

COMSOL Multiphysics

Physics Builder Manual

Physics Builder Manual

© 1998–2017 COMSOL

Protected by U.S. Patents listed on www.comsol.com/patents, and U.S. Patents 7,519,518; 7,596,474; 7,623,991; 8,457,932; 8,954,302; 9,098,106; 9,146,652; 9,323,503; 9,372,673; and 9,454,625. Patents pending.

This Documentation and the Programs described herein are furnished under the COMSOL Software License Agreement (www.comsol.com/comsol-license-agreement) and may be used or copied only under the terms of the license agreement.

COMSOL, the COMSOL logo, COMSOL Multiphysics, Capture the Concept, COMSOL Desktop, LiveLink, and COMSOL Server are either registered trademarks or trademarks of COMSOL AB. All other trademarks are the property of their respective owners, and COMSOL AB and its subsidiaries and products are not affiliated with, endorsed by, sponsored by, or supported by those trademark owners. For a list of such trademark owners, see www.comsol.com/trademarks.

Version: COMSOL 5.3

Contact Information

Visit the Contact COMSOL page at www.comsol.com/contact to submit general inquiries, contact Technical Support, or search for an address and phone number. You can also visit the Worldwide Sales Offices page at www.comsol.com/contact/offices for address and contact information.

If you need to contact Support, an online request form is located at the COMSOL Access page at www.comsol.com/support/case. Other useful links include:

- Support Center: www.comsol.com/support
- Product Download: www.comsol.com/product-download
- Product Updates: www.comsol.com/support/updates
- COMSOL Blog: www.comsol.com/blogs
- Discussion Forum: www.comsol.com/community
- Events: www.comsol.com/events
- COMSOL Video Gallery: www.comsol.com/video
- Support Knowledge Base: www.comsol.com/support/knowledgebase

Part number: CM020009

C o n t e n t s

Chapter 1: Introduction

About the Physics Builder	14
What Can You Do With the Physics Builder?	14
Where Do I Access the Documentation and Application Libraries?	15
Overview of the Manual	17

Chapter 2: Physics Builder Design

Overview of the Physics Builder	20
Creating a New Physics Builder File	20
The Physics Builder Window	21
The Physics Builder Manager	22
Saving and Opening Custom Physics Interfaces.	22
Designing the GUI Layout	23
User Inputs and GUI Components	23
User Input Group GUI Options	26
Entering Names and Expressions	31
Entering Names	31
Using Customized Names and Descriptions.	34
Entering Names of Operators and Functions	35
Adding a Delimiter to a String	36
Tensor Parser	36
Using Ctrl+Space to Access Expressions	39
Using Coordinate Systems	41
The Base Vector System	41
The Input Base Vector System	42
Transformation Between Coordinate Systems	43

Specifying Selections	46
Selection Section Settings	46
Selection Terminology	48
The Physics Builder Manager	51
Testing Custom Physics Interfaces	51
The Development Files	52
Compiling an Archive	52
Working with Builder Archives	53
Searching in Archives	55

Chapter 3: Physics Builder Tools

Building Blocks	58
Components	58
Properties	59
Features	59
Multiphysics Couplings.	59
Code Editor.	60
About Links.	60
Dependencies	60
External Resources	62
Import.	62
Definitions Library	63
Components	64
Creating Components	64
Component.	65
Physics Interface Component	66
Usage Condition	66
Equation Display	70
Component Link	72
Extra Dimension Link	73

Properties	76
Property	76
Property Link	77
Tensor-Valued Function	78
Physics and Multiphysics Interfaces	79
Creating a Physics Interface or a Multiphysics Interface	80
Physics Interface	81
Multiphysics Interface	84
Contained Interface.	85
Physics Interface Component Link	86
Auxiliary Settings (Physics Interface)	87
Disable Allowed Study Types	89
Menu	89
Menu Item	89
Physics Interface — Preview	90
Multiphysics Interface — Preview.	90
Features	91
Generic Feature	92
Domain Condition	96
Boundary Condition	97
Global Feature.	97
Domain Feature	98
Boundary Feature	98
Edge Feature	98
Point Feature	99
Pair Feature.	99
Contact Pair Feature	100
Device Model Feature	101
Moving Frame Domain Condition	101
Moving Frame Boundary Condition	102
Periodic Feature	104
Feature Link.	105
Multiphysics Feature.	105
Multiphysics Coupling	106
Generic Multiphysics Coupling.	107
Global Multiphysics Coupling	108

