

Dymola

Dynamic Modeling Laboratory

Installation

Contents: Chapter 4 “Appendix – Installation”
extracted from the manual “Dymola User
Manual 1A: Introduction, Getting Started, and
Installation”.

March 2025 (Dymola 2025x Refresh 1)

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

© Copyright 1992-2025 by Dassault Systèmes AB. All rights reserved.
Dymola® is a registered trademark of Dassault Systèmes AB.
Modelica® is a registered trademark of the Modelica Association.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Dassault Systèmes AB
Ideon Gateway
Scheelevägen 27 – Floor 9
SE-223 63 Lund
Sweden

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

Contents

1	Appendix — Installation.....	5
1.1	Installation on Windows	6
1.1.1	Dymola as 32-bit and 64-bit application.....	6
1.1.2	Installing the Dymola software.....	6
1.1.3	Installing a C compiler	10
1.1.4	Installing the Dymola license file	22
1.1.5	Additional setup.....	30
1.1.6	Changing the setup of Dymola	58
1.1.7	Removing Dymola.....	59
1.1.8	Checking for updates	59
1.1.9	Installing updates.....	60
1.2	Installation on Linux	61
1.2.1	Installing Dymola	63
1.2.2	Additional setup.....	64
1.2.3	Removing Dymola.....	66
1.3	Dymola License Servers on Windows	67
1.3.1	FLEXnet Publisher license server.....	67
1.3.2	Dassault Systèmes License Server (DSLS)	79
1.4	Dymola License Server on Linux.....	80
1.4.1	FLEXnet Publisher license server.....	80
1.4.2	Dassault Systèmes License Server (DSLS)	82
1.5	Utility programs	83
1.5.1	Obtaining a host id.....	83
1.6	System requirements	84
1.6.1	Hardware requirements.....	84

1.6.2	Hardware recommendations	84
1.6.3	Software requirements	85
1.6.4	Dymola license server	88
1.7	License requirements	90
1.7.1	General	90
1.7.2	License for Dymola – Simulink interface.....	91
1.7.3	License for real-time simulation	91
1.7.4	Licenses for exporting code.....	91
1.7.5	Licenses for executing/importing/using code	93
1.7.6	Licenses for libraries in the Dymola library menu	94
1.7.7	Licenses for libraries not in Dymola library menu	96
1.8	Troubleshooting	96
1.8.1	License file	96
1.8.2	Compiler problems	99
1.8.3	Simulink	99
1.8.4	Change of language	100
1.8.5	Other Windows-related problems.....	100
2	Index	101

1 Appendix — Installation

This chapter describes the installation of Dymola on Windows and Linux, and related topics. The content is the following:

In section 1.1 **”Installation on Windows”** starting on page 6 the installation on Windows is described, including installation of Dymola software, C compiler and license (sharable or node-locked). The sub-section “Additional setup” starting on page 30 treats specific issues as installing Dymola as administrator on a computer that should be used by non-administrators and remote installation of Dymola. Finally, change of setup, removal of Dymola and installing updates are described.

In section 1.2 **”Installation on Linux”** starting on page 61 the installation on Linux is described, in a similar way as the previous section. The sub-section “Additional setup” starting on page 64 describes e.g. compilation of model code and simulation from the command line.

In section 1.3 **”Dymola License Servers on Windows”** starting on page 67 the installation of a license server on Windows is described, as is the borrowing of licenses.

In section 1.4 **”Dymola License Server on Linux”** starting on page 80 the installation of a license server on Linux is described, as is the borrowing of licenses.

In section 1.5 **”Utility programs”** starting on page 83 a utility program for finding a host id on a computer is described.

In section 1.6 **”System requirements”** starting on page 84 the hardware and software requirements/recommendations are listed.

In section 1.7 “**License requirements**” starting on page 90 the license requirements for various features are listed.

In section 1.8 “**Troubleshooting**” starting on page 96 the solution to various problems are described. It might be license file problems, compiler problems, issues with Simulink, change of language etc.

1.1 Installation on Windows

This section refers only to the Windows version of Dymola.

To install Dymola the following tasks must be performed:

- Install the Dymola software and libraries.
- Install a C compiler (if it has not been done before).
 - Install the Dymola license file.
 - Install a license server (sharable license only).

Following installation, the user may do additional setup. The installation of updates and removal of Dymola is also described below.

1.1.1 Dymola as 32-bit and 64-bit application

From Dymola 2018 FD01, the Dymola program is only available as a 64-bit application. The Dymola program (and its associated DLLs) is located in the folder `Program Files\Dymola 2025x Refresh 1\bin64` after installation.

1.1.2 Installing the Dymola software

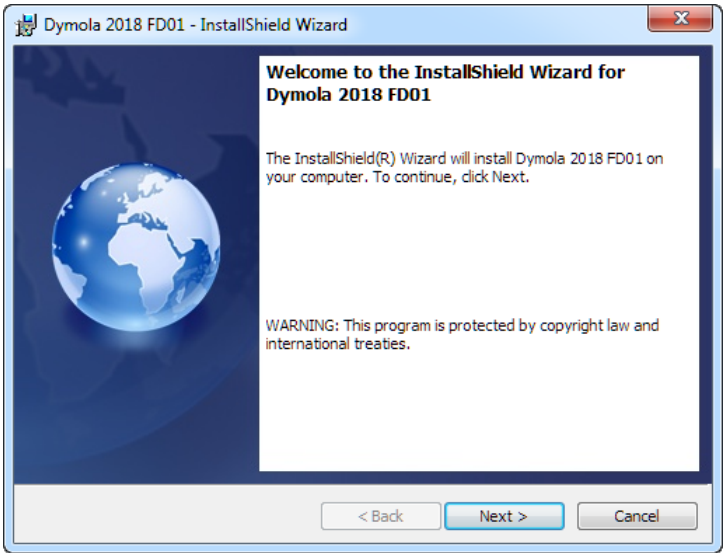
Dymola and appropriate libraries is distributed on a single DVD or downloaded electronically. With electronic download, the DVD-image is provided as two separate .zip files. **Note** that both zip-files must be extracted to the same location before starting the installation.

Starting the installation

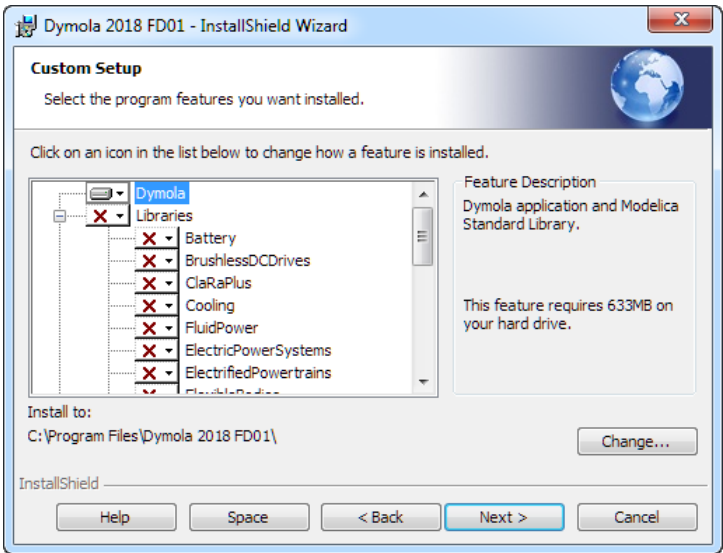
Please note that Administrator privileges are required for this installation. When Dymola has been installed, any user can run it.

The installation normally starts when you insert the distribution DVD. If autostart has been disabled, please start `D:\setup.exe` (assuming your DVD drive is labeled D) from Windows Explorer by double clicking on the file or use the **Start** button in Windows, select **Run**, enter `D:\setup.exe` and click **OK**.

Dymola installation setup.



Clicking **Next>** will display license conditions that must be accepted in order to proceed. Accepting by selecting that alternative and then clicking **Next>** will display the following:

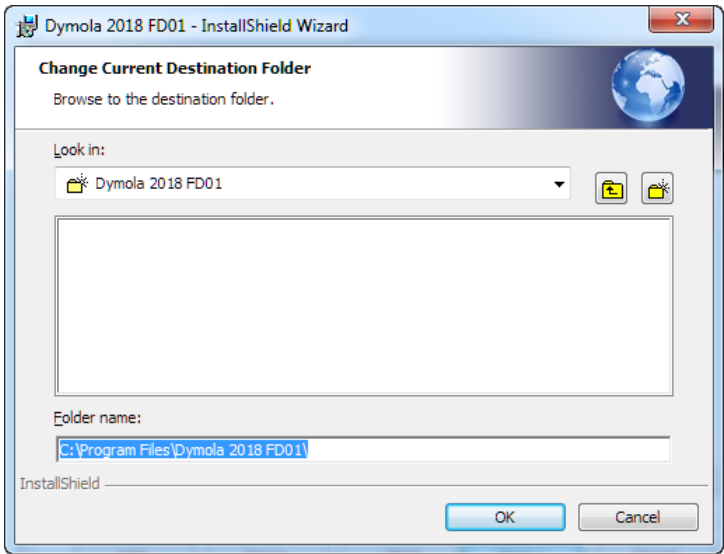


Location of installation directory

The first choice in the installation procedure is the type of installation and the name of the Dymola installation directory. The default is: C:\Program Files\Dymola + the version number of Dymola.

Dymola installation directory.

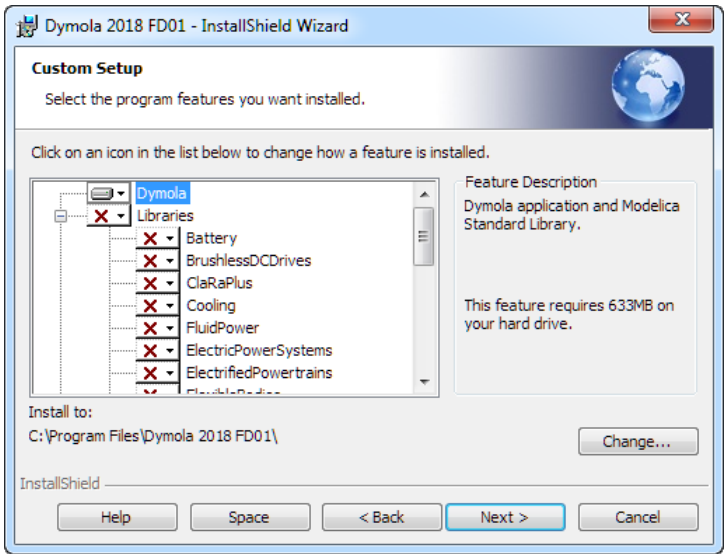
This path is displayed under `Install to:`. If the path should be changed, click on the **Change...** button. Here the path can be changed; a change has to be acknowledged by clicking **OK**.



Selecting components

The second choice is to select optional components of the distribution. By unselecting components some space can be saved.

Component selection.

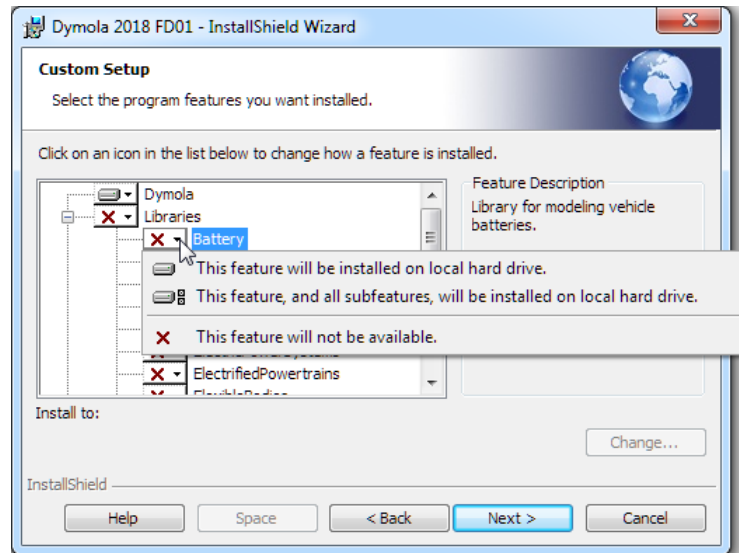


The first alternative **Dymola** is the default contents of the Dymola distribution, including the development environment and the Modelica standard library. This component should always be installed (except when only a license server should be installed).

The **Libraries** section contains several commercial libraries that require a license option to use. Install libraries according to your current options.

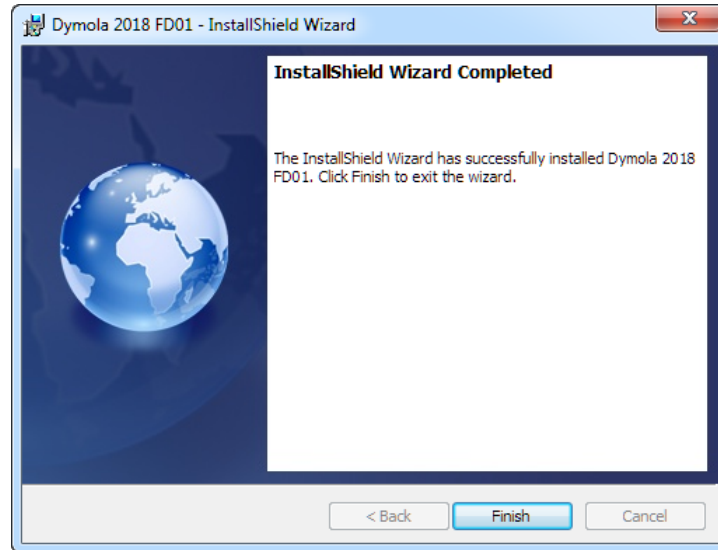
The last section, **License server**, makes it possible to install Dymola license server without having to install Dymola. Please note that the **Dymola** component should be unchecked in that case.

To add/remove a component from the installation, click on it and select the appropriate alternative in the menu.



When Dymola is successfully installed the following will appear:

**Installation of Dymola
has finished.**



1.1.3 Installing a C compiler

To translate models in Dymola you must also install a supported C compiler. The C compiler is not distributed with Dymola. The C compiler needs to be installed only once, even if you install multiple versions of Dymola. You can select a Microsoft compiler or a MinGW GCC compiler. Furthermore, you can select to cross-compile for Linux on Windows. For some compiler you can select to use Clang.

The C compiler can be installed before or after you install Dymola. You can run Dymola and browse models, but to translate any model you must install the C compiler. The selected compiler is stored as a per user setting and for the future kept for new installations of Dymola. Note that you need administrator rights to install the compiler.

Compilers

Microsoft compilers

Dymola supports Microsoft Visual Studio 2015, 2017, 2019, and 2022, the following editions:

Visual Studio 2015:

- Visual Studio Professional 2015
- Visual Studio 2017 Express for Windows Desktop

Visual Studio 2017:

- Visual Studio Professional 2017
- Visual Studio Enterprise 2017

- Visual Studio 2017 Desktop Express **Note!** This compiler only supports compiling to Windows 32-bit executables
- Visual Studio 2017 Community
- Visual Studio 2017 Build Tools **Notes:**
 - The recommended selection to run Dymola is the workload “Visual C++ build tools” + the option “C++/CLI support...”
 - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
 - This installation is not visible as specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2017 alternative: **Visual Studio 2017/Visual C++ 2017 Express Edition (15)**.
 - For more information about installing and testing this compiler with Dymola, see [Installing Visual Studio Build Tools for Dymola.pdf](#).

Visual Studio 2019:

- Visual Studio Professional 2019
- Visual Studio Enterprise 2019
- Visual Studio Community 2019
- Visual Studio 2019 Build Tools **Notes:**
 - The recommended selection to run Dymola is the workload “C++ build tools” + the option “C++/CLI support...”
 - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
 - This installation is not visible as specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2019 alternative: **Visual Studio 2019/Visual C++ 2019 (16)**.
 - For more information about installing and testing this compiler with Dymola, see [Installing Visual Studio Build Tools for Dymola.pdf](#).

Visual Studio 2022:

- Visual Studio Professional 2022
- Visual Studio Enterprise 2022
- Visual Studio Community 2022
- Visual Studio 2022 Build Tools **Notes:**
 - The recommended selection to run Dymola is the workload “Desktop development with C++” + the option “C++/CLI support...”
 - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features

- This installation is not visible as specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2022 alternative: **Visual Studio 2022/Visual C++ 2022 (17)**.
- For more information about installing and testing this compiler with Dymola, see [Installing Visual Studio Build Tools for Dymola.pdf](#).

General notes for Visual Studio:

- When installing any Visual Studio compiler, make sure that the option “C++/CLI support...” is also selected to be installed.
- Visual Studio performs part of its installation the first time it is run. This must be performed in order to use Visual Studio with Dymola, a step that requires administrator privileges. The rights can be elevated by running Dymola as administrator the first time. This is done by right-clicking the Dymola icon in the Windows Start menu as selecting **Run as administrator**. To validate this, any model (e. g. a demo) should be opened and translated in Dymola.
- Bundled Visual Studio toolsets are not supported. The workaround is to install the older build tools separately instead of bundled in e.g. a Visual Studio 2022 installation.
- For Visual Studio 2019 and 2022, you can select to clang as compiler. See below.

To download any Visual Studio compiler (including the latest one), you can use the link:

<https://visualstudio.microsoft.com/vs/older-downloads/>

The C compiler can be installed before or after you install the Dymola. You can run Dymola and browse models, but to translate any model you must install the C compiler.

To get a small improvement of the simulation performance, you can activate the global optimization in the compiler, by setting the flag

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

before generating code. (The default value of the flag is 0.)

This flag works the same for all Visual Studio compilers. Note that although the simulation performance is somewhat improved setting this flag, the compilation of the code might take substantially longer time for large models. The setting corresponds to the compiler command `/Og`.

Note. The Microsoft Visual C++ 2015 redistributable package is automatically installed.

Clang compiler

If you first select to use any of the compilers Visual Studio 2019, Visual Studio 2022, or Linux cross-compiler (WSL), you can then select to use Clang as code generator. This means that you use Clang as code generator instead of native Visual Studio or, for WSL, (normally) GCC. You activate the Clang compiler by the setting **Use clang as compiler for VS 2019, V 2022, and WSL** in the compiler setup menu (see below).

(This option is by default not activated.)

Using Clang as code generator can make the simulation less time-consuming, but the improvement is model dependent.

Note that you need to install Clang support to use it:

- For Visual Studio, search for, and select components named “clang” in the Visual Studio Installer.
- For WSL, a tip is to select Clang and try to run WSL; you will then get information about how to install Clang.

Intel compiler

From Dymola 2022, Intel compiler is not supported.

MinGW GCC compiler

Dymola 2025x Refresh 1 has limited support for the MinGW GCC compiler. The following MinGW GCC versions have been tested (hence, at least the versions in that range should work fine):

- 32-bit: MinGW GCC 6.3 and 8.2
- 64-bit: MinGW GCC 7.3 and 8.1

To download any of these free compilers, please visit

<https://sourceforge.net/projects/mingw-w64/>

Look for the “MinGW-W64 Online Installer”. On our latest visit, it was available under the tab “Files”.

Note that you need administrator rights to install the compiler.

Start the MinGW installer, then select for, respectively:

- 32-bit:
 - Architecture: i686
 - Threads: win32
 - Exception: dwarf
- 64-bit:
 - Architecture: x86_64
 - Threads: posix
 - Exception: seh

Please note:

- To be able to use other solvers than Lsodar, Dassl, and Euler, you must also add support for C++ when installing the MinGW GCC compiler. Usually you can select this as an add-on when installing MinGW GCC.

- There are currently some limitations with MinGW GCC:
 - Embedded server (DDE) is not supported.
 - Support for external library resources is implemented, but requires that the resources support GCC, which is not always the case.
 - No support for Dymola runtime concept. Consequently, FMUs (or executables `dymosim.exe`) must be exported with the code export option enabled, to be useful. (The code export option means having any of the license features **Dymola Binary Model Export** or the **Dymola Source Code Generation**.)
 - For 32-bit simulation, parallelization (multi-core) is currently not supported for any of the following algorithms: Radau, Esdirk23a, Esdirk34a, Esdirk45a, and Sdirk34hw.
 - Compilation may run out of memory also for models that can compile with Visual Studio. The situation is better for 64-bit GCC than for 32-bit GCC.

In general, 64-bit compilation is recommended for MinGW GCC. In addition to the limitations above, it tends to be more numerically robust.

Linux cross-compiler (WSL)

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

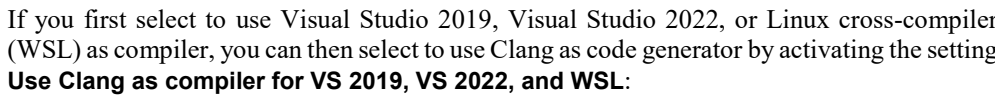
You can select to use the Clang compiler as code generator instead of (normally) GCC, see section “Clang compiler” on page 12 above.

For more information about the needed installations etc., please see “Cross-compilation for Linux on Windows” starting on page 17.

Selecting a compiler

Selecting compiler is required.

To change the compiler Dymola uses to translate the model, use the command **Simulation > Setup** and the **Compiler** tab, see also the manual “*Dymola User Manual 1B: Developing and Simulating a Model*”, the index entry “*simulation setup : compiler tab*” in the index in the end of that manual. (Below is an example of the **Compiler** tab).



C compiler

☐ Visual Studio 2015/Visual C++ 2015 Express Edition (14.0)
 ☐ Visual Studio 2017/Visual C++ 2017 Community (15)
 ☐ Visual Studio 2019/Visual C++ 2019 (16)
 ☒ Visual Studio 2022/Visual C++ 2022 (17)
 ☐ Visual Studio Custom

☐ MinGW GCC
 ☐ Linux cross-compiler (WSL)

☒ Use Clang as compiler for VS 2019, VS 2022, and WSL

☐ Verify also FMU export on success

For details about this compiler, including needed installation, see section “Clang compiler” on page 12.

If you select the MinGW GCC compiler, subchoices for the compiler are seen:

C compiler

☐ Visual Studio 2015/Visual C++ 2015 Express Edition (14.0)
 ☐ Visual Studio 2017/Visual C++ 2017 Community (15)
 ☐ Visual Studio 2019/Visual C++ 2019 (16)
 ☐ Visual Studio 2022/Visual C++ 2022 (17)
 ☐ Visual Studio Custom
 ☒ MinGW GCC

☒ 32-bit:

☒ 64-bit:

☐ Linux cross-compiler (WSL)

☐ Use Clang as compiler for VS 2019, VS 2022, and WSL

☐ Verify also FMU export on success

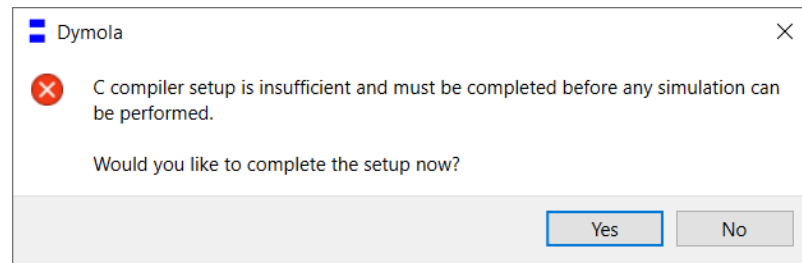
The selected compiler is stored as a per user setting and for the future kept for new installations of Dymola. Switching compiler does not modify Dymola/bin.

For information about the compiler selection **Linux cross-compiler (WSL)**, see “Cross-compilation for Linux on Windows” starting on page 17.

Advanced users can enter custom options for compiler/linker using the last two input lines in the Simulation Setup dialog. **Note** that there is one case where a linker option should not be entered here, but a flag should be used instead. The case is that in some rare occasions, different libraries have different options for what runtime version to use (/MD and /MT option). To force the same version, a flag must be used. For more information about this case, and also other flags that can be used, refer, in the manual “*Dymola User Manual 1B: Developing and Simulating a Model*”, to the index entry “*custom options*” in the index in the end of that manual.

Note the importance of the **Verify compiler** button. As an example, you cannot see from the menu if you have a valid MinGW compiler available, you must use the **Verify compiler** button to see if this is the case. (Note also that you can also perform a test of FMU export by activating **Verify also FMU export on success** before clicking **Verify Compiler**. If this option is activated, both 32-bit and 64-bit FMU export is also checked, if the compiler test itself is successful.)

During the startup of Dymola, the compiler setting is checked. If insufficient, the following message appears:



If you click **Yes**, the menu for the compiler setup is opened to let you complete the setup.

