Systems Modeling and Analysis (VeSyMA) - Engines Library

VeSyMA - Engines is the new name for the Engines Library that has been available for several years. With this release, version 2017.1, it has been tightly integrated into the complete VeSyMA (Vehicle Systems Modeling and Analysis) suite of libraries and features a revised model architecture.

**Examples of improvements and new features**

**Re-design of the architecture of the engine templates**

The engine has been subdivided into 5 main subsystems:

- Intake system (from atmosphere to intake ports)
- Exhaust system (from exhaust ports to atmosphere)
- Pressure charging devices (Turbocharger/supercharger + intercooler)
- Combustion models
- Mechanics

This revised architecture makes it possible to separate the fluids from the mechanics particularly for multi-core use of the models.
Mechanics and Fluids can now be tested independently of each other; all the subsystems in the engine are either entirely fluids or entirely mechanics based.

Re-design of the combustion model

The combustion model has a similar interface to the 2016.4 version but the contents have been rearranged. The combustion models now sit within a dedicated subsystem.
Re-design of the valvetrain

In Engines 2016.4 the fluid and mechanical valves resided within the cylinder head model. In this version of the library, the intake and exhaust fluid valves have been moved into the intake and exhaust airpaths:

The mechanical valves remain similar to 2016.4 and reside within a new cylinder head model which also includes the camshafts (which are also similar to 2016.4):
Thermal modifications

- The engine thermal network has been revised and integrated within the mechanical components. These include component masses and heat transfer models between the fluids (air, coolant and lubricant) and the solid components (head, liners, pistons, block).
- The internal temperature boundary for coolant and lubrication within the engine model has been removed.
- All lubrication and cooling system modeling is done externally to the engine and the only interface between the engine models and the cooling and lubrication circuits are Modelica Standard Library thermal connectors.

Mechanics modifications

- Crankshaft records are now non-replaceable so that the record is parameterized directly within an extended crankshaft model.
- The mechanical models of the valves now reside within the cylinder head module (e.g. Engines.CylinderHeads.Templates.SingleCylinderCarem) together with the camshafts.

Fluids modifications

- A range of improvements has been made to the Claytex.Fluid library. These changes can be viewed within Claytex.UsersGuide.ReleaseNotes.Version_2017_1.
- The swept volume has been updated to include a replaceable reaction model which includes fluids trace substance calculations for emissions modelling.
- Modelica Fluid connectors have been used throughout the fluid models and the hierarchical connectors (fluid + thermal) have been dropped.
- The piston blow-by model has been relocated within the combustion models so that it is split from the mechanics.
- The intake and exhaust valve fluid models now reside within the intake and exhaust air-paths respectively.
• A lubrication system has been added to the library and experiments

Busses modifications
The cylinder bus (now an expandable connector rather than a standard connector) has been extracted from the control bus to make arrays on the cylinder bus more robust in diagnostics and use

Emissions modifications
• Oxygen map within swept volume and emissions block revised to use AFR only as an input
• Oxygen sensor inputs air fuel ratio onto emissions bus

Controllers modifications
• Fuelling control is now improved with the option to either use the Oxygen sensor signal fitted to the exhaust or the AFR signal calculated within the combustion model
• Engine Dynamometer ECU constraining clause has been relaxed so that users can plug in ECUs which extend from the ECU interface rather than GenericECU

Graphical modifications
• Icons orientation for intake and exhaust systems are now inline with the layout of their respective diagram layers leading to a more intuitive navigation of these system models
• Combustion icons revised to portray the model functionality more accurately.

3.10.23 Wind Power Library
A new version 1.0.1 has been released.

Examples of improvements
• The Wind Power library is converted to a standalone product. All dependencies on the Electrified Powertrains library have been removed by integrating the required components.
• Weakened encryption enables more efficient development (basically you can now duplicate all models)
• Changed styling of documentation layer

3.11 Documentation
In the software distribution of Dymola 2018 Dymola User Manuals of version “March 2017” will be present; these manuals include all relevant features/improvements of Dymola 2018 presented in the Release notes.
Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2018 are listed.