Domain Multiphysics Coupling	109
Boundary Multiphysics Coupling	109
Edge Multiphysics Coupling	109
Point Multiphysics Coupling	110
Coupling Type Contribution	110
Contained Feature	110
Auxiliary Settings (Feature Nodes)	111
Auxiliary Settings (Multiphysics Couplings)	113
Geometric Nonlinearity	113
Physics Symbol.	114
User Inputs	117
Creating User Inputs	117
User Input	119
Selectable Input	121
Selection Input.	121
Boolean Input	124
User Input Group	125
Text Label	126
Buttons	127
Section	128
Constraint Settings Section	128
Material Property.	130
External Material List	132
Socket Input	133
Socket Output.	134
Material List.	135
Feature Input	136
Activation Condition	139
Additional Requirement	140
Allowed Values	140
Activating Allowed Values	141
Button.	142
Integer Values Check	143
Real Values Check	143
Regular Expression Check	144
Named Group Members	144

Variables	145
Creating Variables	145
Variables for Degrees of Freedoms	146
Variable Declaration	146
Variable Definition	150
Dependent Variable Definition.	153
Dependent Variable Declaration	154
Discretization	158
Initial Values.	159
Hide in GUI.	159
Disable in Solvers	160
Degree of Freedom Initialization	161
Component Settings	162
Frame Shape	164
ODE States Selection	164
Equations	166
Weak Form Equation	166
General Form Equation	167
Coefficient Form Equation	169
Boundary Element Equation.	171
Shared Quantity Definition	173
Constraints	175
Constraint	175
Weak Constraint.	177
Excluding Selection	178
Device Systems	179
Creating Device Systems	179
Device Model	180
Port Model	181
Device Constants	182
Device Inputs	182
Device.	182
Input Modifier	183
Device Variables	183
Device Equations.	184

Port.	184
Port Connections	185
Device Feature	185
Operators and Functions	187
Operators	187
Functions.	187
Average	188
Integration	188
General Extrusion	189
Maximum.	189
Minimum	189
Integration Over Extra Dimension	189
Physics Areas	190
Physics Area	190
Predefined Multiphysics	191
Contained Multiphysics Coupling.	191
Contained Interface (Predefined Multiphysics)	192
Selections	193
Selection	193
Selection Filter Sequence.	193
Override Rule Filter.	194
Selection Component Filter.	195
Multiphysics Coupling Selection Filter	195
Extra Dimension Selection	196
Extra Dimensions	198
1D Interval	198
Multiple 1D Intervals	198
2D Rectangle	199
2D Circle.	199
3D Sphere	200
Auxiliary Definitions	201
Material Property Group.	201
Material Property (Auxiliary Definitions)	202

Physical Quantity	202
Override Rule	203
Plot Menu Definition	205
Equation Display (Auxiliary Definitions)	205
Mesh Defaults	207
Mesh Size.	207
Study and Solver Defaults	208
Field	209
Absolute Tolerance	209
Segregated Step	210
Outer Job Parameters	210
Eigenvalue Transform	211
Study Sequence	211
Stationary	212
Time Dependent	212
Result Defaults	213
Plot Defaults	214
Migration	216
About Backward Compatibility	216
Version	217
Physics Interface (Migration)	218
Feature (Migration)	218
Property (Migration)	218
Change Type	218
Rename Inputs.	219
Migration Links	219
Documentation	220
Introduction to Comments and Documentation	220
Physics Interface Documentation.	221
User Documentation	221
Comments	222
The Documentation Node	223
Documentation Text Components	224