Classes that contain “Library” annotations to link with external libraries in C are supported for Microsoft Visual Studio compilers. If you link with your own C-libraries, you have to recompile them as multi-threaded; Dymola only supports linking with multi-threaded libraries in Microsoft Visual Studio compilers. For MinGW GCC compilers, see the limitations above.

Cross-compilation for Linux on Windows

Introduction

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

Install WSL

Installing WSL will give you a command-line Linux environment on your Windows computer. This environment can then compile code in a native Linux environment, such as case Ubuntu 20.04 LTS (Long Term Support) as used in the example below. We recommend Ubuntu 20 since it is the most tested version for Dymola. In particular, the integration algorithms RadauIIa, Esdirk23a, Esdirk34a, Esdirk45a, and Sdirk34hw have been configured to work with Ubuntu 20, but not with Ubuntu 18.

Note! If you want to exchange FMU's with the Systems Simulation Design app (sometimes referred as "SID"), Ubuntu 20.04 is also recommended.

The example below shows installation and setup on Windows 10 version 20H2.

The WSL Linux environment can compile the generated model C code from Dymola in order to produce a Linux executable dymosim or a Linux FMU.

To install WSL and download a suitable Linux distribution, in our case Ubuntu 20.04 LTS:

- Open a Command Window in Administrator mode.
- Install WSL and Ubuntu by giving the command:

```
wsl --install -d Ubuntu-20.04
```

You may be asked to reboot your computer to complete the installation.

- Create you user account when prompted.
- Update your system and install the C compiler and other tools (note that the last line installs Clang support):

```
sudo apt update
sudo apt upgrade
sudo apt install gcc g++ zip dos2unix
sudo apt install clang
```

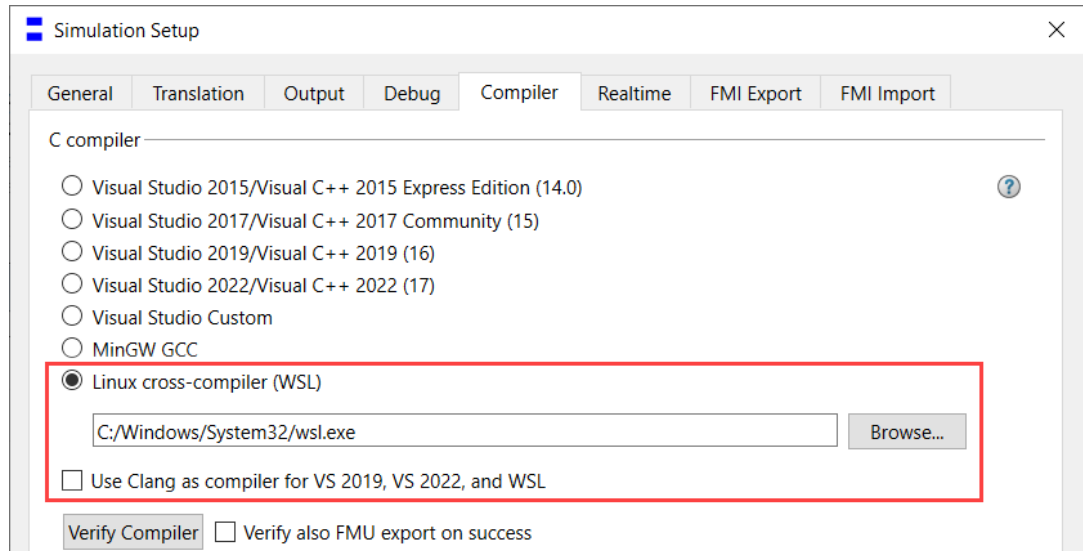
- Ensure that WSL can change file permissions. That can be done in WSL by ensuring that the file `/etc/wsl.conf` has the following two lines (and creating that file if it does not exist):

```
[automount]
options = "metadata"
```

Important! Restart WSL or reboot the computer after any changes of this file.

For a general description, about installing WSL, valid from Windows 10 version 2004, and for Windows 11, see <https://docs.microsoft.com/en-us/windows/wsl/install>. Here you can also find a link on how to install WSL on older Windows 10 versions.

To activate the WSL installation, select **Linux cross-compiler (WSL)** in the simulation setup, reached by the command **Simulation > Setup**, the **Compiler** tab:



To test the WSL installation, click **Verify Compiler**.

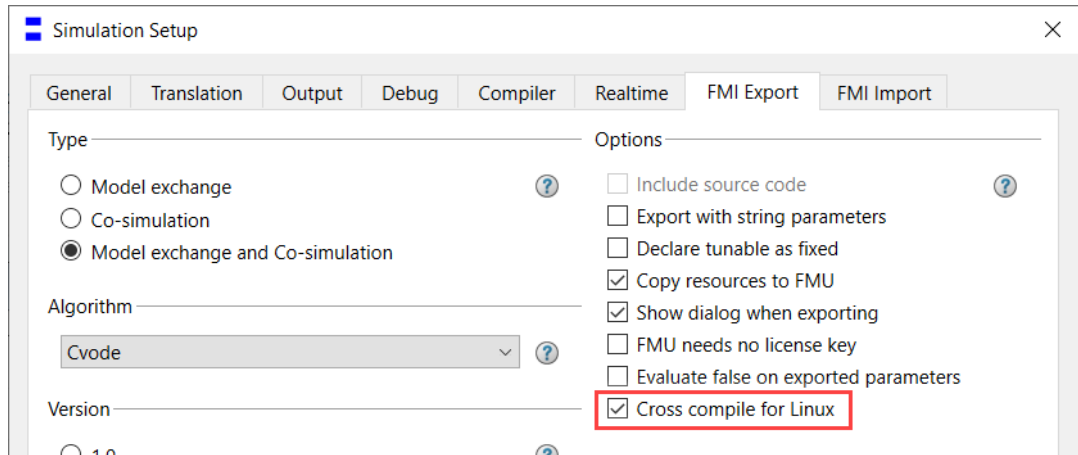
Optionally, you can select to use the Clang compiler as code generator instead of (normally) GCC. You do this by activating the setting **Use Clang as compiler for VS 2019, VS 2022, and WSL** – see the figure above. The general feature, including needed installation, is described in the section “Clang compiler” on page 12.

If you merely wish to use the WSL as the main compiler, your setup is done here. For cross-compilation for FMU exports, see next section.

The settings are saved between sessions.

FMU export for multiple platforms

Dymola partially supports cross-compilation when exporting FMUs on Windows. It is partial in the sense that your main compiler environment is limited to Visual Studio or MinGW and the cross-compiler environment is limited to WSL. So first, you need to select either Visual Studio or MinGW in the simulation setup (see the above figure). Then, to activate cross-compilation, activate the setting **Cross compile for Linux** in the simulation setup, reached by the command **Simulation > Setup**, the **FMI Export** tab:



Now, when exporting an FMU, you will get 64-bit Linux binaries in addition to Windows binaries.

Important! In this case, you should *not* select **Linux cross-compiler (WSL)**; the setting **Cross compile for Linux** sets the suitable cross-compilation.

To activate the setting **Cross compile for Linux** corresponds to setting the flag `Advanced.FMI.CrossExport=true`. (The default value of the flag is `false`.)

Note that the value of the flag/setting is *not* saved between sessions.

Scripting commands and flags for compiler handling

There are two built-in functions for handling of compilers, `GetDymolaCompiler` and `SetDymolaCompiler`. With the first one you can retrieve the settings in the **Compiler** Tab of the simulation setup, except the Custom options settings (see below). With the second you can set up a compiler (except the Custom options settings). The Custom options settings can be handled by flags.

For more information about the two built-in functions and the flags, see the manual “*Dymola User Manual 1B: Developing and Simulating a Model*”, the index entries “*GetDymolaCompiler*”, “*SetDymolaCompiler*” and “*custom options*” in that manual.

Troubleshooting: Verify compiler button and error messages

The compiler used to compile the C code generated by Dymola into executable code for simulation is set in the **Compiler** tab using the command **Simulation > Setup**.

Some potential problems can be found by pressing the **Verify Compiler** button in that tab. Any warning messages indicate problems that need to be resolved before translating a model. Pressing the button performs a number of tests:

- Validates the DOS environment (for Windows) / shell (for Linux).
- Check the Dymola installation for runtime libraries.

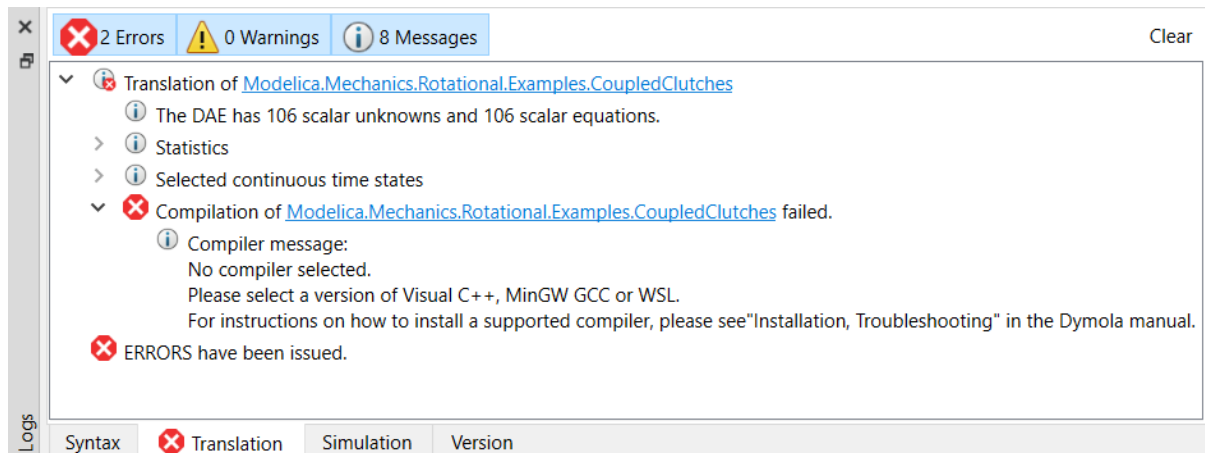
- Verifies that the selected compiler directory contains a valid compiler.
- Validates that the compiler can compile by executing a small test model.
- (For Windows) Validates that an unsupported Visual Studio compiler version has not been selected when using the **Visual Studio Custom** alternative in the setup.

The compiler is tested for both 32-bit and 64-bit mode.

(Note that you can also perform a test of FMU export by activating **Verify also FMU export on success** before clicking **Verify Compiler**. If this option is activated, both 32-bit and 64-bit FMU export is also checked, if the compiler test itself is successful.)

Error messages with information how to proceed will be displayed.

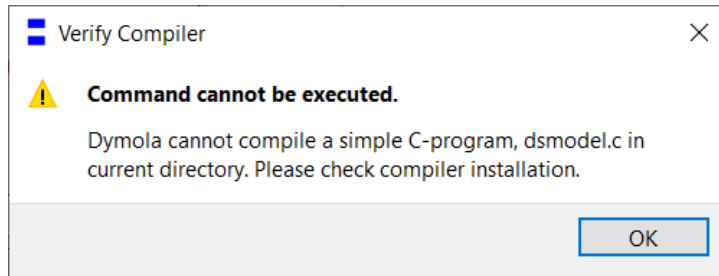
Some error messages may also be displayed in the command log. As an example, if you have not selected the compiler, and try to translate a model, you will get, in the **Translation** tab:



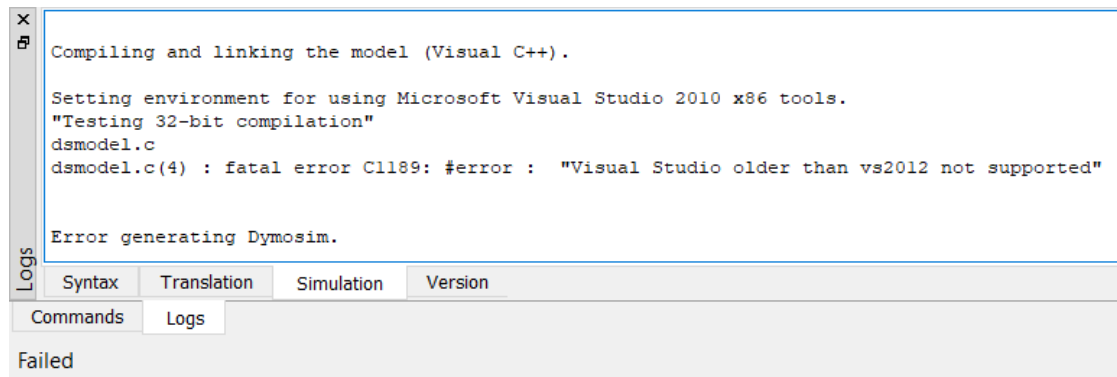
And also in the **Simulation** tab a message will appear, the text here will be:

```
No compiler selected.
Please select a version of Visual C++, MinGW GCC, or WSL.
For instructions on how to install a supported compiler, see
"Installation, Troubleshooting" in the Dymola manual.
```

If an invalid compiler version has been selected when using the **Visual Studio Custom** alternative in the compiler setup, you will have the following message (in this example, a Visual Studio 2010 compiler has been selected):



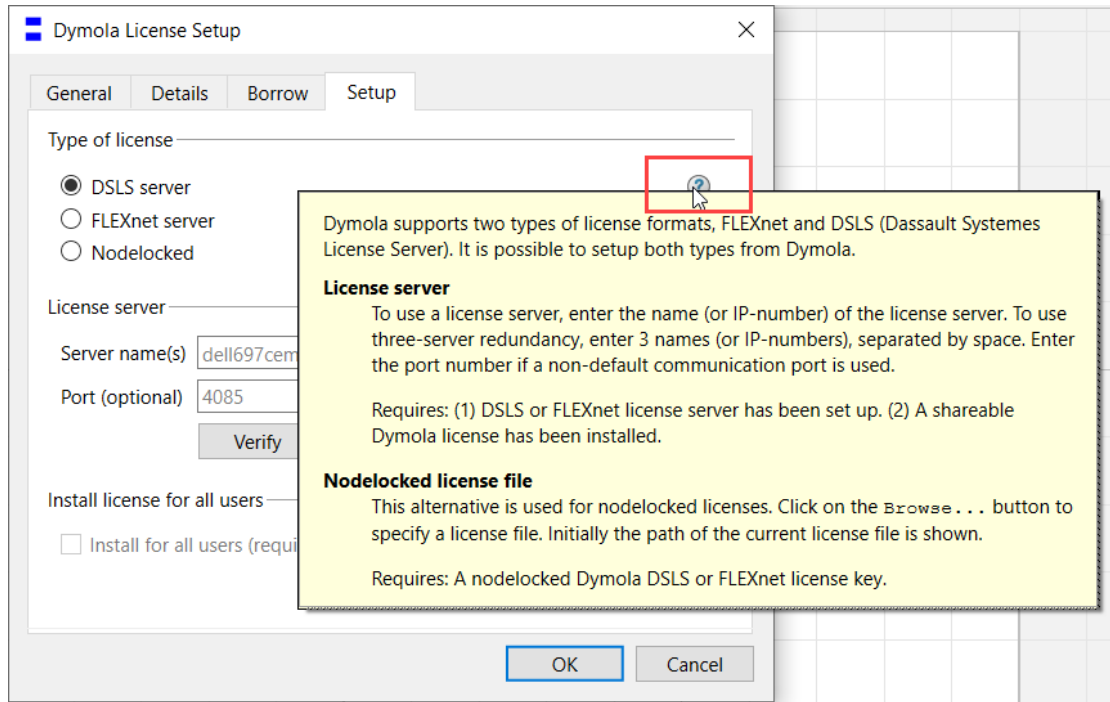
In the command log the following message is given:



1.1.4 Installing the Dymola license file

After installation, Dymola will initially start in “trial” mode. While running in trial mode you can continue with installing the license file. (For information about what license is used when starting Dymola if you already have a valid license, see section “Selection of license when starting Dymola” on page 29.)

When working with Dymola license files, you often work with the command **Tools > License Setup**, the **Setup** tab. Note the feature of clicking “?” to get information about the features:

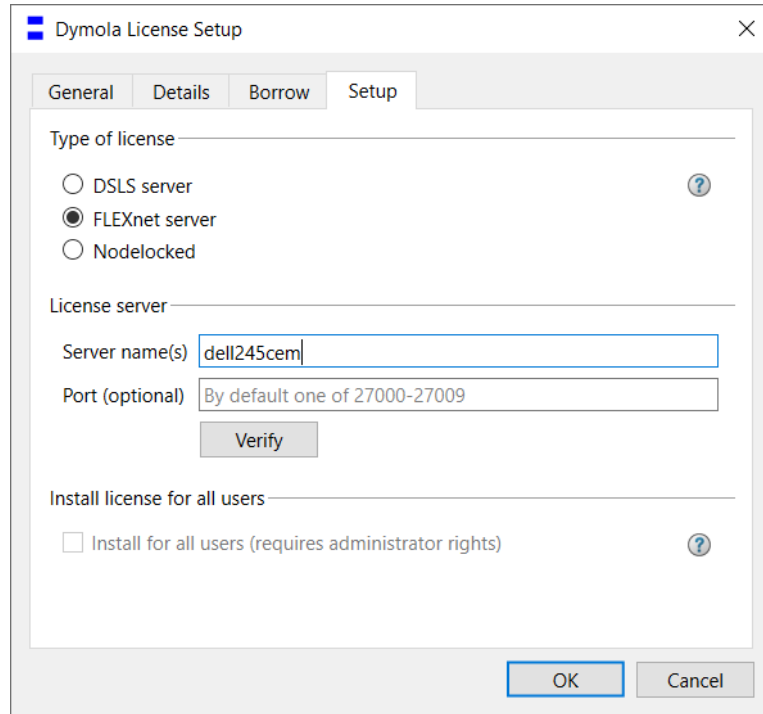


Setting up a sharable license

Sharable licenses are requested by Dymola from a license server. The information normally required on the client computer is just the name (or IP number) of the license server. (For handling of the license server itself, usually handled by IT administration, see dedicated sections about license servers later in the document.)

Start Dymola and select **Tools > License Setup** and then the **Setup** tab. Select **DSLS server** or **FLEXnet server** depending on what server you use. Enter the name or IP number of the server. If so instructed by the system administrator, also enter the port number. By default, leave this field empty. (Note that a DSLS server and a FLEXnet server use different ports, the default for a FLEXnet server is seen in the figure below, and the default value for a DSLS server is 4085.)

License server setup.



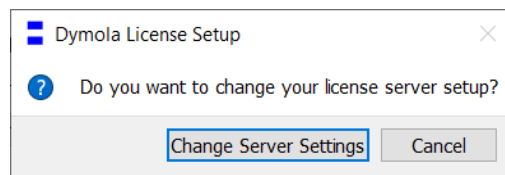
The image shows the 'Dymola License Setup' dialog box with the 'Setup' tab selected. It contains three radio buttons for 'Type of license': 'DSLS server', 'FLEXnet server' (which is selected), and 'Nodelocked'. Below this is the 'License server' section with a text field for 'Server name(s)' containing 'dell245cem' and a text field for 'Port (optional)' with the value 'By default one of 27000-27009'. A 'Verify' button is located below the port field. At the bottom of the dialog is the 'Install license for all users' section with a checkbox labeled 'Install for all users (requires administrator rights)'. The 'OK' and 'Cancel' buttons are at the bottom right.

If you want to use redundant servers, you can add three server names/IP numbers, separated by space. Note that one or three servers must be specified.

You have the option of installing the license file only for the currently logged in user, or for all users on this computer. The latter requires administrator rights.

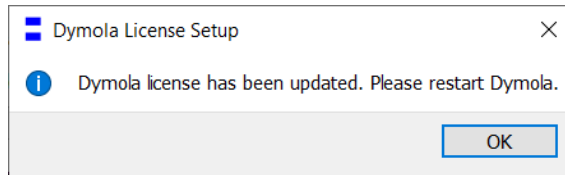
To test the selected server before changing to it, you can click **Verify**. (Note that **Verify** also works for the present server, if you have any.)

If the server you selected server is ok, and you want to select to use it, click on the **OK** button. Dymola will ask for confirmation before overwriting your old license information.



The image shows a smaller 'Dymola License Setup' dialog box with a question mark icon and the text 'Do you want to change your license server setup?'. It has two buttons: 'Change Server Settings' and 'Cancel'.

After changing the license server setup, you must restart Dymola to use the new server.



Note that when looking at the **Setup** tab when having a valid sharable license, you get information about the current license server. (You can of course still type in a new server name.)

If the server name cannot be detected, or if you have a nodelocked license, then nothing is shown in the server name field.

For information about diagnostics when connecting to the license server, see dedicated sections about license servers later in the document.

Installing a node-locked license

Node-locked licenses are stored locally on the computer running Dymola and are not shared with other computers.

Obtaining a host id

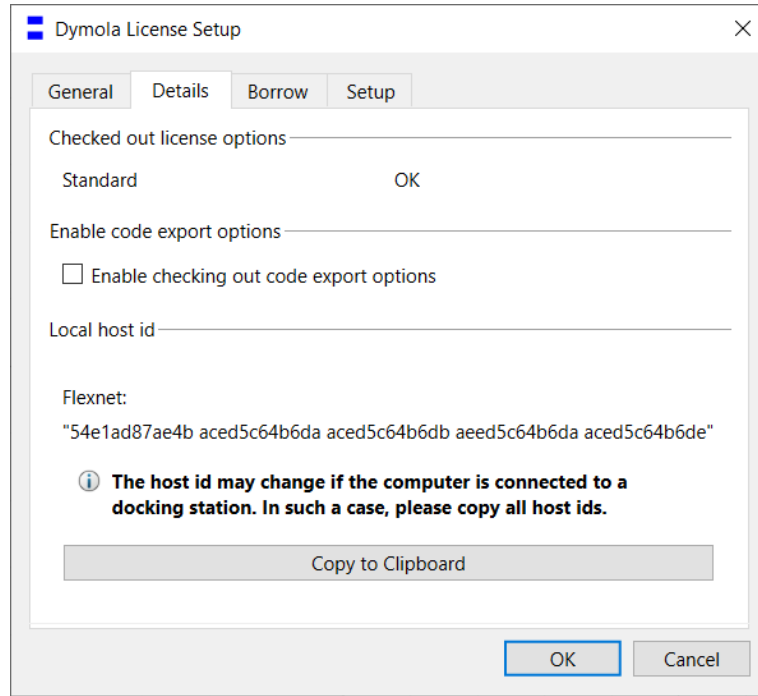
To order a *node-locked license key*, the relevant host id of the computer where Dymola should run must be supplied to your Dymola distributor. The license that you will receive will contain this information.

There are two ways finding out this host id, depending on whether a Dymola trial version is installed before or not. The host id can always be found using the utility program `hostid.exe`. Please see section “Obtaining a host id” on page 83 for more information about this program.

If the Dymola trial version has already been installed, Dymola can be used to find the host id. Start Dymola and select **Tools > License Setup**, and then the **Details** tab. Click on **Copy to Clipboard** to copy the local host id.

Please note that some laptops present different host id's depending on whether they are connected to a docking station or not. In such a case, please copy all host id's.

Local host id of the computer running Dymola.



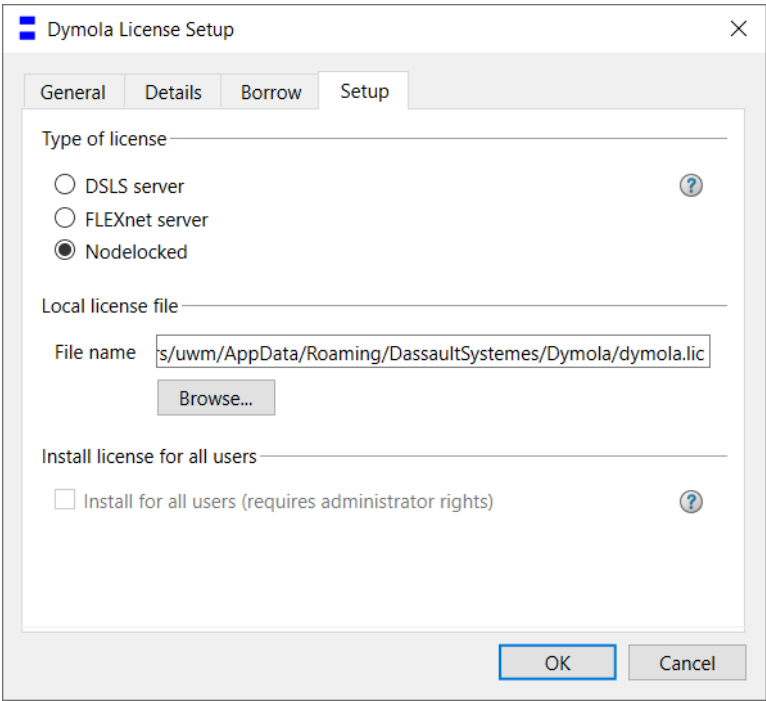
Compose an e-mail containing your local host id (host id's) and send it to your Dymola distributor. **Note.** If you order a new license, you by default will get a license in the Dassault Systemes License Server (DSLS) format. If you instead want a license in the FLEXnet license format, you must specify that when ordering the license.