3.12.1 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 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 4 GB of RAM for 32-bit architecture and 6 GB of RAM for 64-bit architecture.

3.12.2 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions
Dymola 2018 is supported, as 32- and 64-bit application, on Microsoft Windows 7, Windows 8.1, and Windows 10. Since Dymola does not use any features supported only by specific editions of Windows (“Home”, “Professional”, “Enterprise” etc.), all such editions are supported if the main version is supported.

Compilers
Please note that for the Windows platform, a Microsoft C/C++ compiler, an Intel compiler, or a GCC compiler, must be installed separately. The following compilers are supported for Dymola 2018 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)
• Visual Studio 2013 Express Edition for Windows Desktop (12.0)
• Visual Studio 2015 Express Edition for Windows Desktop (14.0)

**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)
• Visual Studio 2013 (12.0)
• Visual Studio 2015 (14.0)

**Intel compiler**
The Intel compiler Intel Parallel Studio XE 2016 is supported.

*Note* that you must also select a Visual Studio compiler when selecting the Intel compiler.

Current limitations:
• Embedded server (DDE or OPC) is not supported.
• Export DLL is not supported.

**GCC compilers**
Dymola 2018 has limited support for the MinGW GCC compiler. The following versions have been tested:
• For 32-bit GCC: version 4.8.1.
• For 64-bit GCC: version 4.9.2.

To download any of these free compilers, please visit [http://www.Dymola.com/compiler](http://www.Dymola.com/compiler) where the latest links to downloading the compilers are available. Needed add-ons during installation etc are also specified here. Note that you need administrator rights to install the compiler.

Current limitations with 32-bit and 64-bit GCC:
• Embedded servers (DDE or OPC servers) are not supported.
• Support for external library resources is implemented, but requires that the resources support GCC, which is not always the case.
• No support for runtime concept\(^3\).
• For 32-bit simulation, parallelization (multi-core) is currently not supported for any of the following algorithms: RadauIIa, Esdirk23a, Esdirk34a, Esdirk45a, and Sdirk34hw.

---

\(^3\) The Dymola runtime concept means that a Dymola model developed by a user that lack export options still can be executed on a different computer with a Dymola license (without an installed Dymola application) by pointing to that license by an environment variable.
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 11.14. This version is part of the Dymola distribution.)

Linux

Supported Linux versions and compilers

Dymola 2018 runs on SUSE Linux (Release 11), 32-bit and 64-bit, with gcc version 4.9.3, and compatible systems. 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 change compiler, change the variable CC in /opt/dymola-<version>-<architecture>/insert/dsbuild.sh, where <architecture> is i586 for 32-bit application, and x86_64 for 64-bit application. As an example, for a 64-bit Dymola 2018 application:

```
/opt/dymola-2018-x86_64/insert/dsbuild.sh
```

Dymola 2018 is supported as a 32-bit and 64-bit application on Linux.

Notes

- 32-bit compilation might require explicit installation of 32-bit libc. E.g. on Ubuntu:
  `sudo apt-get install g++-multilib libc6-dev-i386`
- Dymola is built with Qt 5.5 and thereby inherits the system requirements from Qt. However, several xcb helper libraries are bundled with Qt (in detail, QT was built with the flag –qt-xcb) in order to reduce the system dependencies as much as possible.
- To support older gcc versions, Dymola distributes a libstdc++ (e.g. for 64-bit: bin/lib64/libstdc++.so6) compatible with the currently supported gcc version. However, this may interfere with newer gcc versions. If your system has a newer gcc version than the one recommended for Dymola, the distributed libstdc++ should be removed.
- For rendering of jpg files, libjpg62 must be installed.

Note on libraries

- 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.
- The library UserInteraction is not supported on Linux.

Dymola license server

For a Dymola license server on Linux, all files needed to set up and run a Dymola license server on Linux, except the license file, are available in the Dymola distribution. (This also includes the license daemon, where Dymola presently supports FLEXnet Publisher 11.14.)