The Preview Window	227
Elements	228
Element	228
GeomDim	228
Src	229
Array	229
Record	229
String	229
Elinv.	230
Elpric	230
Event	231
DG Wave Element, General Form	231
Degree of Freedom Re-Initialization.	232
Shape Interpolation Element	233

Chapter 4: Examples of Custom Physics

The Thermoelectric Effect	236
Introduction to the Thermoelectric Effect	236
Equations in the Physics Builder	237
Thermoelectric Effect Implementation	241
Overview.	241
Thermoelectric Effect Interface — Creating It Step by Step.	245
Testing the Thermoelectric Effect Interface	262
Example Model — Thermoelectric Leg	265
Introduction to the Thermoelectric Leg Model	265
Results.	266
Reference	267
Modeling Instructions	267
The Schrödinger Equation	270
Introduction to the Schrödinger Equation	270

Schrödinger Equation Implementation	271
Overview	271
Schrodinger Equation Interface — Creating It Step by Step	273
Testing the Schrodinger Equation Interface	282
Example Model — Hydrogen Atom	284
Introduction to the Hydrogen Atom Model	284
Results	284
Modeling Instructions	287

Introduction

This guide describes the Physics Builder, a set of tools for creating custom physics interfaces directly in the COMSOL Desktop.

In this chapter:

- [About the Physics Builder](#)
- [Overview of the Manual](#)

About the Physics Builder

In this section:

- [What Can You Do With the Physics Builder?](#)
- [Where Do I Access the Documentation and Application Libraries?](#)

What Can You Do With the Physics Builder?

The Physics Builder is a graphical programming environment where application experts can design tailored physics interfaces through an interactive desktop environment and without the need for coding.

With the Physics Builder you can deploy the tailored physics interfaces to create your own products and custom physics interfaces for specific applications.

The workflow for creating new physics interfaces is similar to creating a multiphysics model except that the result is a new user interface rather than a new model.

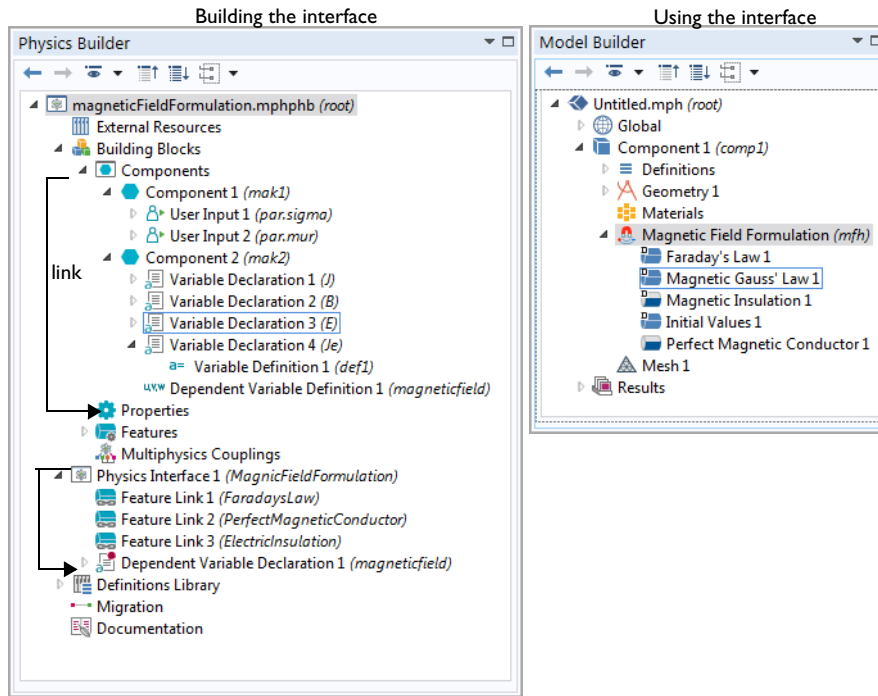


Figure 1-1: The definition of a new physics interface “Magnetic Field Formulation”: in the Physics Builder (left) and the result in the Model Builder (right).

The Physics Builder window contains a tree that represents a physics interface design project. Such a project can define anything from a single physics interface to an entire product with a collection of physics interfaces.