Installing the nodelocked license

When you have received your license file, do save the license somewhere on your computer.

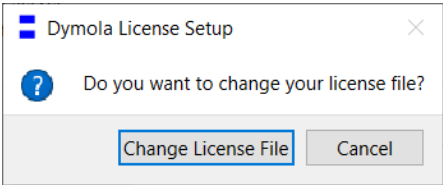
Start Dymola and select **Tools > License Setup**, select the **Setup** tab, and select **Nodelocked**. Click on the **Browse** button and open the license file you saved. Dymola automatically detects if you try to install a DSLS or FLEXnet nodelocked license. The path of the license file is shown in the dialog.

Specifying the license file.

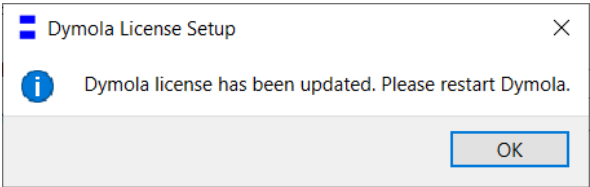


You have the option of installing the license file only for the currently logged in user, or for all users on this computer. The latter requires administrator rights.

Click on the **OK** button. Dymola will ask for confirmation before overwriting your old license information.

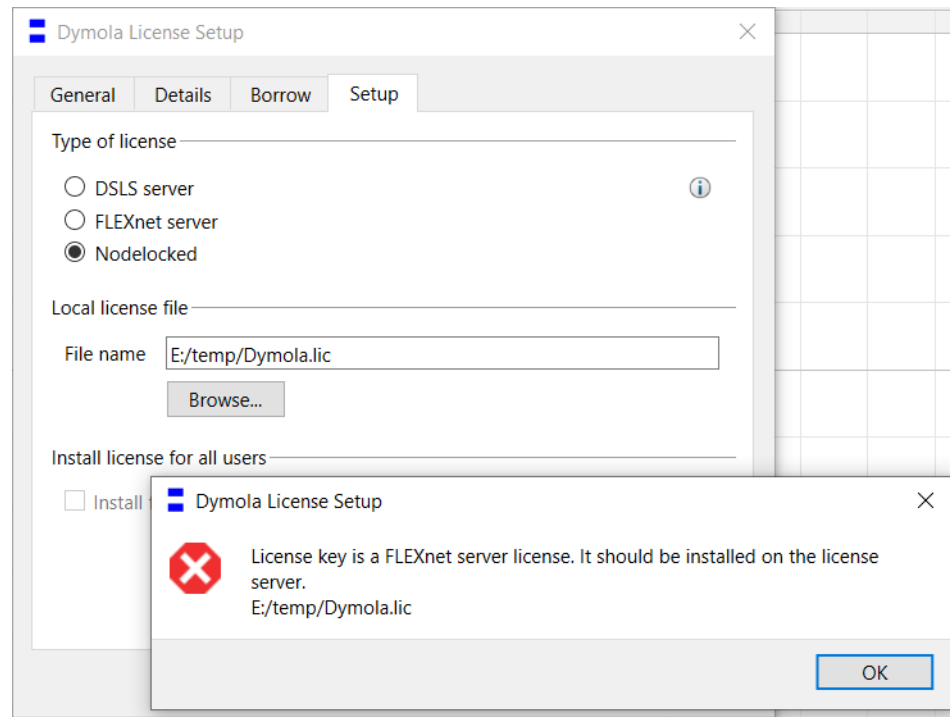


After changing the license server setup, you must restart Dymola to use the new server. You may delete the saved license file; Dymola has created a copy.



Note that you will always see the path of the nodelocked license used in the dialog.

If you by mistake try to install a license file for the license server as your local nodelocked license, you will get a message:



Specifying the license key by the environment variable DYMOLA_RUNTIME_LICENSE

In some specific cases, you need to point to the license key explicitly. To specify the license key (when not using the default license file), it is possible to set the environment variable DYMOLA_RUNTIME_LICENSE. This variable can either point to a license file or specify a license server.

An example pointing to a license file could be:

```
set DYMOLA_RUNTIME_LICENSE=C:\My Documents\dymola.lic
```

An example pointing to a license server could be:

```
set DYMOLA_RUNTIME_LICENSE=27000@my.server.com
```

The main use of this feature is for the Dymola runtime concept below.

Dymola runtime concept

Models developed by users that lack export options can still be run at other computers using a runtime concept. (That is, the executable `dymosim.exe` or an FMU can be put on another

computer and be executed on that computer.) The Dymola runtime concept requires the user of the model to have a Dymola Standard license. The license file should be defined by the environment variable `DYMOLA_RUNTIME_LICENSE`, see above.

Note that the runtime concept is not supported for the following cases:

- Using the MinGW GCC compiler, see limitations in section “MinGW GCC compiler” starting on page 13.
- Using a DSLS license, see limitations in section “Dassault Systèmes License Server (DSLS)” starting on page 79. The limitation is also valid for nodelocked DSLS licenses.

For information about license requirements in general, see section “License requirements” starting on page 90.

For more specific information about export options in particular, see the manual “*Dymola User Manual 2B: Simulation Interfaces and Export*”, chapter “Simulation Environments”, section “Code and Model Export”.

Selection of license when starting Dymola

When you install Dymola the first time, the variable `Advanced.License.Type` is set accordingly. The variable can have three values:

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

When starting a new session, using the command `dymola.exe` or starting Dymola from the Start menu, the value of the variable is used to specify the license format to use.

To force Dymola to use a DSLS license, you must start Dymola with the command `dymola.exe /DSLS` (as you had to do in any case to use a DSLS license in Dymola 2023 and earlier). Using this command ignores the value of the flag above.

To force Dymola to use a FLEXnet license, you must start Dymola with the command `dymola.exe /FLEXnet`. Using this command ignores the value of the flag above.

The options above are not case sensitive.

If a valid license is not found, Dymola will start in trial mode.

Termination of Support (TOS) in license

For each recent licensed Dymola feature (Dymola itself, libraries, etc.), a support period is part of the license. The date when the support period of a feature expires is displayed when you use the command **Tools > About Dymola** as well as if you call the built-in function `DymolaLicenseInfo()`. The expiration date of this support period is often referred to as TOS (Termination of Support).

There is a relation between TOS and WWGA, WWGA is the Dymola release date of a certain Dymola version (Word Wide General Availability date).

Some notes:

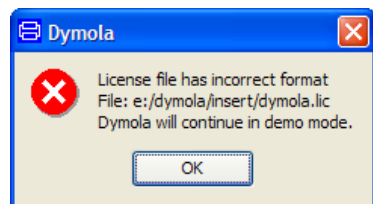
- Support entitles you to run new versions of Dymola. When you have reached TOS, you are no longer entitled to versions that are more recent (WWGA date more recent than TOS date). To use new versions, you must update your support period.
- TOS becomes effective when you install a newly generated license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

Upgrading from Dymola 6.1 and earlier

The license file format of Dymola has been upgraded to include the latest security technology. For that reason, license files for earlier versions of Dymola are not compatible with Dymola 7.0 and later, and license files for Dymola 7.0 and later are not compatible with older versions of Dymola.

If Dymola 7.0 and later finds an old license file at start-up, a diagnostic message about incorrect license file format is displayed. Dymola will then continue execution in trial mode.

**Dymola has started
with an old license file.**



1.1.5 Additional setup

Language

The selection of translation file for Dymola is by default based on the regional language setting.

Please note that currently only English and Japanese are available in Dymola. This means that if you have the Japanese language pack installed, but by default want to have Dymola in English, you must make sure that English comes before Japanese in the regional language

order. As an example, if the order is French, Japanese, and then English, Dymola will use Japanese, since there is no French translation of Dymola.

If the regional language setting should not be used, there are two ways of overriding it.

The first is to use a command line setting of the language: `-translation <language>`. Two examples are important:

- A customer that wants to run Dymola in Japanese on a machine with regional language setting other than Japanese. This can be done by starting Dymola with the command **"C:\Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe" -translation ja** (given using a 64-bit Dymola from the default location).
- A customer that wants to run Dymola in English on a machine with regional language setting Japanese. This can be done by starting Dymola with the command **"C:\Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe" -translation none** (given using a 64-bit Dymola from the default location).

The second way to override the default selection of translation file is to specify what translation file to use, this can be done with the command line option `-translationfile "<filename.qm>"` when starting Dymola. One specific opportunity here is to use a translation file other than the one in the Dymola distribution. The file can be located anywhere on the machine, since the command line option demands the path of the file to be specified. An example could be to start Dymola with the command **"C:\Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe" -translationfile "E:\Extra\NewJapaneseTranslationFile.qm"** (given using a 64-bit Dymola from the default location, and a translation file NewJapaneseTranslationFile located in E:\Extra).

Note that command line options can be included in shortcuts to Dymola, see section "Creating shortcuts to Dymola" below.

Location of startup directory

The startup directory is the directory that is defined as the current working directory when starting Dymola. The current working directory is used as default location for opening files and for saving simulation results.

Having started Dymola, you can change both the startup directory and the current working directory (they do not necessarily have to be the same). For more information, refer, in the manual *"Dymola User Manual 1B: Developing and Simulating a Model"*, to the index entry *"directory : startup directory"* and *"directory : current working directory"* in the index in the end of that manual.

When installing Dymola, an environment variable DYMOLAWORK is automatically defined. It is set internally to the startup directory. If this directory is not set, the variable is set to `...Documents\Dymola`. That subdirectory will be created if it doesn't exist.

Note. The environment variable DYMOLAWORK can only be changed internally from Dymola and cannot be used to change the startup directory.

Using high resolution (4K) screens

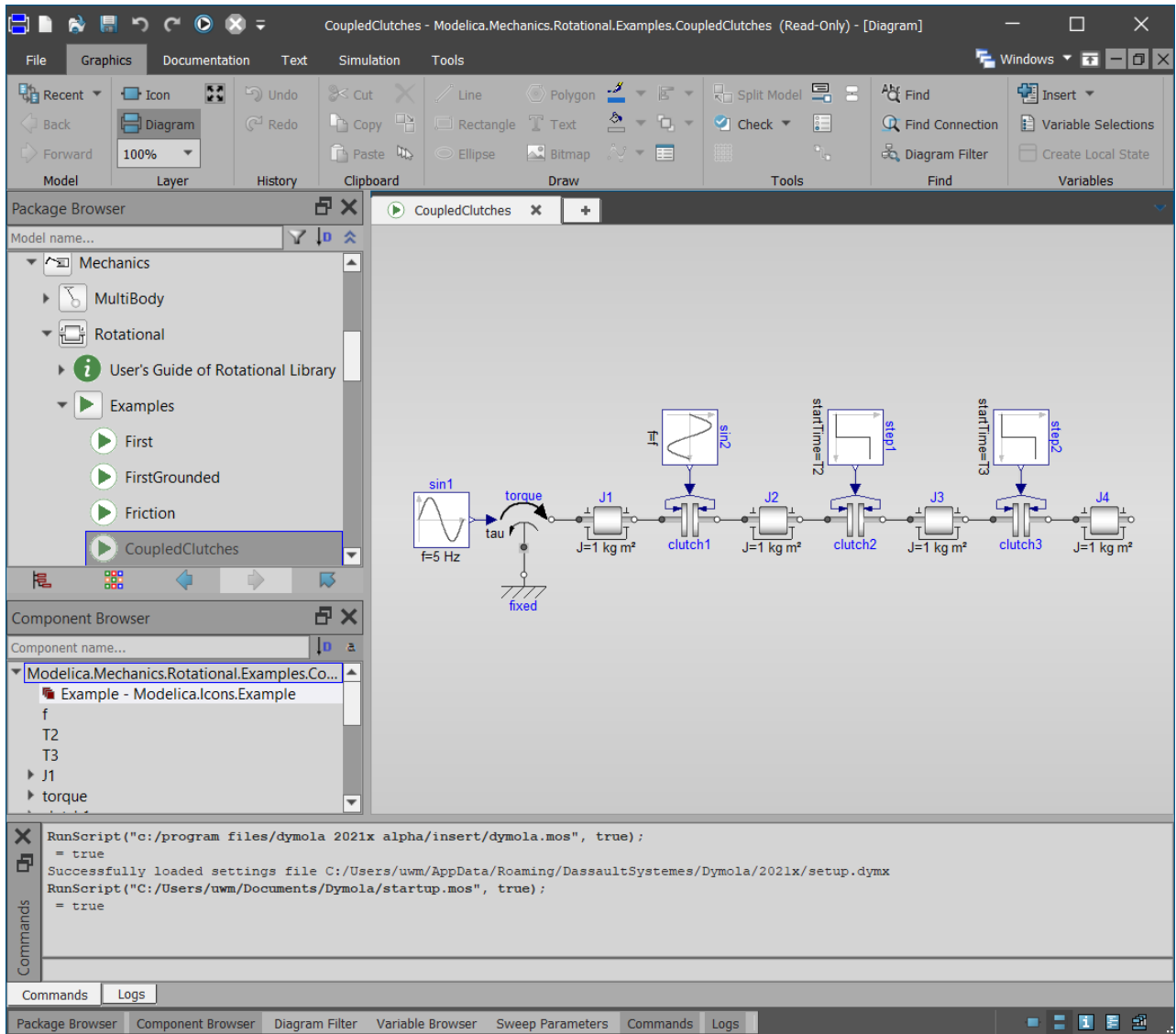
In general, it is recommended to use Windows 10 when using 4K screens. To work with DPI scaling in Dymola, you can activate the setting **Improved high DPI support** in the Options dialog, reached by the command **Tools > Options**, the **General** tab. Then you must restart Dymola for the setting to have effect.

The setting is by default not activated. The setting is saved between sessions.

(You can also activate DPI scaling in Dymola by starting Dymola.exe with the command line switch `-highdpi` from the command prompt of Windows.)

Support for Dymola in “dark mode”

To change Dymola to work in dark mode, you can activate the setting **Dark mode** in the Options dialog, reached by the command **Tools > Options**, the **General** tab. Then you must restart Dymola for the setting to have effect. This will start Dymola with an alternative darker color theme:



The setting is by default not activated. The setting is saved between sessions.

(You can also start Dymola in dark mode by using the command line switch `-dark` or `/dark`.)

To enable the traditional white (high contrast) background in the graphical editor also in “dark” mode, you can set the flag:

```
Advanced.UI.HighContrastDiagram = true
```

(The flag is by default `false`.) This can help if a model is hard to see against the grey background.

Shortcuts to start Dymola with startup script.

Creating shortcuts to Dymola

Sometimes it is convenient to create shortcuts to the Dymola program, typically to make Dymola use a startup script to, for example, open specific packages and set flags.

A shortcut is created as follows:

1. Click the right mouse button on the desktop.
2. Select **New > Shortcut** from the popup menu.
3. Browse for the Dymola program `Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe`.
4. Enter a suitable name and finish the creation of the shortcut.
5. Right-click on the newly created shortcut.
6. Select **Properties** from the popup menu.
7. Select the **Shortcut** tab of the dialog window.
8. If wanted, add command line options in the **Target** field. In our example, to add a startup script `Dymola_startup.mos` that is located in `E:\MyExperiments\MySettings\` as command line option, the full line after adding this option will be `"E\Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe" "E\MyExperiments\MySettings\Dymola_startup.mos"` (Note that there must be a space between the two paths.)
9. Note that from Dymola 2017 FD01, the working directory can be handled automatically in the settings file; it is not needed to change anything in the **Start in** field. See, in the manual *"Dymola User Manual 1B: Developing and Simulating a Model"*, the index entry *"directory:startup directory"* in that manual for finding more information.
10. Click **OK** to create the shortcut.

Preventing splash screen at startup

By default, when starting up Dymola, a splash screen is shown:



If you don't want any splash screen at startup, you can start Dymola using any of the options `-nosplash` or `/nosplash`. As an example, applying the first option in a shortcut for starting Dymola as in the previous section would be:

```
"E:\Program Files\Dymola 2025x Refresh1\bin64\dymola.exe" -nosplash
```

Remote installation of Dymola

Dymola (whether downloaded as a zip file or on DVD) consists of a number of files. Remote installation of Dymola is possible. For example, the following command makes a quiet installation of Dymola and all libraries with Modelica version 3:

```
setup.exe /s /v"INSTALLLEVEL=201 /qn"
```

The value of the `INSTALLLEVEL` property controls which components are installed, presently only 201 and unspecified is used. Unspecified means omitting the commercial libraries, and the command is:

```
setup.exe /s /v" /qn"
```

Note that you need to run any of these commands as administrator, one way is to start the command shell as administrator.

Working with a Modelica version that is newer than the one in the distribution.

To work with a Modelica Standard Library (MSL) version that is newer than the one in the Dymola distribution, you must:

- Download and install the new MSL version. You can use on-demand installation; see “On-demand downloading and installation of libraries” on page 42.
- Set the global flag `Advanced.PlaceDymolaSourceFirst=2`. For more information of the flag, see below.
- If you want to use the new version as default version, do the following:
 - Apply the command **Edit > Options...**, select the **Versions** tab, select the MSL version you want, and tick **Force upgrade of models to this version**.
 - Click **OK** to save the new default version.

Note! The libraries in the Dymola distribution may not support the new MSL version.

The reason why you must set this flag is that there are links in Modelica Standard libraries to internal libraries that are used when compiling models. If `dymola/source` and `dymola/bin/lib` are loaded before MSL, they will be used instead of the corresponding newer internal libraries, and this will not work. The `Advanced.PlaceDymolaSourceFirst` controls in what order files are loaded. The possible values are:

- 0 place `dymola/source` first for compilation
- 1 (default) presently this value works the same as the value 0
- 2....always place `dymola/source` last

Installing earlier Modelica versions including compatible libraries

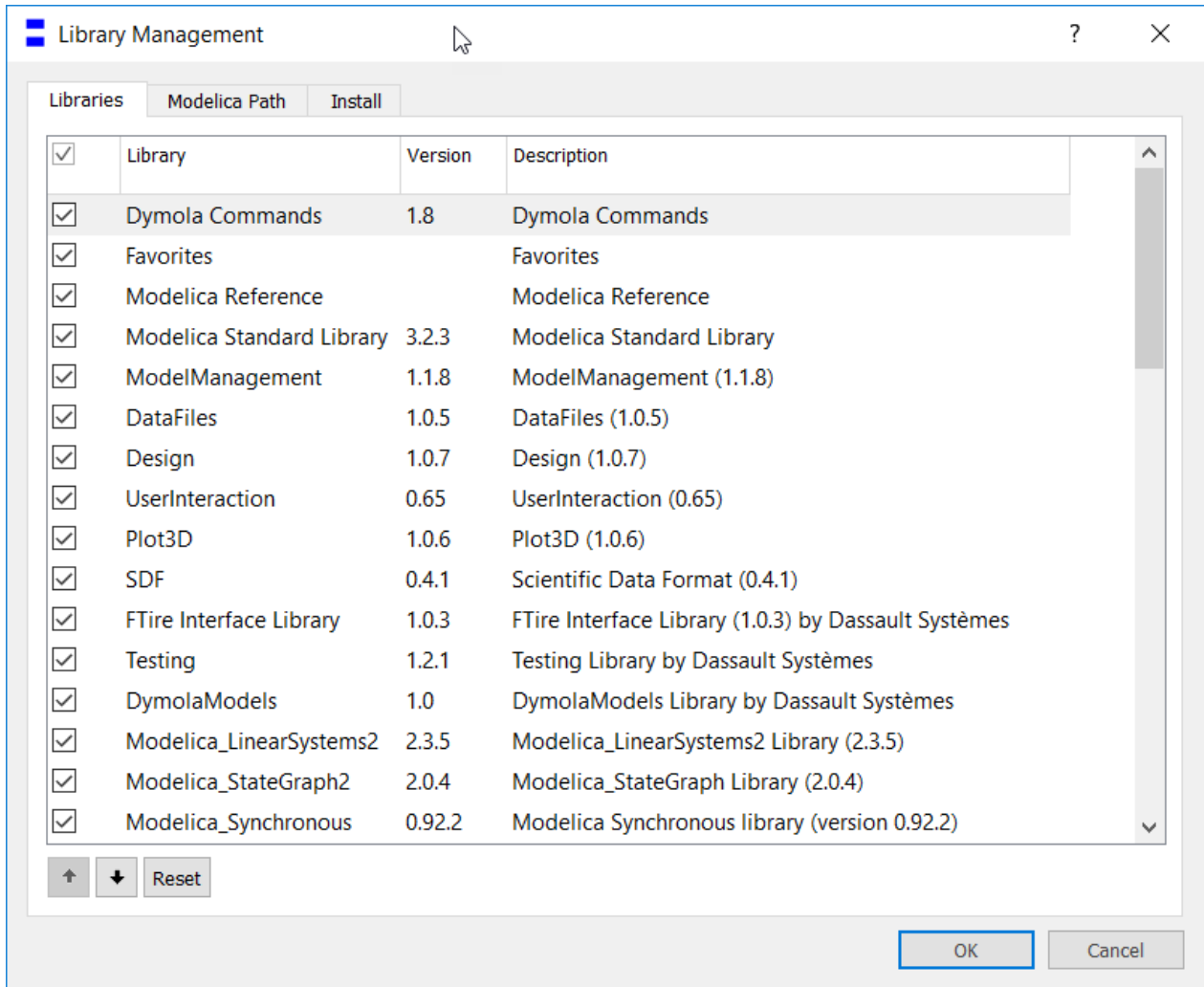
If you are working with MSL 4.0.0 and need to work with an older Modelica model that has not been converted to the new MSL version, you can yourself install MSL 3.2.3 libraries separately. They are included in the Dymola media as `extra\CompatibilityLibraries\MSL_3.2.3.zip`. To use them, unpack them in, typically, `...Program Files\Dymola 2025x Refresh 1\Modelica\Library`. **Note.** Do not forget to apply the command **Tools > Options**, select the **Version** tab, select the Modelica version you want, and tick **Force upgrade of models to this version** to select the proper version.

Please restart after having changed the version, to make sure that you handle also already loaded libraries, if any.

If you want to install older Modelica versions than that in the distribution, please contact support at <https://www.3ds.com/support>. (Note that Dymola 2018 FD01 and newer do not support Modelica 2. Conversion from Modelica 2 to Modelica 3 is still possible in Dymola 2018, but it is easier to verify the conversion in a previous version of Dymola, e.g. Dymola 2015, where Modelica 2 is still fully supported. See [MigrationModelica2.pdf](#) for more information.)

Customizing the File > Libraries menu

You can customize the **File > Libraries** menu by the command **Tools > Library Management**, the **Libraries** tab:



Notes!

- The content of this tab depends on the MODELICAPATH environment variable, which can be changed by for example the **Modelica Path** tab – if you make changes in the **Modelica Path** tab, for example, adding directories, and click **OK** in that tab, the content of the **Libraries** tab may change. See below for more information about MODELICAPATH.
- When you want to install downloaded libraries, you can also select if they should appear in the **File > Libraries** menu when installing, if you use on-demand installation. See “On-demand downloading and installation of libraries” on page 42.

In the Libraries tab you can:

- Include or exclude all libraries by using the checkbox in the heading
- Include or exclude any library in the list by the checkbox in front of it
- Move the selected library upwards or downwards using the arrow buttons

Any action is immediately implemented when clicking **OK**.

The setup is saved between sessions.

To restore the default library configuration, you can click **Reset** in the menu above.

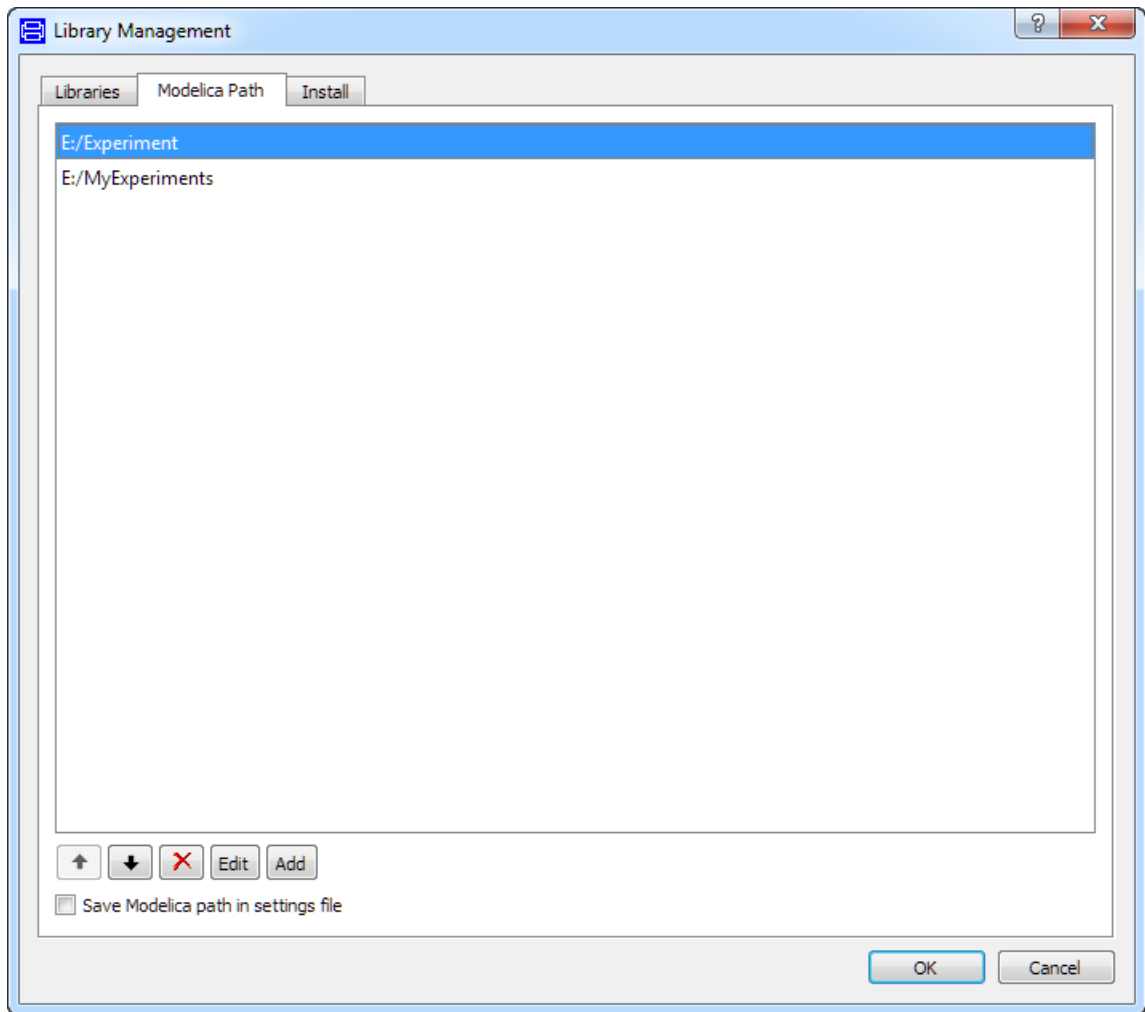
Managing library locations

The environment variable MODELICAPATH and how to manage it

Dymola will find libraries by searching all directories in the environment variable MODELICAPATH. If not set by the user, MODELICAPATH contains only dymola/Modelica/Library.

Note the option to use enclosing package paths in addition to MODELICAPATH in the search, see next section.

It is possible to manage the MODELICAPATH from within Dymola. The most convenient way is to use the command **Tools > Library Management**, the **Modelica Path** tab. This opens a dialog that makes it possible to change, add, or delete directories in the environment variable. An example of the dialog is:



Note that the default path `dymola\Modelica\Library` is not displayed.

You can use the buttons in the dialog to sort, delete, edit, and add paths.

Keys can be used for the actions as well:

- Move up: **Ctrl+Up**
- Move down: **Ctrl+Down**
- Edit: **Space**
- Add: **Plus**

If you only want to add or delete a directory, you can also use the built in function

```
AddModelicaPath(path, erase=false);
```

The function appends a directory, specified by the string variable `path`, to the Modelica path (if `erase=false` (default). If `erase=true` the specified directory is instead erased from MODELICAPATH).

An alternative is to use

```
Modelica.Utilities.System.setEnvironmentVariable("MODELICAPATH"  
, "...");
```

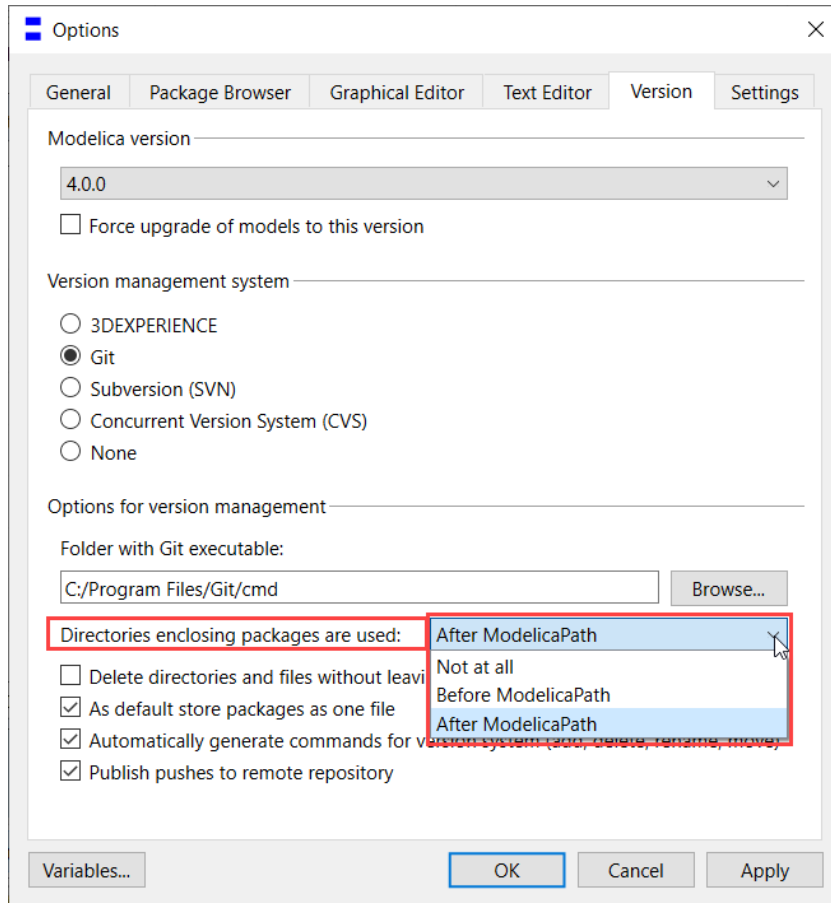
Notes:

- The menus **File > Libraries** and **File > Demos** are rebuilt after a change of MODELICAPATH.
- To keep the change of MODELICAPATH between sessions, it must be saved. This can be done by ticking the setting **Save Modelica path in settings file** in the menu above. This checkbox corresponds to the checkbox **Modelica path** in **Tools > Options**, the **Settings** tab.

Option to also use enclosing package paths in addition to MODELICAPATH

To make it easier to, for example, automatically find multiple libraries that are stored in the same directory, there is an option to also use directories enclosing packages in addition to MODELICAPATH.

This option is available as **Directories enclosing packages are used** in the **Version** tab reached by the command **Tools > Options**:



- **After ModelicaPath** (default) means that the enclosed directories are searched after having searched the paths in MODELICAPATH.
- **Before ModelicaPath** means that the enclosed directories are searched before searching the paths in MODELICAPATH.
- **Not at all** means that only the paths in MODELICAPATH are searched.

The option is also available as a flag `Advanced.Modelica.EnclosingDirectoryModelicaPath`. The value of the flag can be 0 ("Not at all"), 1 ("Before"), or 2 ("After"). The default value is 2.

Note that the content of MODELICAPATH is not changed; the option specifies additional locations to use.

Handling library locations when loading multiple libraries with dependencies from scripts

For an example of using MODELICAPATH etc. when loading libraries, see the manual “*Dymola User Manual 1B: Developing and Simulating a Model*”, the index entry “*library : loading multiple libraries with dependencies from scripts*”.

On-demand downloading and installation of libraries

The command **Tools > Library Management**, the **Install** tab, allows you to install already downloaded libraries, or to download and install libraries from GitHub.

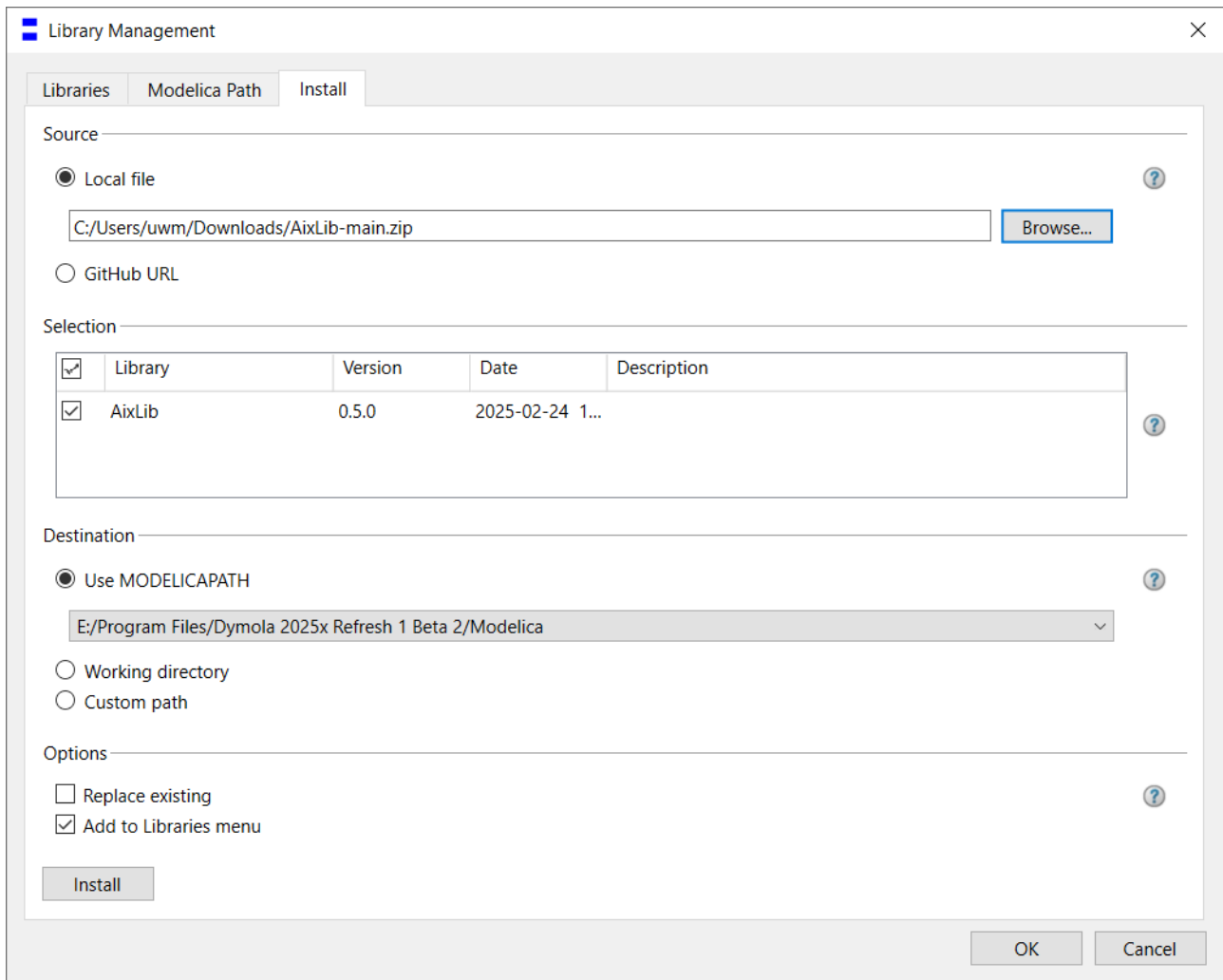
In the examples below the same library is selected, in the first case it is first downloaded and then installed, in the second case it is downloaded as part of the installation.

On-demand installation of downloaded libraries

To use on-demand installation of downloaded libraries, you can use the command **Tools > Library Management**, the **Install** tab, and then select **Local file** in the **Source** group (this is the default). Now you can browse for an mo/moe file or a zip file and install from this source. When loaded, the file content is presented for review before installation. As zip files may contain several packages, checkboxes are provided to enable partial installations.

Only reasonable combinations of packages within zip files are considered. In particular, once a package.mo or package.moe is found on a certain folder level, deeper searches for other packages are truncated.

As an example, consider using the downloaded file `AixLib-main.zip` as a source (note that it may take some time to extract the information for the **Selection** group):



In this case there was only one package in the zip file.

Here we have chosen to use a folder from MODELICAPATH¹ as destination. Note that some destinations may require running Dymola as Administrator (Windows) or root (Linux).

You may also choose whether to replace any existing library or whether to add the library in the **File > Libraries** menu or not. If **Add to Libraries menu** is checked, the file

¹ For more information about MODELICAPATH, see previous section.

`libraryinfo.mos`² is copied³ or created, else it is omitted. In the former case, the `MODELICAPATH` environment variable is updated if necessary.

On-demand downloading and installing libraries from GitHub

To download and install libraries from GitHub, select **GitHub URL** in the **Source** group:

The screenshot shows the 'Library Management' dialog box with the 'Install' tab selected. The 'Source' section has 'GitHub URL' selected. The 'Library' field contains 'URL starting with github.com/ or search phrase' and the 'Tag' field is empty. The 'Next' button is disabled. The 'Selection' section shows an empty table with columns 'Library', 'Version', 'Date', and 'Description'. The 'Destination' section has 'Use MODELICAPATH' selected, with the path 'E:/Program Files/Dymola 2025x Refresh 1 Beta 2/Modelica' shown. The 'Options' section has 'Add to Libraries menu' checked. The 'Install' button is visible at the bottom.

Library	Version	Date	Description
---------	---------	------	-------------

² For more information about `libraryinfo.mos`, see next section.

³ Any existing `libraryinfo.mos` is copied without validation. If it is incorrect, the library may not show up in the **File > Libraries** menu.

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

Now, to do the download and installation from GitHub, do the following:

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

Source

☐ Local file ?

☒ GitHub URL

Library Next

Tag

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

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

Source

☐ Local file ?

☒ GitHub URL

Library Next

Tag

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

Source

☐ Local file ?

☒ GitHub URL

Library Next

Tag

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

Library Management

Libraries

Modelica Path

Install

Source

☐ Local file
☒ GitHub URL

Library

github.com/RWTH-EBC/AixLib

Tag

v2.1.0

Next

Selection

<input checked="" type="checkbox"/>	Library	Version	Date	Description
<input checked="" type="checkbox"/>	AixLib	0.5.0		

Destination

☒ Use MODELICAPATH

E:/Program Files/Dymola 2025x Refresh 1 Beta 2/Modelica

☐ Working directory
☐ Custom path

Options

☐ Replace existing
☒ Add to Libraries menu

Install

OK

Cancel

Select what you want to download from the Selection group, and also specify your selections for the **Destination** and **Options** group (see previous section).

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

More about libraries and the building of menus of libraries and demos

General information

Dymola can automatically recognize different libraries in order to e.g. build the **File > Libraries** and **File > Demos** menus. It is easy to add new libraries and to add new versions of existing libraries.

All information about a library exists in a local file, so it is possible to just “unzip” a subdirectory containing a package, and it will automatically be recognized by Dymola.

No update of a common file is needed, hence no need for special installation scripts. It also makes it easy to delete libraries, just delete the directory.

Using library information

Associated with each package is a Modelica script that is automatically located by Dymola at program start. This script can contain a set of commands that describes the package and builds e.g. **File > Libraries**.

The script is called `libraryinfo.mos` and can be stored in three alternative locations. Assuming the package XYZ is stored as `dymola/Modelica/Library/XYZ`, the script can be stored as any of:

- `dymola/Modelica/Library/XYZ/libraryinfo.mos`
- `dymola/Modelica/Library/XYZ/Scripts/libraryinfo.mos`
- `dymola/Modelica/Library/XYZ/Resources/Scripts/Dymola/libraryinfo.mos`

(The above is also the order of searching for this file.)

Note that you must yourself create the scripts for your own libraries if you e.g. want to add them in menus. It is wise to look both below, and at the already present `libraryinfo.mos` files for libraries already in the **File > Libraries** menu when doing this.

Building menus

There is currently a low-level script command to build libraries and demos menus, e.g.:

For a simple example of a library (please also compare the section “Customizing the File > Libraries menu” above):

```
LibraryInfoMenuCommand(  
  category="libraries",  
  text="Cooling",  
  reference="Cooling",  
  version="1.2",  
  isModel=true,  
  description="Cooling Library (1.2) by Dassault Systèmes",  
  ModelicaVersion=">=3.2.2"  
  pos=525);
```

For a simple example of a demo:

```
LibraryInfoMenuCommand(category="demos",  
  text="My Demo",  
  reference="MyDemo.MyDemoModel",  
  isModel=true,  
  description="My demo, basic",  
  pos=9999);
```

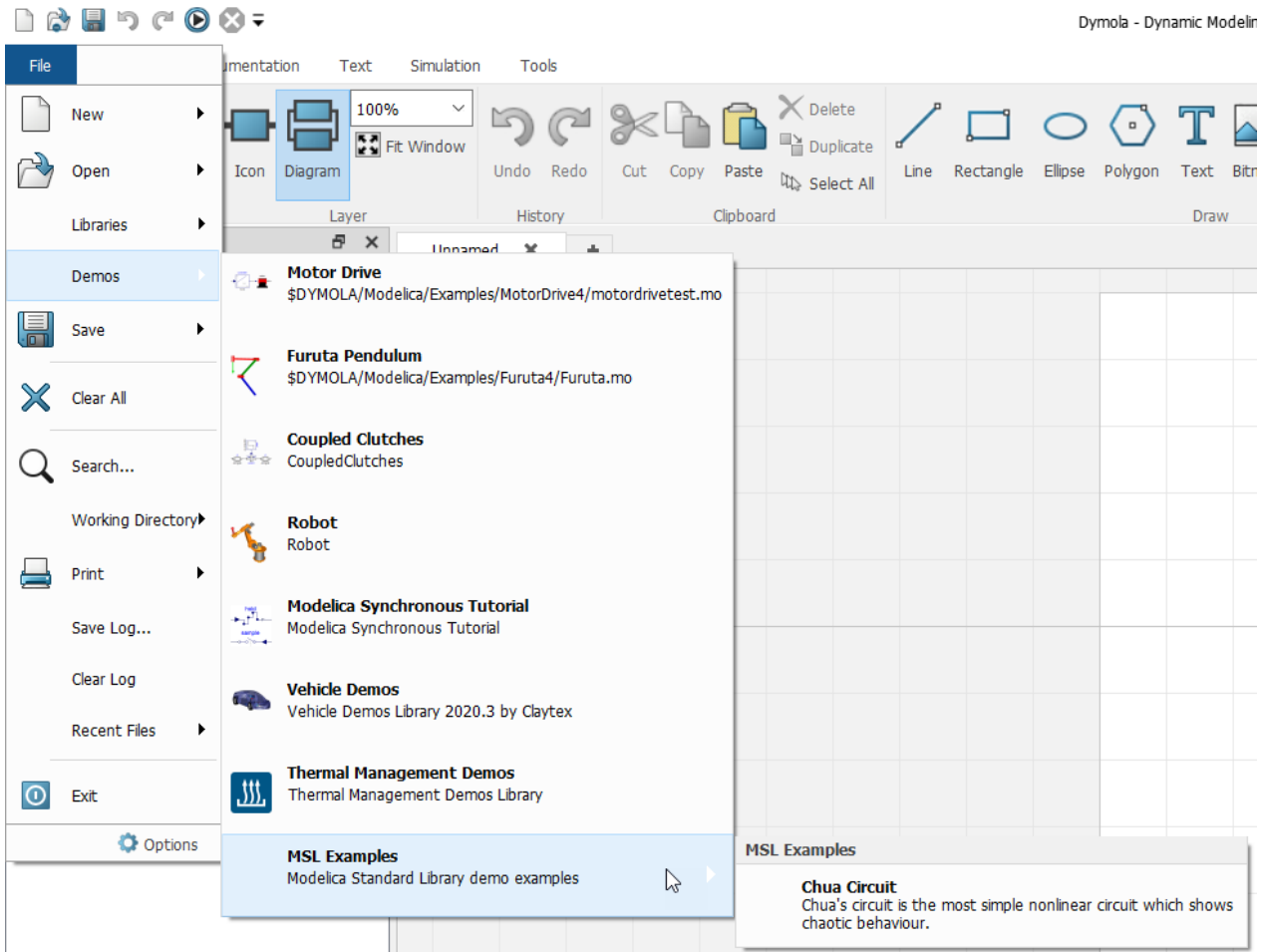
A demo with sub-category of a menu could be:

```

LibraryInfoMenuCommand(category="demos",
    subCategory={"MSL Examples", "Modelica Standard Library demo
examples"},
    text="Chua Circuit",
    reference="Modelica.Electrical.Analog.Examples.ChuaCircuit",
    isModel=true,
    description="Chua's circuit is the most simple nonlinear circuit which
    shows chaotic behaviour.",
    pos=9999);

```

The result of the command **File > Demos** when using this function in libraryinfo.mos is:



The attributes have the following meaning (the list contains more attributes than the examples above):

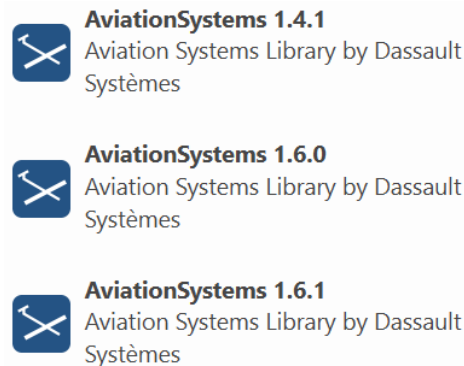
Attribute	Meaning
category	Primary menu category (“libraries”, “demos”, or “persistent”)
subCategory[:]	Optional sub-category of the menu entry, a vector of zero to two strings. The first string is a sub-menu name; the second string is a longer description text. For example, subCategory = {"MSL Examples", "Examples from the Modelica Standard Library"}
text	Text shown in menu
reference	Model path or command string
isModel	If true, the text is a model path, otherwise a command.
description	Longer description, for example shown in status bar
version	Version of library (does not apply to demos). See Important below.
ModelicaVersion	Required version of Modelica Standard Library, e.g. “> 2”. The value “2” means “>=2”.
pos	Position in the menu. The menu alternatives are sorted according to this attribute, lowest numbered at the top.
requiredFlags[:]	If flags are specified, they have to be true for the menu to be active. If a flag is preceded by ! it means that the flag must be false for the menu to be active.
iconName	For specifying an external icon, if needed. Please see next section.

To handle different libraries and groups of libraries, and to make sure Dymola has a consistent ordering of Libraries, Dassault Systèmes allocates ranges of positions to different library vendors. For example, 0 to 999 could be reserved for Dassault Systèmes, 1000 to 1999 for DLR, etc.

Important: For libraries, the version must be specified in the libraryinfo.mos file, and in the corresponding library file (package.mo). The latter is done by the version annotation. The version specified must be the same in both files.

Handling of multiple versions of a library

If there are multiple versions of a library, resulting in duplicated menu entries in the **File > Libraries** menu, Dymola will attempt to disambiguate the versions by appending the version number. For example:



Note. The version number is appended to *all* libraries, not only the ones with multiple versions.

This functionality can be disabled by setting the flag

```
Advanced.File.DisambiguateLibraries = false
```

Thereby restoring the behavior of Dymola 2024x Refresh 1 and older versions. (The flag is by default `true`.)

Automatically avoiding double Libraries menu entries

Library developers may sometimes find duplicate entries in the **File > Libraries** menu. The reason is typically that the `MODELICAPATH` system variable contain multiple library sources. This can be checked (and fixed) using **Tools > Library Management > Modelica Path**. (For more information about this, see “The environment variable `MODELICAPATH` and how to manage it” starting on page 38.)

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

- Same library name.
- Same description.
- Same version number.

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

Support for external icons for the menus **File > Libraries** and **File > Demos**

To support external icons in the **File > Libraries** and **File > Demos** menus, there is an input parameter `iconName` of type `String` available in `LibraryInfoMenuCommand` (see table above). The default value is `"libraryicon.png"` which means that the easiest solution to add an external icon is to create an icon with that name in the same folder as `libraryinfo.mos`. Notes:

- Other file formats are also supported, for example, `svg`.
- A relative path to the icon can also be added if the icon is located in another folder than `libraryinfo.mos`.

- Oversized icons are scaled to fit in the menus.

Adding a menu separator

It is possible to add a separator (horizontal line) in the menus. For example,

```
LibraryInfoMenuSeparator(  
    category="libraries",  
    pos=101);
```

The arguments have the meaning described in the table above.

Option to use a custom setup file

It is possible to run Dymola with a custom setup file. This enables having different setups for different projects; for example:

- Preloading project-specific libraries
- Maintaining conflicting and alternative library versions by having different Modelica paths
- Setup tailored for simulation, customer-specific presentations, or with GUI layout optimized for projectors.
- Using different compiler settings for different customers

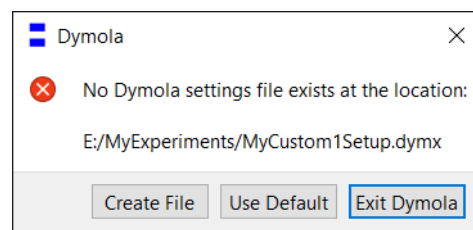
(For more information about the content and handling of the setup file, see the manual “*Dymola User Manual 1B: Developing and Simulating a Model*”, index entry “*setup file*”.)

You can in fact copy the whole installation to another location, and use a custom setup file when starting the copy, using the command line as below. This copy of Dymola will be totally independent from the original, except for the licensing.

To specify the path to the custom setup.dymx file, the command line argument `-setup` is used, for example:

```
Dymola.exe -setup "E:\MyExperiments\MyCustom1Setup.dymx"
```

If this cannot be found, an error message appears:



You have the following alternatives:

- **Create File** tries to create an empty `setup.dymx` at the specified location.
- **Use Default** uses the default setup file, on Windows typically:

```
C:\Users\<user>\AppData\Roaming\DassaultSystemes\Dymola\
2025x Refresh 1\setup.dymx
```

- **Exit Dymola** exits Dymola. If Dymola is started with any of the command line arguments `-nowindow` or `-externalinterface` then Dymola is exited with an error code.

Loading a package with user-defined menus and toolbars that should not be deleted by File > Clear All

A package with user-defined menus and toolbars where the menus and toolbars should not be deleted by the command **File > Clear All** can be automatically loaded by a `libraryinfo.mos` file with `category="persistent"` (see also above section).

```
LibraryInfoMenuCommand(
  category="persistent",
  reference="<Class to preload>",
  text="dummy")
```

For more information about creating such menus, see the manual “*Dymola User Manual Volume 2C: Advanced Concepts*”, chapter “User-defined GUI”, section “Extendable user interface – menus, toolbars and favorites”.

License expiration settings

The default behavior of Dymola is:

- To start to warn the user that a license is to expire, 30 days before expiration.
- To continue in trial mode if a license has expired or is faulty.

The behavior can be configured with a command line argument

Consider a user wanting to have the first warning 5 days before the license is expiring, and wanting to terminate Dymola if the license is not found or invalid. Assuming a 64-bit Dymola with default location on a MS Windows computer, Dymola could be started with the following command line using the Command Prompt in Windows:

```
"C:\Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe" /days -5
```

The value (5) controls how many days that should be left to expiration when warning, and the minus before the value is added if Dymola should terminate if the license is not found or invalid.

“-” can be used instead of “/”; the example above will then be:

```
"C:\Program Files\Dymola 2025x Refresh 1\bin64\Dymola.exe" -days -5
```

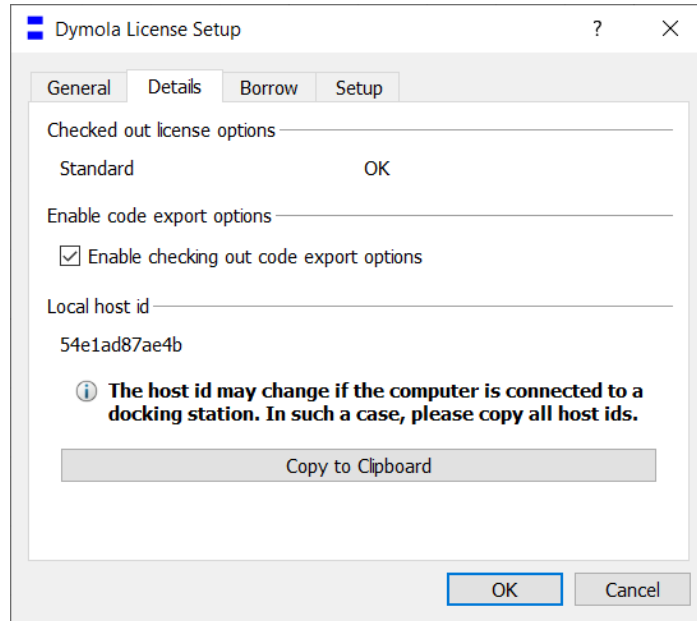
1 day is the minimum start time for warning of license expiration; the warning cannot be completely disabled.

Preventing checking out license options from a license server

It is possible to prevent Dymola from checking out certain license options from the license server, if a sharable license is used. (It is also possible using a node-locked license, e.g. if a user wants to test if a certain model still works without a certain library.)

Using the command **Tools > License Setup** and then looking in the **Details** tab reveals license options currently checked out.

Example of license options checked out.



If the user wants to prevent some option from being checked out, it can be done in a number of ways:

- For code export options, you can untick **Enable checking out code export options** in the menu above. Doing this, you will prevent both BinaryModelExport and SourceCodeGeneration to be checked out. The setting will be remembered between Dymola sessions. The setting can also be accomplished using the flag `Advanced.EnableCodeExport=false;`

For more information concerning code export, please see the manual “*Dymola User Manual 2B: Simulation Interfaces and Export*”, chapter “Simulation Environments”.

- By modifying the shortcut to Dymola.
- By starting Dymola with a certain command line option using the Command Prompt in Windows.

Modifying the shortcut will result in prevention of check out of specified options each time Dymola is started using that shortcut, as. Starting Dymola using a modified command from

the command prompt in Windows will only result in prevention of check out of specified options in that session.

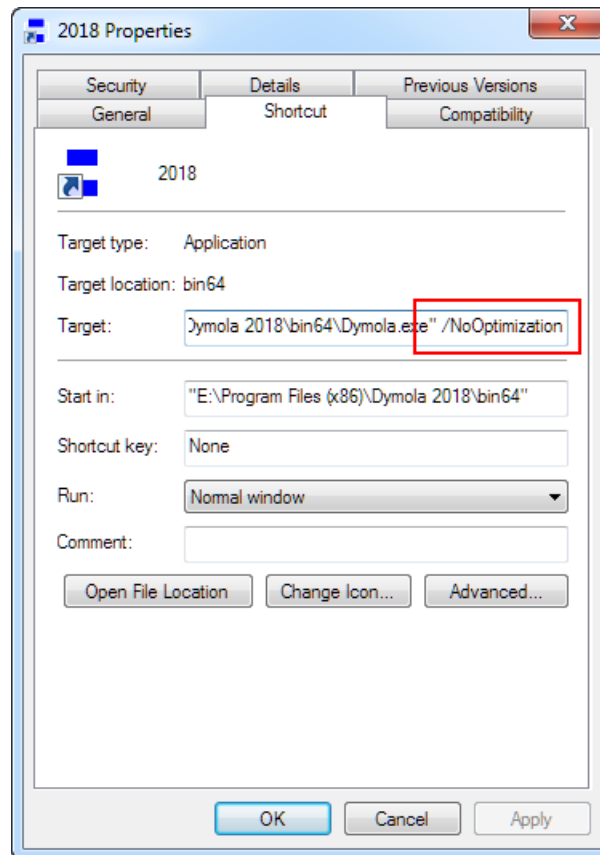
Since the command for prevention of checking out license options is generic, it is very important to use the correct name of the option, including correct use of capitals. The best way is to look at the checked out options using the command above, and mark and copy the name (for example, Optimization) of the option that should not be checked out, to insert that name when using any command.

Modification of the shortcut to Dymola

If a new shortcut is needed, please look at the section “Creating shortcuts to Dymola” above.

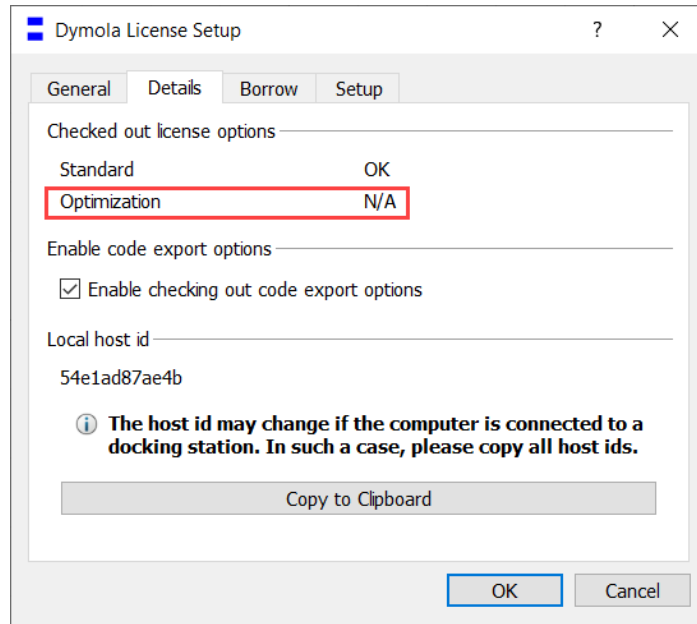
To modify the shortcut to prevent checking out a certain option, right-click the shortcut and modify the **Start in:** by adding <space>/No<optionname> in the end of the command. If the option Optimization should not be checked out, the shortcut should be modified like in the figure below. (We prevent check-out of the Optimization option.)

Modified shortcut.



Closing Dymola and starting it again, the following information will be found in the license tab:

Prevention of checking out a license option.



Now the `Optimization` license option will not be possible to check out. As long as the shortcut is not modified, `Optimization` will not be possible to check out from Dymola started by that shortcut.

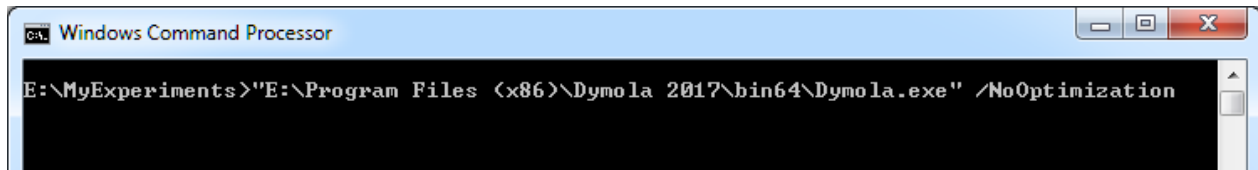
To enable check out of `Optimization`, Dymola must be closed and then restarted using a shortcut without the command line option for `Optimization`.

More than one option can be prevented from check out – just add more strings like the one used. Do not forget the space.

Starting Dymola using a modified command in Command Prompt of Windows

A Windows command prompt can be activated using **Help > All Programs > Accessories > Command Prompt** in Windows.

To start one session in Dymola where the license option `Optimization` cannot be checked out like in the example above, the command in the command prompt will look like:



Teleworking (remote login) in Dymola

Teleworking (to work from home by making a remote login to work) is allowed in the Dymola license conditions. In earlier versions of Dymola, this required a sharable (server)

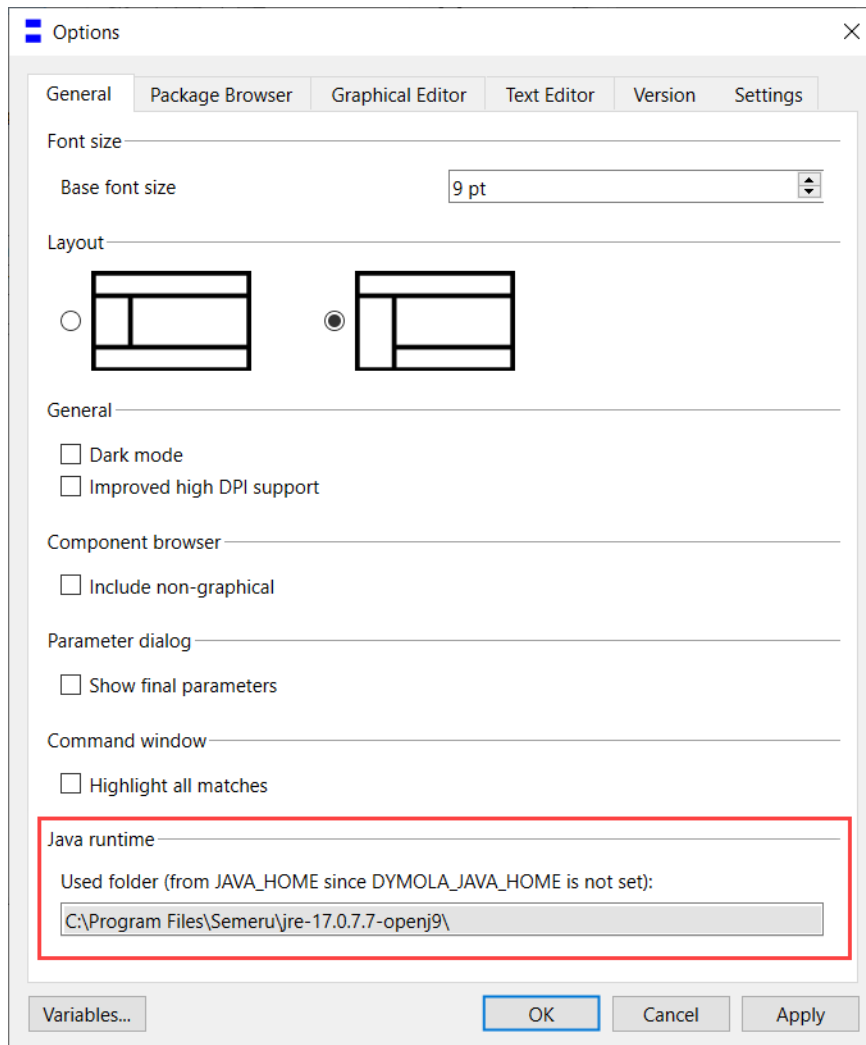
license, but, from Dymola 2024x, new FLEXnet license keys support teleworking also for nodelocked licenses. The following restrictions apply:

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

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

Java Runtime Environment location

You can see the used Java Runtime Environment location by the command **Tools > Options**, the **General** tab:



You can set the Java Runtime Environment folder to use in two ways:

- The environment variable **DYMOLA_JAVA_HOME**
- The environment variable **JAVA_HOME**

Having two environment variables allow you to have more than one Java Runtime Environment installed, and use them for different applications.

For the Dymola related applications “Design with Dymola” and “eFMI”, these applications first look if the first environment variable above is set. If not, they look for the other variable. This means that this part of the dialog can look like any of the two images below:

Java runtime

Used folder (from DYMOLA_JAVA_HOME):

C:\Program Files\Semeru\jdk-17.0.7.7-openj9

Java runtime

Used folder (from JAVA_HOME since DYMOLA_JAVA_HOME is not set):

C:\Program Files\Semeru\jdk-8.0.402.6-openj9\

If none of the enviroment variables are set, the location field above is left empty.

If you are to specify the Java Runtime Environment location, use the first choice DYMOLA_JAVA_HOME since this is specific for Dymola and related applications, the other enviroment variable is a more general one that might be used by other applications (with other version demands) as well.

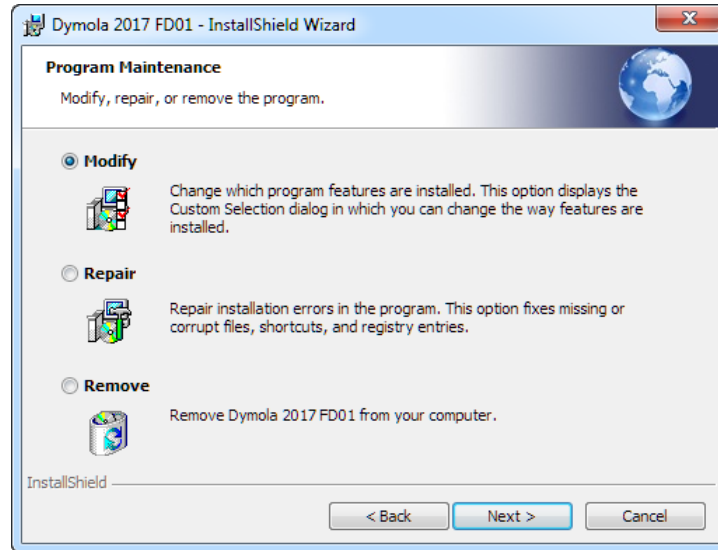
1.1.6 Changing the setup of Dymola

Under Windows, you can change the setup of Dymola, for example to install additional libraries. Click on the **Start** button in the Taskbar, select **Control Panel** and open **Programs and Features**. Select the relevant version of Dymola and click on the **Change** button.



Selecting **Next>** will display

Changing Dymola setup.



To change the setup, choose **Modify**. The rest of the procedure will be the same as when installing Dymola from scratch. Please see previous sections. To restore files in the Dymola distribution that have been deleted by mistake, choose **Repair**. **Remove** will remove the installation.

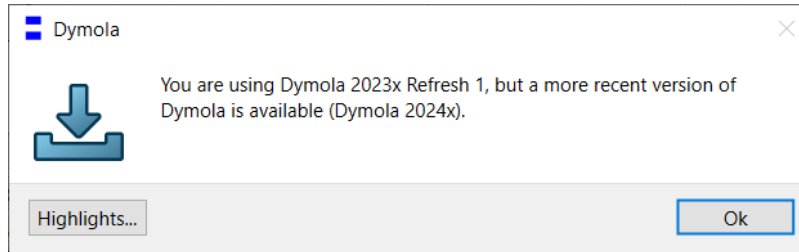
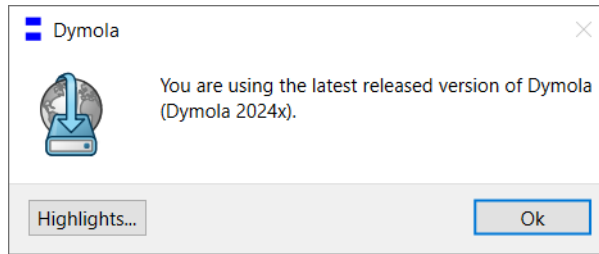
1.1.7 Removing Dymola

Please see previous section. Do not delete or rename the Dymola directory. Microsoft Windows Installer keeps track of all installed directories and will try to repair if altered. The installation will, by default, use a directory name that reflects the version of Dymola, but this can of course be changed during setup.

Note that files that you have created in the Dymola distribution directory, for example by running the demo examples, are not deleted when removing Dymola. The remaining files and directories (if any) may be deleted through the Explorer.

1.1.8 Checking for updates

To check for possible updates of Dymola, use the command **Tools > About Dymola**, and then click the button **Check for Updates**. Some possible outcomes:



To see the highlights of the latest release, click the **Highlights...** button.

1.1.9 Installing updates

Updated versions of Dymola either are distributed on CD, or can be downloaded from a location provided by your sales channel.

Multiple versions of Dymola can be installed, but you cannot install into an existing Dymola directory. Configuration settings and the license file are shared by all installed versions of Dymola.

1.2 Installation on Linux

This section refers only to the Linux version of Dymola.

This section covers Linux-specific parts of the installation. For general items, e.g. how to handle the Dymola installation wizard; please see corresponding section on Windows installation, in particular section “Installing the Dymola license file” starting on page 22.

The default directory for installation on Linux is `/opt/dymola-<version>-x86_64`. As an example, the default directory for installation of Dymola 2025x Refresh 1 on Linux is `/opt/dymola-2025xRefresh1-x86_64` (the package manager on the target system however typically allows choosing another default location).

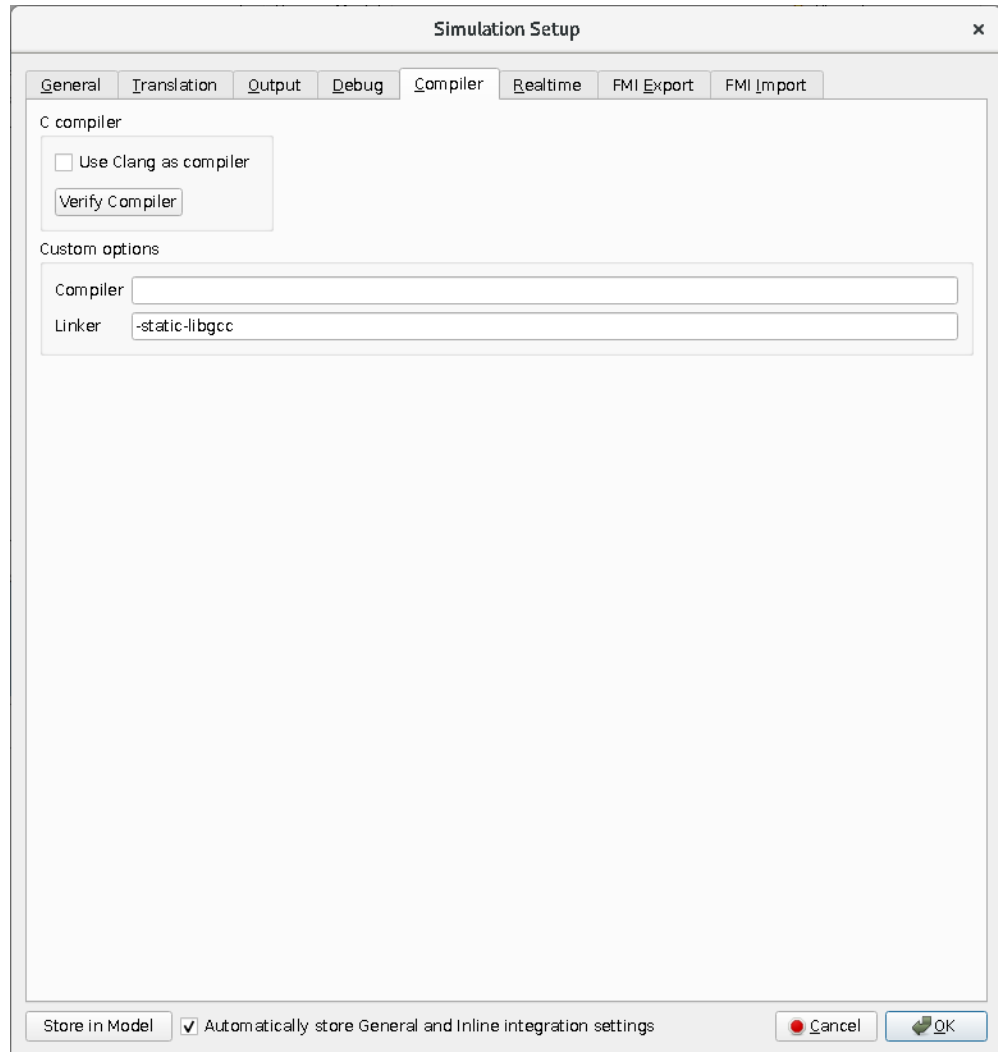
Dymola 2025x Refresh 1 runs on Red Hat Enterprise Linux (RHEL) version 8.6, 64-bit, with gcc version 11.2.1, and compatible systems. (For more information about supported platforms, do the following:

- Go to <https://doc.qt.io/>
- Select the relevant version of Qt, for Dymola 2025x Refresh 1 it is 6.8.1. For earlier versions of Dymola, to find the relevant Qt version, see the corresponding Dymola Release Notes.
- Select Supported platforms)

Any later version of gcc is typically compatible.

In addition to gcc, the model C code can also be compiled by Clang. To be able to select Clang, it must be installed. See section “Compilation of model code” on page 64.

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



The **Verify Compiler** dialog can look like:



**Only 64-bit simulation
on Linux.**

Dymola 2025x Refresh 1 is supported as a 64-bit application on Linux. Corresponding support for 64-bit export and import of FMUs is included. **Note** that support for 32-bit simulation on Linux (including 32-bit export and import of FMUs) is discontinued from Dymola 2025x.

The default setting is to use 64-bit Dymola when translating models.

Notes

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

Please also 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 following libraries are currently not supported on Linux.

- Process Modeling Library
- Thermodynamics Connector Library
- UserInteraction Library

More Linux-specific notes are available using the command

```
man dymola
```

1.2.1 Installing Dymola

Dymola for Linux is distributed as an RPM package. The package is installed using the command

```
# rpm -i name-of-distribution.rpm
```

Optional libraries are installed through separate RPM files.

For installation on e.g. Debian or Kubuntu systems conversion to the deb format is required if you want to use their default package manager. One way is using the `alien` command:

```
# alien -cki name-of-distribution.rpm
```

Note however that the command `rpm` can often be used on these systems as well, although you might get warnings about “using rpm directly”. Except for updating the Dymola startup script and the manual, the Dymola installation does not affect the system directories (such as `/usr/lib64`) but writes only to its own installation directory. Hence, using the command `rpm` directly is harmless with respect to the system.

Setup and environment variables

The shell script `/usr/local/bin/dymola-<version>-x86_64` (see above concerning “-<version>”) contains commands to set environment variables before starting Dymola, but will need editing if Dymola is installed in a non-standard location; then the following environment variables must be defined in order to run Dymola:

DYMOLA Directory root of the distribution (`/opt/dymola-<version>-x86_64`).

DYMOLAPATH Search path for additional Dymola libraries and the license file. The directories of the path may be separated by blanks or colon. **DYMOLAPATH** is optional if the license file is in `$DYMOLA/insert`.

MODELICAPATH Search path for libraries. Concerning the use of **MODELICAPATH**, please see section “Managing library locations” starting on page 38.

(Dymola defines an environment variable **DYMOLAWORK** which value is the Dymola startup directory. For more info, see section “Location of startup directory” on page 31. Note that the startup directory/current working directory cannot be an UNC path in Linux.)

1.2.2 Additional setup

Subjects in the corresponding section on Windows are not applicable unless explicitly referenced from here.

Compilation of model code

Dymola produces C code which must be compiled in order to generate a simulation model. On Linux systems we by default rely on an ANSI/ISO C compiler already installed on the computer. You can however also select to use Clang as compiler.

On Linux systems the compilation of the generated C code is performed by a shell script, `/opt/dymola-<version>-x86_64/bin/dsbuild.sh` (see above concerning “-<version>”). If necessary this script can be modified to select Clang as compiler, provide special options to the compiler, add application-specific libraries etc. **Notes:**

- To be able to select Clang, you must install Clang. You can do this in Red Hat Enterprise Linux (RHEL) with the command `sudo yum install clang` – on Ubuntu you can use the command `sudo apt install clang`.
- You can perform some of the settings above in the simulation setup, reached by the command **Simulation > Setup**, the **Compiler** tab – see image above.
- Simulation performance can be improved by tuning the compilations options in the shell script, however note that the compiler time may increase significantly by doing so.

Dymola supports external C libraries on Linux. Classes which contain “Library” annotations to link with external libraries in C are supported.

Simulation from the command line

The simulator executable `dymosim` can be executed from the shell.

If you move the code to another computer with different Dymola installation path, you first need to set the environment variable LD_LIBRARY_PATH in the shell. E. g. for a 64-bit simulation:

```
# export LD_LIBRARY_PATH=/opt/dymola-<version>-x86_64/bin/lib64
```

For Dymola 2025x Refresh 1 this would be:

```
# export LD_LIBRARY_PATH=/opt/dymola-2025xRefresh1-x86_64/bin/lib64
```

Preventing splash screen at startup

Please see corresponding section for Windows installation.

Working with a Modelica version that is newer than the one in the distribution

Please see corresponding section for Windows installation.

Customizing the File > Libraries menu

Please see corresponding section for Windows installation.

Managing library locations

Please see corresponding section for Windows installation.

On-demand downloading and installation of libraries

Please see corresponding section for Windows installation.

More about libraries and the building of menus of libraries and demos

Please see corresponding section for Windows installation.

Option to use a custom setup file

For Linux the feature is the same as in Windows (see that section), but Linux have other paths; for example, the default path for the default setup file in Linux is, under the user's home directory:

```
.dassaultsystemes/Dymola/2025xRefresh1/setup.dymx
```

Loading a package with user-defined menus and toolbars that should not be deleted by File > Clear All

Please see corresponding section for Windows installation.

License expiration settings

Please see corresponding section for Windows installation, but note the paths are different in Linux. Also, an option is specified with “-“ and not “/”.

Preventing checking out license options from a license server

In the corresponding section on Windows the alternative of starting Dymola using a modified command is applicable also for Linux (with relevant changes for Linux). Please see page 55.

Teleworking (remote login) in Dymola

Teleworking (to work from home by making a remote login to work) is allowed in the Dymola license conditions. Please see page 55.

1.2.3 Removing Dymola

Remove the Dymola distribution by using the `rpm -u` command.

1.3 Dymola License Servers on Windows

This section refers only to the Windows version of Dymola.

There are two possible license servers for Dymola on Windows. The default one is FLEXnet Publisher license server. As an alternative, the Dassault Systèmes License Server (DSLS) can be used.

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

1.3.1 FLEXnet Publisher license server

Background

These are instructions for manually installing a FLEXnet Publisher license server for Dymola on Windows. They only apply to users with a sharable license. For non-sharable licenses (the common case), installation of the license file is automatic.

All files needed to set up and run a Dymola license server on Windows, except the license file, are available in the Dymola distribution, in Program Files\Dymola 2025x Refresh 1\bin64.

Dymola is installed on all machines that will run the software. On the designated machine, the license server is then installed as described below.

The license server consists of two processes:

- The vendor daemon (called `dynasim.exe`) dispenses licenses for the requested features of Dymola (the ability to run Dymola and various options). This program is specific for Dymola.
- The license server manager (called `lmgrd.exe`) sends requests from application programs to the right vendor daemon on the right machine. The same license server manager can be used by all applications from all vendors, as this license server manager processes no requests on its own, but forwards these requests to the right vendor daemon. (For use of the newer web based `lmadmin`, see “Using the web based license server manager `lmadmin` instead of `lmgrd`” on page 71.)

If you are already running an application that uses FLEXnet Publisher, you most likely already have a running license server manager. In this case, only the vendor daemon (`dynasim.exe`) is required.

Flexera Software recommends that you use the latest version of the FLEXnet Publisher `lmgrd.exe` at all times as it includes bug fixes, enhancements, and assures the greatest level of compatibility with all of your FLEXnet Publisher licensed applications. Flexera Software guarantees that it will work correctly with all earlier versions of FLEXnet Publisher.

Old license server managers cannot be used!

Dymola requires support of FLEXnet Publisher version 11.16.2.1. A recent version of `lmgrd.exe` is part of the Dymola distribution.

Installing the license server

This section describes the simple case where we assume there are no other FLEXnet Publisher license server managers. We also assume that the Dymola program itself should not be installed on the server. For updating of a present license server, see separate section below.

To purchase a server license, the relevant host id of the computer where the license server should run must be supplied to your Dymola distributor before purchasing the license. The license that you will receive will contain this information. To find out the host id of that computer, the utility program `hostid.exe` can be used. Please see section “Obtaining a host id” on page 83 for more information.

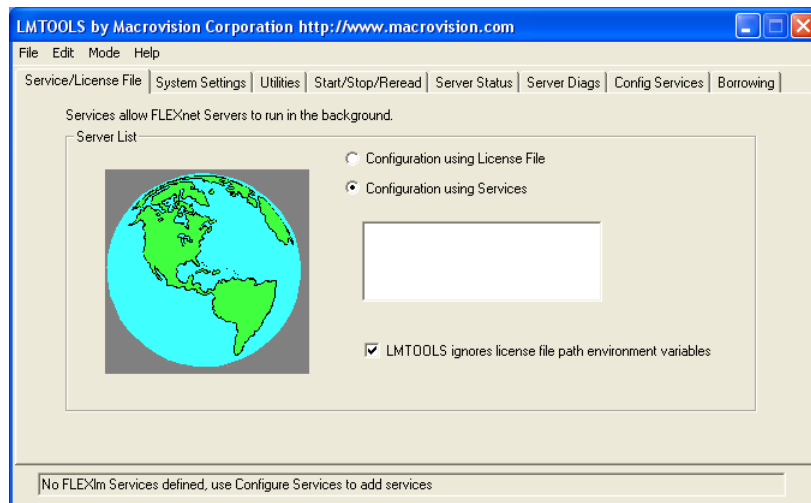
1. Before installation of the license server, the Dymola license file (*filename.lic*) may have to be updated with the actual name (or IP-number) of the server, if the license file contains a line identifying the server:

```
SERVER server.name.here 000102DE37CD
```

The part *server.name.here* must be changed to the name of the actual server before installing the license file. It should be noted that the last part (the hostid) cannot be edited by the user.

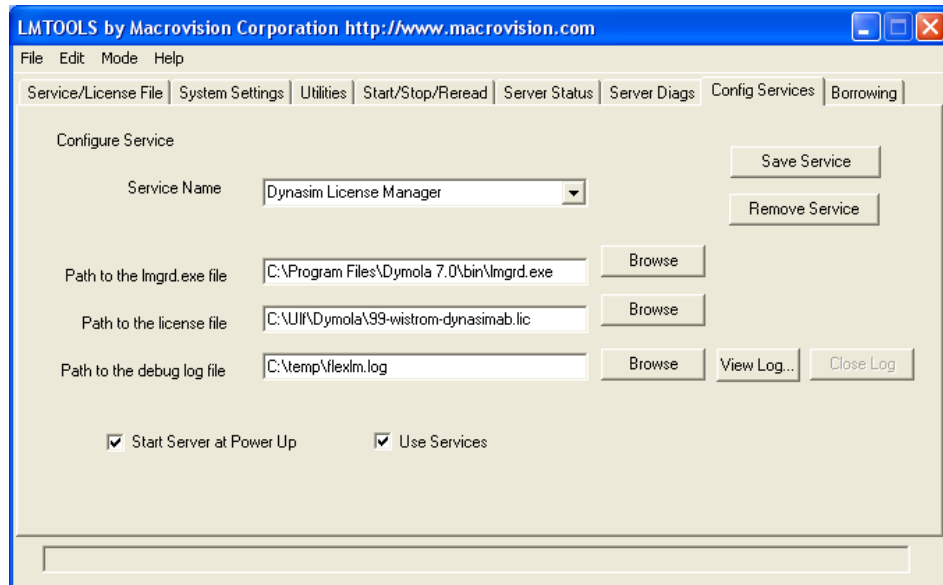
2. Install *only* the Dymola software component **License server** (see beginning of this chapter). A folder will be created containing all needed files, default `C:\Program Files\Dymola 2025x Refresh 1\bin64`.
3. Start the utility program `lmtools.exe` (one of the above files).
4. In the **Service/License File** tab:
 - a. Select the radio button **Configuration using Services**.
 - b. Activate **LMTTOOLS ignores license file path environment variables**.

License server setup.



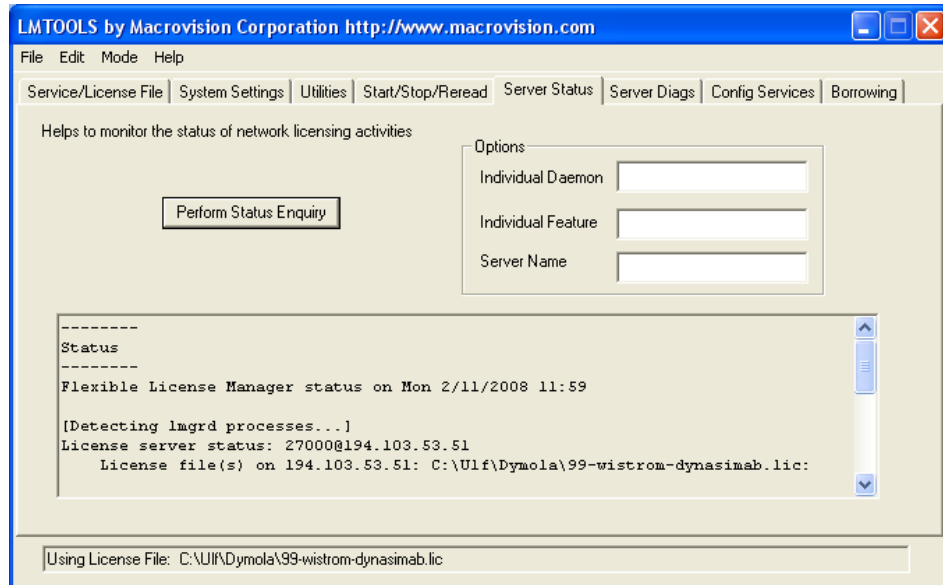
5. In the **Config Services** tab (please see figure on next page):
 - a. Enter a new service name, e.g. “Dynasim License Server”.
 - b. Enter the path to the license server manager, Dymola 2025x Refresh 1\bin64\lmgrd.exe.
 - c. Enter the path to your server license file.
 - d. Enter the path to a debug log file (anywhere you want).
 - e. Enable **Use Services** and then **Start Server at Power Up**.
 - f. Click on **Save Service**. Click on **Yes** to confirm.

Configuration of the license server.



6. In the **Start/Stop/Reread** tab:
 - a. Select the Dynasim license server.
 - b. Click on **Start Server**.
7. In the **Server Status** tab:
 - a. Click on **Perform Server Enquiry** and check the output of the log window. You should see lines identifying the server processes and what features are available.

**Checking the operation
of the license server.**



- b. Also check the log file to verify that the server has started and that Dymola features can be checked out. The following is an example of the FLEXnet Publisher logfile:

```
12:30:48 (lmgrd) pid 2728
12:30:48 (lmgrd) Detecting other license server manager (lmgrd) processes...
12:30:48 (lmgrd) Done rereading
12:30:48 (lmgrd) FLEXnet Licensing (v11.4.100.0 build 50818 i86_n3) started
on 194.103.53.51 (IBM PC) (2/11/2008)
12:30:48 (lmgrd) Copyright (c) 1988-2007 Macrovision Europe Ltd. and/or
Macrovision Corporation. All Rights Reserved.
12:30:48 (lmgrd) US Patents 5,390,297 and 5,671,412.
12:30:48 (lmgrd) World Wide Web: http://www.macrovision.com
12:30:48 (lmgrd) License file(s): C:\Ulf\Dymola\99-wistrom-dynasimab2.lic
12:30:48 (lmgrd) lmgrd tcp-port 27000
12:30:48 (lmgrd) Starting vendor daemons ...
12:30:48 (lmgrd) Started dynasim (pid 4180)
12:30:48 (dynasim) FLEXnet Licensing version v11.4.100.0 build 50818 i86_n3
12:30:48 (dynasim) Server started on 194.103.53.51 for: DymolaStandard
12:30:48 (dynasim) DymolaAnimation DymolaModelCalibration
DymolaModelManagement
12:30:48 (dynasim) DymolaOptimization DymolaRealtime DymolaSimulink
12:30:48 (dynasim) DymolaFlexibleBodiesLib DymolaHydraulicsLib
DymolaPowertrainLib
12:30:48 (dynasim) DymolaSmartElectricDrivesLib
12:30:48 (dynasim) EXTERNAL FILTERS are OFF
12:30:48 (lmgrd) dynasim using TCP-port 2606
12:30:56 (dynasim) TCP_NODELAY NOT enabled
10:39:20 (lmgrd) Detecting other lmgrd processes...
10:39:35 (lmgrd) FLEXlm (v7.2c) started on x.x.x.x (3/27/2001)
```

```

10:39:35 (lmgrd) FLEXlm Copyright 1988-2000, Globetrotter Software
10:39:35 (lmgrd) US Patents 5,390,297 and 5,671,412.
10:39:35 (lmgrd) World Wide Web: http://www.globetrotter.com
10:39:35 (lmgrd) License file(s): C:\DAG\dymola.lic
10:39:35 (lmgrd) lmgrd tcp-port 27000
10:39:35 (lmgrd) Starting vendor daemons ...
10:39:35 (lmgrd) Started dynasim (pid 124)
10:39:36 (dynasim) Server started on x.x.x.x for:DymolaStandard
10:39:36 (dynasim) DymolaSampledLib DymolaLiveObjects DymolaRealtime
10:39:36 (dynasim) DymolaSimulink DymolaAnimation DymolaSupport
10:39:36 (lmgrd) dynasim using TCP-port 1042

```

The license server should now be correctly configured. Please start Dymola to verify correct operation. The FLEXnet Publisher logfile (see above) should contain additional lines showing what features were checked out. You can also do **Perform Status Enquiry** to check how many licenses are currently checked out.

Note. The license server by default uses the ports 27000-27009. They can be configured if needed, e.g. if there are issues with firewalls.

Using the web based license server manager lmadmin instead of lmgrd

From Dymola 2021 you can use the web based license server manager `lmadmin` instead of the older command line based `lmgrd`, for both Windows and Linux.

Note. The Flexera FlexNet license server manager `lmadmin` is not included in the Dymola distribution; you must download this software yourself.

Some short notes about the handling:

- start: run `lmadmin/lmadmin`
- access: <http://localhost:8090>
- stop: Administration – Server Configuration – Stop Server
- log on first time: user: “admin”, passwd: “admin”

For more information, please see the documentation for Flexera Software LCC FlexNet Publisher Licensing Toolkit 11.16.

Installation of license when a license server already exists on the target machine

Limiting ourselves to the case of a single server, there are two strategies handling multiple vendor daemons:

- Use a single license server manager
- Use one license server manager per vendor daemon

How you should add a vendor daemon for the first case depends on the license server manager used:

- If using `lmgrd`, a restart is required, with the new license file added to the `-c` option.
- If using `lmadmin`, the new vendor daemon can be added via the GUI.

For the second case above (using one license server manager per vendor daemon), you must deal with the clashing of listening ports for the multiple license server managers. However, the license server manager will alert, during startup, such clashes. Hence, a simple strategy would be to first try the default port and upon failure explicitly try ports in the default range (27000-27009) or some other range, depending on the IT department port blocking policies.

For more information, please see the documentation for FlexNet Publisher 2016, in particular the section “Managing Licenses from Multiple Software Publishers” in fnp_LicAdmin.pdf.

Updating the license server with new license file

When receiving a new Dymola license file (filename.lic), first perform step 1 in the first section “Installing the license server” above.

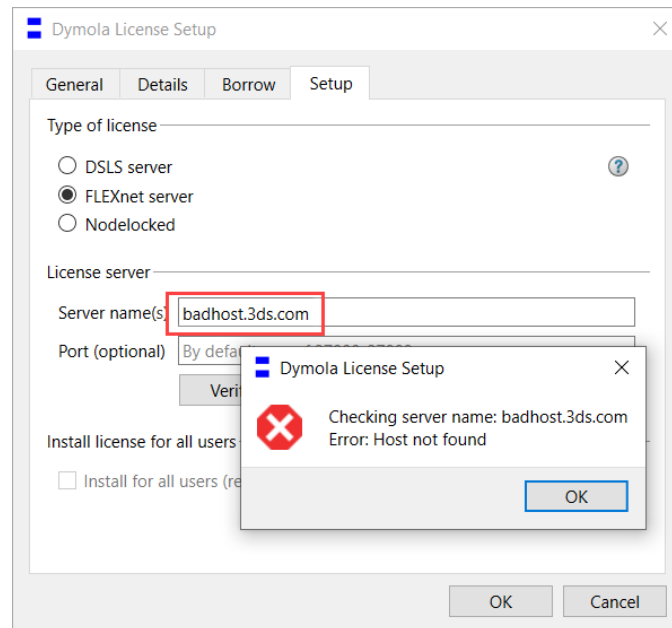
Now you can replace the old license file with the new (it might be good to temporarily keep the old one by adding e.g. _previous to the name of the old file, in case you must go back to that one).

Now you can start the utility program lmttools.exe. In the Start/Stop/Reread tab, click the button **ReRead License File**.

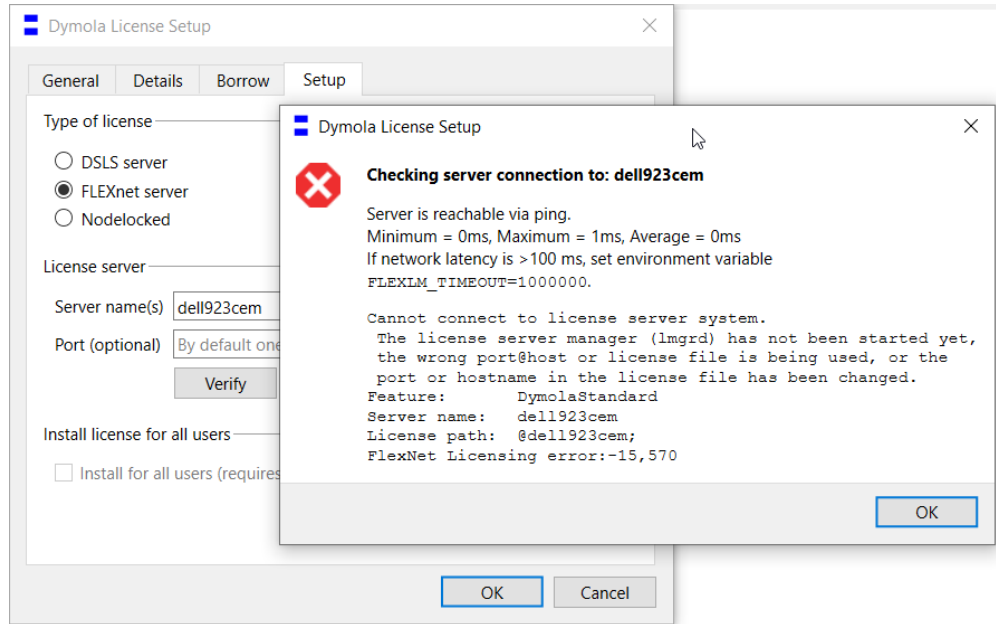
The license server is now updated.

Testing the license server at startup

When trying to connect to a DSLS or FLEXnet license server host, a basic DNS lookup is performed. If the host cannot be found, the following message appears:



If the host is found, but Dymola fails to contact the license server, a ping test is performed to check if the network is alive, and if timing problems occur. In addition, a test checkout of Dymola Standard license is requested from this server. An example with successful ping test but failed Dymola license checkout is:

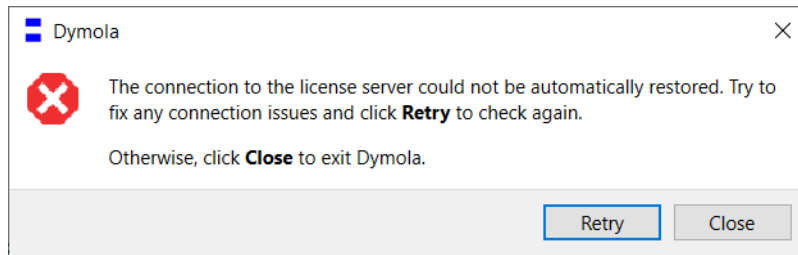


Note the tip about the environment variable to increase the time of the network timeout, if needed.

It is still possible to change the license file. You do this by answering **Change Server Settings** on the question whether to update the license server. This question follows clicking **OK** in the dialog above. This can be used for additional testing.

Checking for lost server connection

If you use the FLEXnet license server, Dymola checks the contact to the license server every 5 minutes. If the connection is lost, warning messages are displayed every 5 minutes. If the connection has been down for 20 minutes, Dymola displays the message:



In this case, you are locked out from normal use of Dymola and you have two options:

- Fix the connection problem and click **Retry**. Dymola will check again if the connection to the server works, and if so, you are returned to the Dymola session. If the check fails, Dymola loops back to the above dialog.
- Give up and accept that the connection to the license server cannot be restored. You then click on **Close**, meaning that Dymola will go through the usual process for the command **File > Exit**, giving you a chance to save modified models before closing Dymola.

For details, please see the document [Dymola 2023x: Updates for FLEXnet licenses](#).

License borrowing

Overview

Dymola on Windows can support "borrowing", the possibility to transfer a license from a license server to laptop for a limited period of time. If Dymola is used on a computer that is intermittently disconnected from a license server, that license can be issued as a sharable license with borrowing facility. Such a license can be borrowed from a license server via a special checkout and used later to run an application on a computer that is no longer connected to the license server.

For license borrowing, an end user initiates borrowing and specifies the expiration date a borrowed license is to be returned to the sharable license pool. While still connected to the network, the application is run from the client computer. This writes licensing information locally onto the client computer. The client computer can now be disconnected from the network.

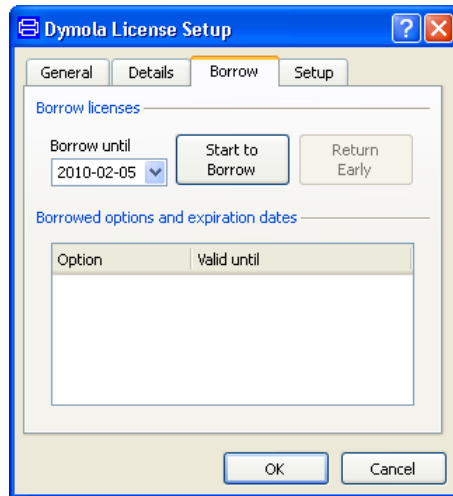
The license server keeps the borrowed license checked out. The client application automatically uses the local secured data file to do checkouts during the borrowing period. Upon the expiration of the borrowing period or the early return of a borrowed license, the local data file no longer authorizes checkouts and the license server returns the borrowed license to the pool of available licenses. No synchronization is required between the license server machine and the client machine when the borrowing period expires.

License borrowing

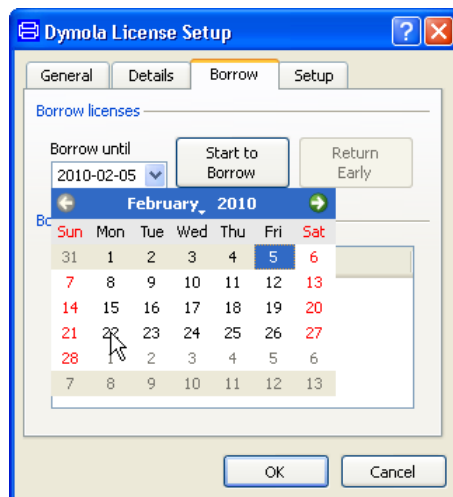
License borrowing and early returns are performed from Dymola.

In order to borrow, do the following:

1. While Dymola is connected to the server, use the command **Tools > License Setup**, and select the **Borrow tab**.

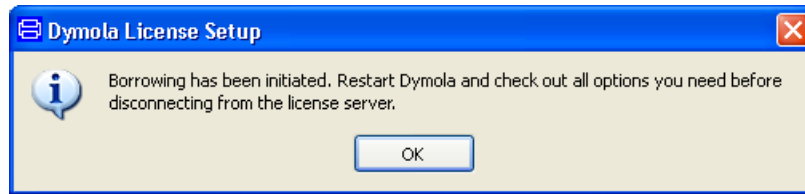


2. Select an end date, either by changing the date in the input field for **Last date borrowed** or by clicking on the arrow to display a calendar for selection of date. Clicking the arrow will display:

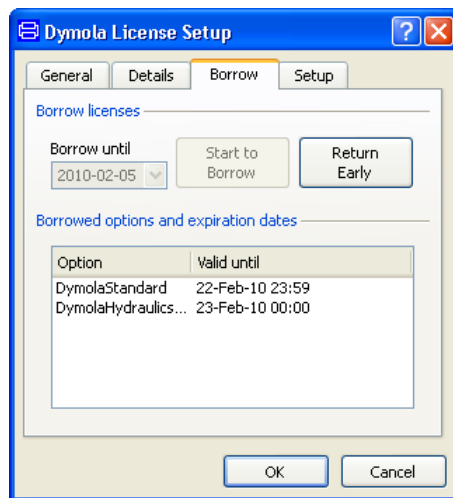


Here the possible selection of dates is clearly visible. Clicking on a date will change the input field to that date.

3. Click on **Start to Borrow**. The following message will appear:



4. Click **OK** and **OK** and restart Dymola (while still connected to the server); now the basic borrowing is performed. (Borrowing will be indicated in several ways, please see next section.)
5. Open all libraries/options that you will need during your borrowing time. This will ensure that the appropriate license features are stored locally. The list in the lower half of the dialog displays currently borrowed licenses and when they will be automatically returned to the server.



In this example, the Hydraulics library was opened; DymolaStandard indicates borrowing of Dymola without any options.

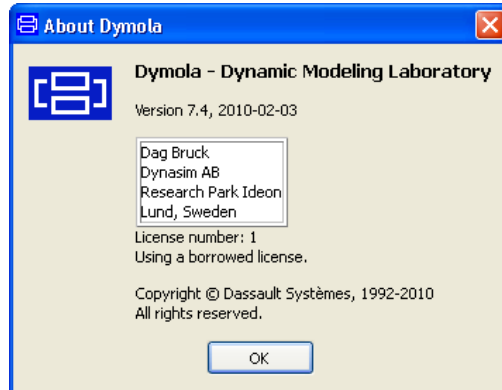
Please be careful not to open libraries/options that might be needed for others unless you really intend to do so. (Borrowing an option only available for one user only might not be appreciated by others.)

6. Finally disconnect from the license server **while Dymola is still running**. This step will create the local license file with the borrowed license. After disconnecting Dymola can be stopped.

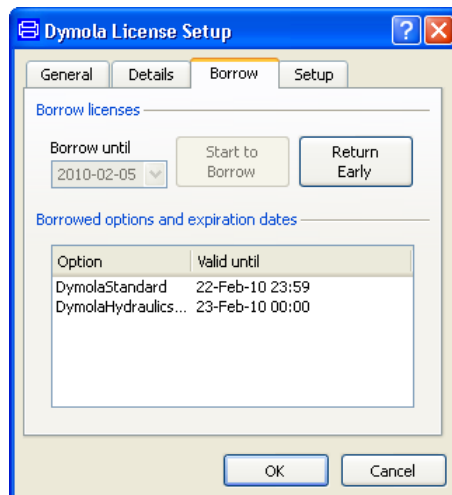
Running Dymola

During the borrowing period, Dymola can be started and stopped as often as needed. When license borrowing is used, Dymola displays it on the splash screen shown when starting Dymola and when using the command **Tools > About Dymola**:

Borrowing period in About dialog.



Most information is given using the command **Tools > License Setup**, in the **Borrow** tab.

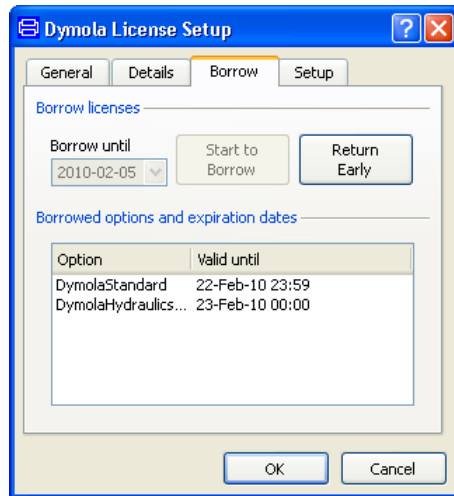


Returning a license before expiration of borrowing (early return)

Currently borrowed licenses can be returned early when the computer is connected to the license server again.

In order to do an early return, do the following:

While Dymola is connected to the server, use the command **Tools > License Setup**, and select the **Borrow** tab.



Now click on **Return Early**. The license (including all listed options) is returned to the server. Next time Dymola is restarted, the license is checked out the usual way.

It is a good idea to check e.g. the splash screen when starting up to convince oneself that the return was successful (in that case borrowing will not be mentioned in the splash screen).

A license returned to the license server cannot be checked out again until after approximately 2 minutes. If licenses are returned by e.g. exiting Dymola, but Dymola is restarted within approximately 2 minutes, the return is never performed.

License server options file

FLEXnet include tools for the local administrator. The options file allows the license administrator to control various operating parameters of the Dymola license server.

For example, it allows the administrator to

- Allow or deny the use of options by users.
- Reserve licenses for specified users.
- Control how many licenses can be borrowed and for how long.

The options file shall be called `dynasim.opt` and placed in the same directory as the Dymola license file of the license server.

An example of an options file that reserves a Dymola + Hydraulics library license for the user Bob is

```
RESERVE 1 DymolaStandard USER Bob
RESERVE 1 DymolaHydraulics USER Bob
```

Applicable “feature” and user names can be found in the license server log file. The details of the options file are described in Chapter 5 of “FLEXnet Licensing End User Guide”, which is available on request.

1.3.2 Dassault Systèmes License Server (DSLS)

General

As an alternative to the FLEXnet license server on Windows, Dassault Systèmes License Server (DSLS) is now also supported for Dymola, for sharable and nodelocked licenses.

Some notes about installation and use:

- To generate a target (host) id to order a license key, start Dymola with the command `dymola.exe /DSLS` (without license) and then, in Dymola, use the command **Tools > License Setup > Details** to copy the target id.
- A nodelocked DSLS license key should be saved in `C:\ProgramData\DassaultSystemes\Licenses`.
- The DSLS license server can be downloaded from the same location as the Dymola media. After downloading, you can install it as described in the DSLS documentation.
- For information on what license is selected when Dymola starts, and options to force selection of license, see section “Selection of license when starting Dymola” on page 29.
- To connect to an existing DSLS license server, use, in Dymola, the command **Tools > License Setup > Setup**. Type in the server name and press **OK**. The default port number can be used in almost any case.
- Dymola checks the contact to the license server the same way as for FLEXnet. See section “Checking for lost server connection” on page 73.
- DSLS license server has to be updated more often than the FLEXnet license server if you update to a new Dymola version. For DSLS the following is the case:
 - For each “autumn release” of Dymola, a new DSLS version is released, and you have to update to that one. (As an example, if you upgrade to Dymola 2025x, you must upgrade the DSLS to 2025x as well.)
 - If you update to the “spring/refresh version”, that is, e.g. Dymola 2025x Refresh 1, you don’t have to update the DSLS license, the previous DSLS version (DSLS 2025x in this case) is still valid.
 - Note that older versions of Dymola can use newer versions of DSLS, for example, Dymola 2022x can use DSLS 2025x.
- DSLS licenses use other ports than FLEXnet licenses.

Limitations:

- Borrowing of licenses (the Dymola command **Tools > License Setup > Borrow**) is not supported.
- The Dymola runtime concept is not supported, that is:

- Running FMUs that were exported without export license is not supported.
- Running executables `dymosim.exe` that were exported without export license is not supported.

Cloud service for license management (Managed DSLS)

Cloud service for license management, that is, the license server is managed for you by Dassault Systèmes, is supported from Dymola 2022x. The feature is sometimes referred to as “Managed DSLS”. The advantages for you are:

- No need to manage serves or license keys
- Automatic renewal of license keys
- Available on high-availability cloud server
- No additional cost

The requirements for “Managed DSLS” are:

- Support is active
- Access to Internet (HTTPS port) while Dymola is running

1.4 Dymola License Server on Linux

This section refers only to the Linux version of Dymola.

There are two possible license servers for Dymola on Linux. The default one is FLEXnet Publisher license server. As an alternative, the Dassault Systèmes License Server (DSLS) can be used.

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

1.4.1 FLEXnet Publisher license server

This section covers Linux-specific parts of the Dymola license server. For general items, e.g. background and how to set up the server using `lmtools.exe`, and checking for lost connection, please see corresponding section on Dymola License Server on Windows.

Note! Dymola requires support of FLEXnet Publisher version 11.16.2.1. This version is part of the Dymola distribution for Linux.

The Linux license server for Dymola is located in a separate tar file.

To start the server, the `dynasim` and the `lm*` files need to be installed, for example in `/usr/local/bin`. The server is started with the command

```
lmgrd -c <path to license file> -l <path to logfile>
```


A check with `pg aguxf` should show two new processes, `lmgrd` and `dynasim`. The server status can be checked with `lmutil lmstat -a`. In case of problems, the log file should be examined. Note also the general license server check at startup, see section “Testing the license server at startup” on page 72.

Note that it is possible from Dymola 2021 to use the web based license server `lmadmin` instead of the command line based `lmgrd` for both Linux and Windows. Please see section “Using the web based license server manager `lmadmin` instead of `lmgrd`” on page 71 for more details.

To start the license server automatically when the system is rebooted, please update e.g. `/etc/rc.d/rc.local` accordingly. Note that the license server needs not to run as “root”.

Note. The license server by default uses the ports 27000-27009. They can be configured if needed, e.g. if there are issues with firewalls.

Full details of FLEXnet license server installation can be found in the FLEXnet User’s Manual, which can be downloaded from www.flexera.com.

License borrowing on Linux

License borrowing is enabled by setting the environment variable `LM_BORROW`. The value must specify beginning and end dates of the borrowing period, as well as the vendor name “dynasim”. The general format is:

```
LM_BORROW=<start date>:dynasim:<end date>
```

An example (using bash) which specifies the start date 10 November 2009 and the end date 12 November 2009 is:

```
export LM_BORROW=10-nov-2009:dynasim:12-nov-2009
```

After setting the environment variable `LM_BORROW`, Dymola must be restarted and the appropriate license options checked out before disconnecting from the license server.

The status of borrowing can be displayed in the Linux server using a status command. An example, for Dymola 2025x Refresh 1, is:

```
/opt/dymola-2025xRefresh1-x86_64/bin64/lmutil lmborrow -status
```

The command displays the names of borrowed features and the expiration dates.

Returning a license before expiration of borrowing (early return)

Currently borrowed licenses can be returned early when the computer is connected to the license server again.

The names of the features that are currently borrowed can be seen using the status command in the previous section. When returning, any of these names must be used in the return command below.

In order to do an early return, give a return command while Dymola is connected to the server. An example returning the license for Pneumatics Library, for Dymola 2025x Refresh 1, is:

```
/opt/dymola-2025xRefresh1-x86_64/bin64/lmutil lmborrow -return -c  
~/dynamim/  
dymola.lic DymolaPneumaticsLib
```

Whether the return was made can be seen using the status command in previous section.

A license returned to the license server cannot be checked out again until after approximately 2 minutes. If licenses are returned by e.g. exiting Dymola, but Dymola is restarted within approximately 2 minutes, the return is never performed.

1.4.2 Dassault Systèmes License Server (DSLS)

General

As an alternative to the FLEXnet license server on Linux, Dassault Systèmes License Server (DSLS) is now also supported for Dymola, for sharable and nodelocked licenses.

Some notes about installation and use:

- To generate a target (host) id to order a license key, start Dymola with the command `dymola.exe /DSLS` (without license) and then, in Dymola, use the command **Tools > License Setup > Details** to copy the target id.
- A nodelocked DSLS license key should be saved in `var/DassaultSystemes/Licenses`.
- The DSLS license server can be downloaded from the same location as the Dymola media. After downloading, you can install it as described in the DSLS documentation.
- For information on what license is selected when Dymola starts, and options to force selection of license, see section “Selection of license when starting Dymola” on page 29.
- To connect to an existing DSLS license server, use, in Dymola, the command **Tools > License Setup > Setup**. Type in the server name and press **OK**. The default port number can be used in almost any case.
- Dymola checks the contact to the license server the same way as for FLEXnet. See section “Checking for lost server connection” on page 73.
- DSLS license server has to be updated more often than the FLEXnet license server if you update to a new Dymola version. For DSLS the following is the case:
 - For each “autumn release” of Dymola, a new DSLS version is released, and you have to update to that one. (As an example, if you upgrade to Dymola 2025x, you must upgrade the DSLS to 2025x as well.)
 - If you update to the “spring/refresh version”, that is, e.g. Dymola 2025x Refresh 1, you don’t have to update the DSLS license, the previous DSLS version (DSLS 2025x in this case) is still valid.

- Note that older versions of Dymola can use newer versions of DSLS, for example, Dymola 2022x can use DSLS 2025x.
- DSLS licenses use other ports than FLEXnet licenses.

Limitations:

- Borrowing of licenses (the Dymola command **Tools > License Setup > Borrow**) is not supported.
- The Dymola runtime concept is not supported, that is:
 - Running FMUs that were exported without export license is not supported.
 - Running executables `dymosim.exe` that were exported without export license is not supported.

Cloud service for license management (Managed DSLS)

Cloud service for license management, that is, the license server is managed for you by Dassault Systèmes, is supported from Dymola 2022x. The feature is sometimes referred to as “Managed DSLS”. The advantages for you are:

- No need to manage serves or license keys
- Automatic renewal of license keys
- Available on high-availability cloud server
- No additional cost

The requirements for “Managed DSLS” are:

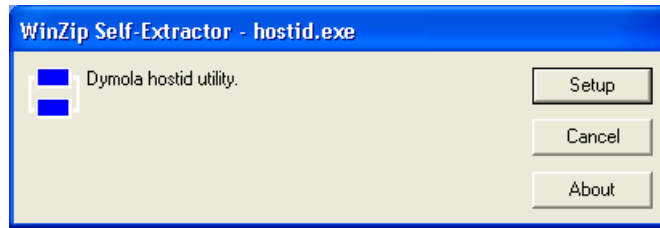
- Support is active
- Access to Internet (HTTPS port) while Dymola is running

1.5 Utility programs

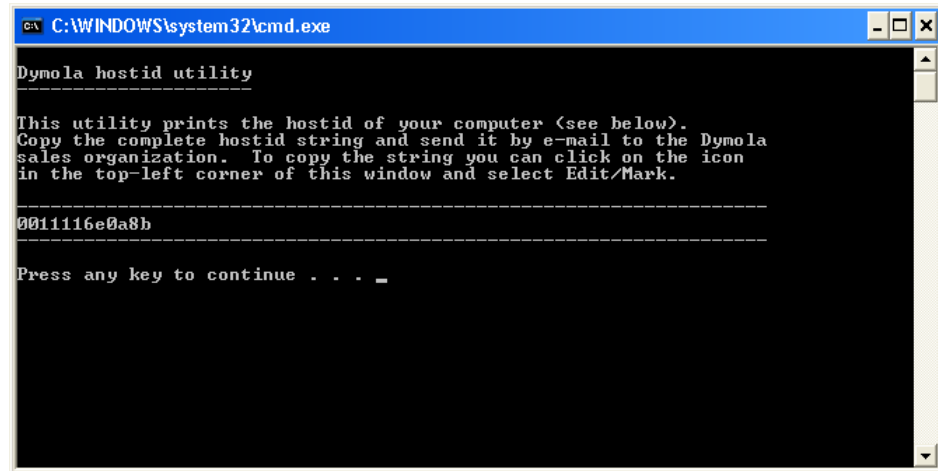
1.5.1 Obtaining a host id

To be able to easy find out the host id of a computer without having Dymola installed, a small file `hostid.exe` can be obtained from your Dymola distributor. (If Dymola trial version is installed, the host id can also be found using Dymola, please see section “Obtaining a host id” on page 25.)

Executing this file (by double-clicking it or opening it), the following menu will be displayed:



Selecting **Setup** will display the following:



Clicking in the upper left corner and selecting **Edit > Mark** makes it possible to selecting the host id by dragging the cursor over it. Once selected, **Edit > Copy** will place the host id in the clipboard, from where it should be pasted into a mail to your Dymola distributor.

1.6 System requirements

1.6.1 Hardware requirements

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

1.6.2 Hardware recommendations

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

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

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

1.6.3 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions

Dymola 2025x Refresh 1 is supported, as a 64 bit application, on Microsoft Windows 10 and Windows 11. Since Dymola does not use any features supported only by specific editions of Windows (“Home”, “Professional”, “Enterprise” etc.); all such editions are thus supported if the main version is supported.

Compilers

Please note that for the Windows platform, a Microsoft C/C++ compiler, or a GCC compiler, must be installed separately. The following compilers are supported for Dymola 2025x Refresh 1 on Windows:

Visual Studio - Free editions:

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

- Visual Studio 2015 Express for Windows Desktop Edition (14.0)
- Visual Studio 2017 Desktop Express **Note!** This compiler only supports compiling to Windows 32-bit executables
- Visual Studio Community 2017
- Visual Studio 2017 Build Tools **Notes:**
 - The recommended selection to run Dymola is the workload “Visual C++ build tools” + the option “C++/CLI support...”
 - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
 - This installation is not visible as specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2017 alternative: **Visual Studio 2017/Visual C++ 2017 Express Edition (15).**
 - For more information about installing and testing this compiler with Dymola, see [Installing Visual Studio Build Tools for Dymola.pdf](#).

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

Visual Studio - Professional editions:

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

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

- Visual Studio Professional 2022 (17). Note that you can select to use Clang code generator for this compiler.
- Visual Studio Enterprise 2022 (17). Note that you can select to use Clang code generator for this compiler.

Clang compilers: If you first select to use Visual Studio 2019 or Visual Studio 2022 as compiler, you can then select to use Clang as code generator instead of native Visual Studio. For more information about this, see section “Clang compiler” on page 12.

MinGW GCC compilers:

The following versions have been tested (hence, at least the versions in that range should work fine):

- 32-bit MinGW, with GCC version 6.3 and 8.2
- 64-bit MinGW, with GCC version 7.3 and 8.1

Note – the MinGW GCC compilers have some limitations, and demand for add-ons during installation etc., please see section “MinGW GCC compiler” on page 13.

Linux cross-compiler (WSL):

Dymola on Windows supports cross-compilation for Linux via the use of Windows Subsystem for Linux (WSL). The default WSL setup is 64-bit only and Dymola adopts this limitation. For more information, see “Cross-compilation for Linux on Windows” starting on page 17. Note that you can select to use Clang as code generator.

Linux

Supported Linux versions and compilers

Dymola 2025x Refresh 1 is supported on Red Hat Enterprise Linux (RHEL) version 8.6, 64-bit, with gcc version 11.2.1, and compatible systems (For more information about supported platforms, do the following:

- Go to <https://doc.qt.io/>
- Select the relevant version of Qt, for Dymola 2025x Refresh 1 it is 6.8.1. For earlier versions of Dymola, to find the relevant Qt version, see the corresponding Dymola Release Notes.
- Select Supported platforms)

Any later version of gcc is typically compatible.

In addition to gcc, the model C code can also be compiled by Clang. To be able to select Clang, it must be installed. See section “Compilation of model code” on page 64.

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

**Only 64-bit simulation
on Linux.**

Dymola 2025x Refresh 1 is supported as a 64-bit application on Linux. Corresponding support for 64-bit export and import of FMUs are included. Note the support for 32-bit simulation on Linux (including 32-bit export and import of FMUs) is discontinued from Dymola 2025x.

Notes:

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

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

1.6.4 Dymola license server

FLEXnet Publisher license server

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

**Dymola 2023 and
later.**

Dymola 2023 and later use FLEXnet Publisher version 11.16.2.1 which supports at least the following operating systems:

Microsoft Windows 32-bit:

- Windows 10
- Windows Server 2019
- Windows Server 2016
- Windows 7 SP1

Microsoft Windows 64-bit:

- Windows 10
- Windows Server 2019
- Windows Server 2016
- Windows 7 SP1

Linux 32-bit:

- RedHat Enterprise Linux 6 and 7
- SUSE Linux Enterprise 11 SP4 and 12 SP3

Linux 64-bit:

- RedHat Enterprise Linux 6, 7, and 8⁴
- SUSE Linux Enterprise 11 SP4, 12 SP3, and 15
- Ubuntu 16.04, 18.04, and 18.10

**Dymola 2018 –
Dymola 2022x**

Dymola 2018 up to, and including, Dymola 2022x, use FLEXnet Publisher version 11.14.0.2 which supports at least the following operating systems:

Microsoft Windows 32-bit:

- Windows 10
- Windows Server 2008, including SP1 and SP2
- Windows 8.1
- Windows 8
- Windows 7, including SP1
- Windows Server 2012 R2
- Windows Server 2012

Microsoft Windows 64-bit

- Windows 10
- Windows Server 2008, including SP1, SP2, and R2
- Windows 8.1
- Windows 8
- Windows 7, including SP1

⁴ For RedHat Enterprise Linux 8 (RHEL8) we have confirmed that the 64-bit license server works fine but not the 32-bit. To support also RHEL8, we have therefore added the 64-bit variant under bin64 in the Dymola distribution.

- Windows Server 2012 R2
- Window Server 2012

Linux 32-bit:

- RedHat Enterprise Linux 6 and 7
- SUSE Linux Enterprise 10, 11, and 12

Linux 64-bit:

- RedHat Enterprise Linux 6, 7, and 8⁵
- SUSE Linux Enterprise 10, 11, and 12

Dassault Systèmes License Server (DSLS)

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

Note that there are some limitations for DSLS, for example, borrowing of licenses is not supported for this license server. For more information, see section “Dassault Systèmes License Server (DSLS)” starting on page 79.

1.7 License requirements

1.7.1 General

This description covers the license requirements for different features; for description of the features themselves, please see relevant documentation in “*Dymola User Manual 2B: Simulation Interfaces and Export*”, chapter “Simulation Environments”, and the documentation for the libraries.

This description assumes that machines having Dymola installed have the Dymola **Standard Configuration** license.

Apart from the **Standard Configuration** license, there are other licenses for different categories of users:

- **Trial** license. This license has some limitations compared with the Standard Configuration license.
- Licenses for academia (**Academic Learn/Innovate**, **Student Learn/Innovate**)
The Student licenses have some limitations compared to the Academic ones, for

⁵ For RedHat Enterprise Linux 8 (RHEL8) we have confirmed that the 64-bit license server works fine but not the 32-bit. To support also RHEL8, we have therefore added the 64-bit variant under bin64 in the Dymola distribution.

example, it is not possible using a Student license to run dymosim.m from Matlab/Simulink or dymosim.exe from a command window. You need any of the Academic licenses to do that.

Licenses can be node-locked (to a certain machine) or sharable (accessible from a license server). Note that node-locked and sharable licenses cannot be mixed.

1.7.2 License for Dymola – Simulink interface

To use the traditional connection between Dymola and Simulink (Dymola-on-Windows Mode), you need a **Simulink Interface** license in Dymola.

Note that this mode is not available for Linux. For Linux, see section “Import to Simulink”; the second part.

1.7.3 License for real-time simulation

In Dymola

To be able to enable inline integration, needed for real-time simulation, you must have the **Real-Time Simulation** license. (The **Source Code Generation** license includes this functionality as well.) The **Real-Time Simulation** license is from Dymola 2017 FD01 included in the Dymola standard license.

On dSPACE and xPC Target platforms

To be able to use real-time simulation on dSPACE or xPC Target platforms, you must have

- The **Simulink Interface** license in Dymola to use the Dymola-Simulink interface.
- Any of the licenses **Real-Time Simulation** or **Source Code Generation**, depending on what you want to do:
 - If you only want to use models that use inline integration, the **Real-Time Simulation** license is sufficient. This license is from Dymola 2017 FD01 included in the Dymola standard license.
 - If you want to use other models as well, you need the **Source Code Generation** license.

1.7.4 Licenses for exporting code

The assumption is that the model is to be imported/executed on another machine than the one where it was created.

Exporting a Dymola model to a machine without a Dymola license

See the corresponding import section below, that also explains the export.

Exporting a model to Simulink

In Windows you can use the “Dymola-on-Windows” mode of the Dymola-Simulink interface (using DDE communication between Dymola and Simulink). In this mode, you can export a compiled model and execute it on another Windows computer. You don’t need any license to export the compiled model, but there might be requirements depending on how you want to use it – see the section about importing a model to Simulink below.

For other platforms (for example Linux) – but also optionally in Windows – you can use the “Import” mode of the Dymola-Simulink interface. You can export models to be imported in this mode. To do this, you must use the `ExternalInterfaces` library, the `ExternalInterfaces.Export.toSimulink` function. You need the **Simulink Interface** license to do this. Depending on how the model should be used, there are additional requirements, see section “Importing a model to Simulink” below.

Exporting a model for integration with Real-Time Workshop (dSPACE and xPC)

You can do this by using the `ExternalInterfaces` library, the `ExternalInterfaces.Export.toSimulink` function. You need to have both the **Simulink Interface** license, and the **Real-Time Simulation** license to do such an export. (If you have a **Source Code Generation** license, that one includes **Real-Time Simulation** as well.) The **Real-Time Simulation** license is from Dymola 2017 FD01 included in the Dymola standard license.

Exporting a model to FMU format

See the section “Importing an FMU” below; this section also explains the export.

Exporting a model to Binary

You can export a model to binary. This requires the license **Binary Model Export**. (The **Source Code Generation** license includes this functionality as well.) Such an exported model can be executed on a machine without having any Dymola license.

Exporting a model to Source Code (C code)

You can export a model to source code. This requires the license **Source Code Generation**. Such an exported model can be executed on a machine without having any Dymola license. Additional features are available, see the manual “*Dymola User Manual 2B: Simulation Interfaces and Export*”, chapter “Simulation Environments”.

1.7.5 Licenses for executing/importing/using code

Executing a Dymola model on another computer

There are two cases:

- The model has been generated by a machine having a **Binary Model Export** license (or **Source Code Generation** license). In this case, no Dymola license is required to execute the model.
- The model has been generated without any of the two licenses above. You can execute the model if the machine has a Dymola **Standard Configuration** license, and the environment variable DYMOLA_RUNTIME_LICENSE is set to the path where the license is located. (This is needed for the executable model to find the license file, see “Specifying the license key by the environment variable DYMOLA_RUNTIME_LICENSE” on page 28.) **Note.** This alternative is not supported for the following cases:
 - Using the MinGW GCC compiler, see limitations in section “MinGW GCC compiler” starting on page 13.
 - Using a DSLS license, see limitations in section “Dassault Systèmes License Server (DSLS)” starting on page 79. The limitation is also valid for nodelocked DSLS licenses.

Import to Simulink

If you have used the “Dymola-On-Windows” mode to export a compiled model, you can use it in a number of ways:

- You can execute the model on a Windows computer without Dymola installation in two cases:
 - The model has been generated by a machine having a **Binary Model Export** license (or **Source Code Generation** license). In this case, no Dymola license is required to execute the model.
 - The model has been generated without any of the two licenses above. You can execute the model if the machine has a Dymola **Standard Configuration** license, and the environment variable DYMOLA_RUNTIME_LICENSE is set to the path where the license is located (see “Specifying the license key by the environment variable DYMOLA_RUNTIME_LICENSE” on page 28). **Note.** This alternative is not supported for the following cases:
 - Using the MinGW GCC compiler, see limitations in section “MinGW GCC compiler” starting on page 13.
 - Using a DSLS license, see limitations in section “Dassault Systèmes License Server (DSLS)” starting on page 79. The limitation is also valid for nodelocked DSLS licenses.

- You can import it in Simulink using the “Import” mode of the Dymola-Simulink interface. This requires that the model have been generated by a machine having a **Binary Model Export** license (or **Source Code Generation** license).

If you have exported a model using the using the ExternalInterfaces library, the `ExternalInterfaces.Export.toSimulink` function, you can import it using the “Import” mode of the Dymola-Simulink interface. Such an imported model can be compiled, simulated, and also downloaded to real-time platforms (the last option requires that it has been exported from a machine having the **Real-Time Simulation** license, or the **Source Code Generation** license). The **Real-Time Simulation** license is from Dymola 2017 included in the Dymola standard license.

Importing an FMU

You have three cases:

- The FMU is a Dymola FMU, and generated by a machine having a **Binary Model Export** license (or **Source Code Generation** license); in this case, you can import it and execute it on a computer without a Dymola license.
- The FMU is a Dymola FMU, and generated by a machine without any of the licenses above; in this case, you must fulfill two conditions to be able to execute the FMU: (**Note.** This alternative is not supported for the MinGW GCC compiler; see limitations in section “MinGW GCC compiler” starting on page 13.)
 - The machine you import to must have a Dymola **Standard Configuration** license.
 - The environment variable `DYMOLA_RUNTIME_LICENSE` must be set to the path where the license is located. (This is needed for the dll of the FMU to find the license file.) For information about this variable, see “Specifying the license key by the environment variable `DYMOLA_RUNTIME_LICENSE`” on page 28. **Note.** This alternative is not supported for the following cases:
 - Using the MinGW GCC compiler, see limitations in section “MinGW GCC compiler” starting on page 13.
 - Using a DSLS license, see limitations in section “Dassault Systèmes License Server (DSLS)” starting on page 79. The limitation is also valid for nodelocked DSLS licenses.
- The FMU is from another vendor. For this case, please see the documentation for that vendor; there are no additional requirements from Dymola.

1.7.6 Licenses for libraries in the Dymola library menu

A number of libraries in the Dymola library menu can be used in Dymola without any additional licenses, a number of libraries require additional licenses when working with more advanced features, and a number of libraries always require additional licenses.

Note:

- If you want to export code, the above requirements for export licenses apply as well.

Free libraries and packages in the Dymola library menu

- DataFiles
- Design.Calibration
- Design.Experimentation
- Design.Validation
- DymolaCommands
- DymolaModels
- FTire Interface Library
- Modelica Reference
- Modelica Standard Library
- Modelica_DeviceDrivers
- Modelica_LinearSystems2
- Modelica_StateGraph2
- ModelManagement
- Plot 3D
- SDF (Scientific Data Format)
- Thermal Management Demos (in the menu **File > Demos**; note that commercial libraries are needed to run the demo)
- Testing Library
- UserInteraction
- Vehicle Demos (in the menu **File > Demos**; note that commercial libraries are needed to run the demo)
- VehicleInterfaces

Libraries in Dymola library menu that require additional licenses when for example running more advanced examples

- Optimization (and the older Design.Optimization) – requires **Design Optimization** license when running more advanced examples.

Libraries in Dymola library menu always requiring additional licenses

Libraries in the Dymola library menu not listed above always require additional licenses.

1.7.7 Licenses for libraries not in Dymola library menu

There are a number of libraries not in Dymola library menu, for example:

- Libraries can be downloaded from the homepage of Modelica Association (www.Modelica.org) or be downloaded from GitHub (for the latter, see “On-demand downloading and installing libraries from GitHub” starting on page 44). Some of these libraries are free, some require license.
- Third party vendors can use the license handling in Modelica to add license requirements on their libraries.

1.8 Troubleshooting

This is a common section for both Windows and Linux. If a problem only is applicable for e.g. Linux, it is stated.

Occasionally the installation will not succeed, or the program will not operate as intended after installation. This section will outline some of the problems that have been detected in the past.

1.8.1 License file

The license file used is not the one wanted

There are a number of standard paths where Dymola searches for a valid license. In an old invalid license is stored by mistake in one of those locations, that license might be tried instead of the correct one. Information about which license is currently in use by Dymola is given using the command **Tools > License Setup > Setup**. The path to that license is specified by `Filename` in that tab.

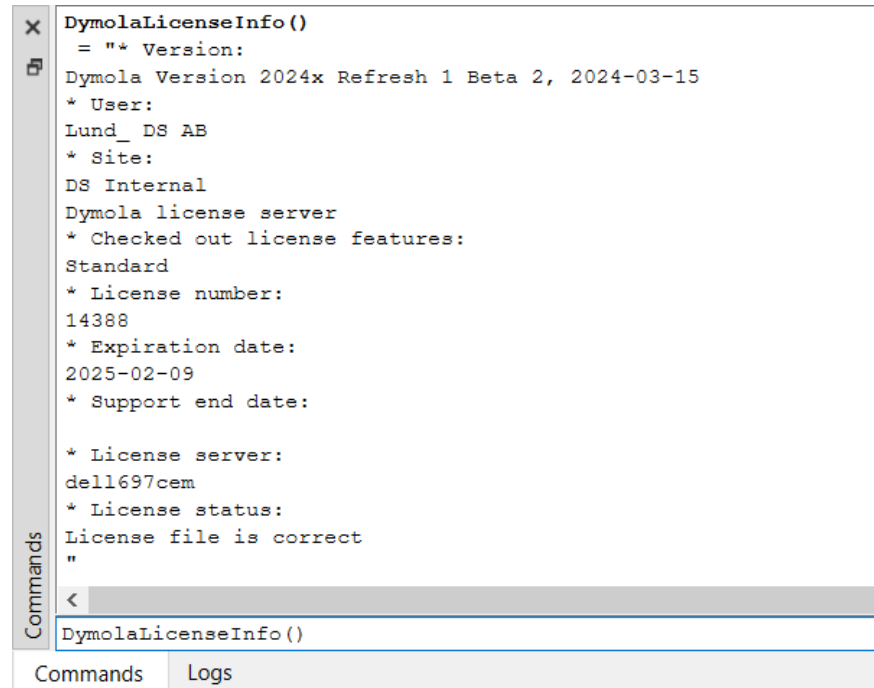
License file is not authentic

The error message “License file not authentic” indicates either an error in the license file, or a mismatch between your computer system and your license file.

- The license file is locked to your computer system, which means that you cannot execute Dymola from another computer.
- The license file format has been changed in Dymola 7.0 and later versions. If you also have older versions of Dymola installed, please check that you have a new license file as well.

Using a built-in function to get license information

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



```
DymolaLicenseInfo()
= "* Version:
Dymola Version 2024x Refresh 1 Beta 2, 2024-03-15
* User:
Lund_DS AB
* Site:
DS Internal
Dymola license server
* Checked out license features:
Standard
* License number:
14388
* Expiration date:
2025-02-09
* Support end date:

* License server:
dell697cem
* License status:
License file is correct
"
<
DymolaLicenseInfo()
```

The screenshot shows a software interface with a 'Commands' tab selected. The command window displays the output of the `DymolaLicenseInfo()` function, which includes version information, user details, site information, license features, license number, expiration date, support end date, license server, and license status. The status is 'License file is correct'. Below the command window, there are tabs for 'Commands' and 'Logs'.

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

Additional information

If there is some error in the license file or with the license server, Dymola presents a short error message by default. On top of that, you can use the built-in function `DymolaLicenseInfo()` as described above. For a FLEXnet license, a more detailed description, including FLEXnet Publisher error codes, is produced if Dymola is started with the command line option `/FLEXlmDiag`. On Windows, start a command (DOS) window (using the command **Start > All Programs > Accessories > Command Prompt** in Windows) and issue the following commands (assuming Dymola 2025x Refresh 1 64-bit is used on a 64-bit computer):

```
cd \Program Files\Dymola 2025x Refresh 1\bin64
dymola.exe /FLEXlmDiag
```

On Linux the command will be:

```
dymola /FLEXlmDiag
```

The additional information is usually helpful in any correspondence with support.

License server

For FLEXnet, correct operation of the FLEXnet Publisher license server should be verified with `lmtools.exe`, see “Installing the license server” on page 68. The FLEXnet Publisher logfile provides additional information about the day-to-day operation of the server.

Always using the latest version of the FLEXnet Publisher license daemon `lmgrd.exe` is strongly recommended. It is guaranteed to be compatible with all earlier versions of FLEXnet Publisher.

License borrowing

Note that borrowing is not supported for the Dassault Systèmes License Server (DSLS).

Different versions of Dymola

There are limitations regarding license borrowing when borrowing is done in one version of Dymola, and using the borrowed license is used in another version of Dymola on the same PC.

For Windows, a license borrowed using Dymola 7.4 FD01 or older cannot be used by Dymola 2012 or newer without being connected to the license server.

For Linux, a license borrowed using Dymola 2012 FD01 or older cannot be used by Dymola 2013 or newer without being connected to the license server.

If access to e.g. both Dymola 7.4 FD01 and Dymola 2012 is required on a Windows PC, both versions must be used to borrow, by the following procedure:

- Initiate borrowing with any Dymola version.
- Open Dymola 7.4 FD01 (or older) and check out the required features.
- Open Dymola 2012 (or newer) and check out the required features.
- Validate by checking that there are two entries of all the required features in the **Details** tab, using the command **Tools > License Setup**.
- Disconnect from the network and validate that both versions can be run as expected.

License borrowing of 32/64-bit Dymola

When borrowing license, only the license of the Dymola version you run will be borrowed: and 64-bit and 32-bit Dymola are seen as different versions. For the few cases when you want to

- Borrow a license, AND
- run Dymosim.exe outside of Dymola, AND
- do not have export option;

we advise that you in 64-bit Dymola generate a 64-bit Dymosim.exe using the flag `Advanced.CompileWith64=2`. For more information about this flag, see the manual “*Dymola User Manual 1B: Developing and Simulating a Model*”, chapter “Simulating a model”, section “Dynamic Model Simulator”.

Sharable licenses

Please note that if a new session is started in Windows by using **Log Off > Switch User** the original user is still logged on and any Dymola program occupies a sharable license.

1.8.2 Compiler problems

See section “Troubleshooting: Verify compiler button and error messages” starting on page 20.

1.8.3 Simulink

If the Dymola-Simulink interface does not work, please check the following (some of which may sound elementary):

- You have a Dymola license that supports the Simulink interface. Note that Simulink support is a separate option.
- You have included the three directories `Dymola 2025x Refresh 1\mfiles`, `Dymola 2025x Refresh 1\mfiles\traj` and `Dymola 2025x Refresh 1\mfiles\dyntools` in the Matlab path. These have to be included every time you want to use the Dymola-Simulink interface and it is a good idea to store the included paths in Matlab.
- You can find the interface in Simulink's browser as `Dymola Block/DymolaBlock` (if not, you have probably not included the directories, mentioned above, into the Matlab path).
- Make sure that you have a Visual Studio C++ compiler installed on your computer. Make sure that the Matlab mex utility has been configured to use that compiler (type `mex -setup` in Matlab to configure). Finally, test by trying to compile and link an example mex file, e.g. `matlab\extern\examples\mex\yprime.c`.
- You have created external inputs to the Dymola Block, and outputs from the Dymola Block, in a correct way. See also the manual “*Dymola User Manual 2B: Simulation Interfaces and Export*”, chapter “Simulation Environments”, section “Dymola – Matlab interface”, subsection “Using the Dymola-Simulink interface”.
- You have compiled all Dymola models used in the model; otherwise, you will get an error message.
- If “Allow multiple copies of block” is unchecked, you should not copy the block. Unchecking it should only be done if you have a dSPACE system.

Also, note that the parameterizations differ between blocks in the Modelica Standard Library and in Simulink. For example, the frequency of Simulink's Sine-block is measured in rad/s,

which is commonly known as angular frequency and should thus be 2π times the frequency in the corresponding source in Modelica.

Only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. The LCC compiler is not supported.

1.8.4 Change of language

Dymola is available in Japanese. Sometimes the user wants to change the language after installation. Please see section “Language” on page 30 on how to do this.

1.8.5 Other Windows-related problems

Starting the installation

The installation normally starts automatically when you insert the distribution DVD. If auto start has been disabled, please start `D:\setup.exe` (assuming your DVD drive is labeled D) from Windows Explorer by double-clicking on the file or use the **Start** button in Windows, select **Run**, enter `D:\setup.exe` and click **OK**.

Deep directory hierarchies

Compilation and simulation of the model may fail in a very deep directory hierarchy, if the length of the directory path exceeds 250 characters. This is caused by a bug in Microsoft software, and we are investigating ways to work around it.

Writable root directory

Due to a bug in some versions of the Microsoft runtime library, the root directory `C:\` should be writable in order to store temporary files. If that is not the case, Dymola will create working files in the current directory, which are not automatically deleted.

2 Index

4

4K
using 4K screens, [32](#)

A

AddModelicaPath, [40](#)

B

borrowing
general, [74](#)
on Linux, [81](#)
on Windows, [74](#)

C

C compiler, [10](#)
clang (on Linux), [61](#)
clang (on Windows), [10](#)
clang compiler, [12](#)
compiler, [10](#)

clang, [12](#)
clang (on Linux), [61](#)
clang (on Windows), [10](#)
cross-compilation for Linux on Windows, [10](#)
GCC (on Linux), [61](#)
GCC (on Windows), [10](#)
installation, [10](#)
Microsoft Visual C++, [10](#)
troubleshooting, [20](#)
Visual Studio, [10](#)

D

daemon
vendor, [67](#)
dark mode, [32](#)
DSLS
on Linux, [82](#)
on Windows, [79](#)
Dymola
32-bit application on Linux, [61](#)
32-bit application on Windows, [6](#)
64-bit application on Windows, [6](#)
checking for updates, [59](#)
latest release available, [59](#)

latest release available - highlights, [59](#)
Dymola license server
on Linux, [80](#)
on Windows, [67](#)
DymolaLicenseInfo, [97](#)

E

environment variables
DYMOLA, [64](#)
DYMOLA_JAVA_HOME, [56](#)
DYMOLA_RUNTIME_LICENSE, [28](#)
DYMOLAPATH, [64](#)
DYMOLAWORK, [31](#)
JAVA_HOME, [56](#)
LM_BORROW, [81](#)
MODELICAPATH, [38](#), [64](#)

F

file extensions
.cab, [35](#)
.msi, [35](#)
FLEXnet, [67](#)
license server options file, [78](#)
supported operating systems, [88](#)
FMU export for multiple platforms, [19](#)

G

GCC (on Linux), [61](#)
GCC (on Windows), [10](#)
GetDymolaCompiler, [20](#)
GitHub
download and install libraries, [44](#)

H

host id, [25](#)

I

incorrect license file format, [30](#)
installation
environment variables, [64](#)
license daemon, [67](#)
license server on Linux, [80](#)
license server on Windows, [67](#)
Linux, [61](#)
remote on Windows, [35](#)

troubleshooting, [96](#)
windows, [6](#)

J

java
environment variables, [56](#)
java runtime environment, [56](#)

L

language setting, [30](#)
Japanese, [30](#)
library
customizing the File > Libraries menu, [36](#)
download and install from GitHub, [44](#)
managing libraries location, [38](#)
managing multiple versions of a library, [49](#)
Modelica libraries newer than the ones in Dymola
distribution, [35](#)
Modelica libraries older than the one in Dymola
distribution, [36](#)
on-demand installation of downloaded libraries, [42](#)
libraryinfo.mos, [47](#)

license
borrowing - general, [74](#)
borrowing - on Linux, [81](#)
borrowing - on Windows, [74](#)
daemon. *See* license : server manager
demands, [90](#)
early return - on Linux, [81](#)
early return - on Windows, [77](#)
error message, [96](#)
expiration - settings, [52](#)
incorrect file format, [30](#)
node-locked, [25](#)
not found or invalid - settings, [52](#)
preventing checking out - in Linux, [66](#)
preventing checking out - in Windows, [53](#)
requirements, [90](#)
runtime concept, [28](#)
server
on Linux, [80](#)
on Windows, [67](#)
server manager, [67](#)
server options file, [78](#)
sharable, [23](#)
support period, [29](#)
support period termination (TOS), [29](#)
TOS (Termination of Support), [29](#)

Linux
cross-compilation for Linux on Windows, [10](#)

- Dymola as 32-bit application, [61](#)
- Dymola license server, [80](#)
- installation, [61](#)
- local host id, [25](#)
- login
 - remote, [55](#)

M

- Microsoft
 - Visual C++, [10](#)
 - Windows installation, [6](#)
- Modelica path, [38](#)
- Modelica Standard Library
 - newer than the one in Dymola distribution, [35](#)
 - older than the one in Dymola distribution, [36](#)
- MODELICAPATH, [38](#)

N

- node-locked license, [25](#)

P

- Program Files, [7](#)

R

- release
 - latest Dymola release available, [59](#)
- remote login, [55](#)
- resolution
 - using 4K screens, [32](#)
- runtime
 - concept, [28](#)
- run-time. *See* runtime

S

- script
 - startup script, [34](#)
- scripting
 - compiler settings, [20](#)
- SetDymolaCompiler, [20](#)

- settings
 - custom setup file, [51](#)
- setup file
 - custom setup file, [51](#)
- sharable licenses, [23](#)
- Simulink
 - troubleshooting, [99](#)
- splash screen, [34](#)
- starting Dymola
 - selection of license, [29](#)
 - startup script, [34](#)
- system requirements
 - hardware, minimum, [84](#)
 - hardware, recommended, [84](#)
 - software, Linux, [87](#)
 - software, Windows, [85](#)
- Systems Simulation Design, [18](#)

T

- teleworking, [55](#)
- troubleshooting
 - compiler, [20](#)
 - language, [100](#)
 - license borrowing, [98](#)
 - license file, [96](#)
 - license server, [98](#)
 - other windows problems, [100](#)
 - Simulink, [99](#)

U

- upgrading Dymola, [30](#)

V

- Visual Studio compiler, [10](#)

W

- Windows
 - Dymola as 32-bit application, [6](#)
 - Dymola as 64-bit application, [6](#)
 - Dymola license server, [67](#)