The following chapters describe the tools that you use in the Physics Builder and provide detailed examples of how to create custom physics.

Where Do I Access the Documentation and Application Libraries?

A number of internet resources have more information about COMSOL, including licensing and technical information. The electronic documentation, topic-based (or

context-based) help, and the application libraries are all accessed through the COMSOL Desktop.



If you are reading the documentation as a PDF file on your computer, the [blue links](#) do not work to open an application or content referenced in a different guide. However, if you are using the Help system in COMSOL Multiphysics, these links work to open other modules (as long as you have a license), application examples, and documentation sets.

CONTACTING COMSOL BY EMAIL

For general product information, contact COMSOL at info@comsol.com.

To receive technical support from COMSOL for the COMSOL products, please contact your local COMSOL representative or send your questions to support@comsol.com. An automatic notification and a case number are sent to you by email.

COMSOL ONLINE RESOURCES

COMSOL website	www.comsol.com
Contact COMSOL	www.comsol.com/contact
Support Center	www.comsol.com/support
Product Download	www.comsol.com/product-download
Product Updates	www.comsol.com/support/updates
COMSOL Blog	www.comsol.com/blogs
Discussion Forum	www.comsol.com/community
Events	www.comsol.com/events
COMSOL Video Gallery	www.comsol.com/video
Support Knowledge Base	www.comsol.com/support/knowledgebase

Overview of the Manual

This *Physics Builder Manual* contains information that helps you to get started with creating custom physics using the Physics Builder in the COMSOL Multiphysics product. The information in this guide is specific to this functionality. Instructions on how to use COMSOL in general are included with the *COMSOL Multiphysics Reference Manual*.



As detailed in the section [Where Do I Access the Documentation and Application Libraries?](#) This information can also be searched from the COMSOL Multiphysics software **Help** system.

TABLE OF CONTENTS AND INDEX

To help you navigate through this guide, see the [Contents](#) and [Index](#).

DESIGN

The [Physics Builder Design](#) chapter has an overview of the tools available and includes information about [Designing the GUI Layout](#), [Entering Names and Expressions](#), [Using Coordinate Systems](#), [Specifying Selections](#), and [The Physics Builder Manager](#).

TOOLS

The [Physics Builder Tools](#) chapter has a description of each of the tools in the Physics Builder that allow you to create custom physics interfaces for specific application.

EXAMPLE

The [Examples of Custom Physics](#) chapter provide two examples to show how to create custom physics interfaces: [The Thermoelectric Effect](#) and [The Schrödinger Equation](#).

Physics Builder Design

The information in this chapter is useful at various stages of the design of the physics interfaces.

In this chapter:

- [Overview of the Physics Builder](#)
- [Designing the GUI Layout](#)
- [Entering Names and Expressions](#)
- [Using Coordinate Systems](#)
- [The Physics Builder Manager](#)

Overview of the Physics Builder

The Physics Builder is a graphical programming environment where you can design custom physics interfaces using an interactive desktop environment without the need for coding.


In this section:

- [Creating a New Physics Builder File](#)
- [The Physics Builder Window](#)
- [Saving and Opening Custom Physics Interfaces](#)
- [The Physics Builder Manager](#)



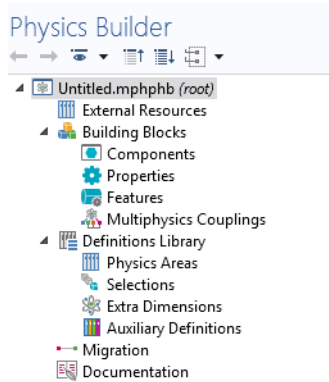
This manual is a companion to the *COMSOL Multiphysics Reference Manual*, where extensive information is available for some of the concepts and features used in the Physics Builder. See [Where Do I Access the Documentation and Application Libraries?](#) to learn how to use topic-based help in COMSOL Multiphysics.

Creating a New Physics Builder File

- 1 Open COMSOL.
- 2 Open the **Preferences** dialog box. Select it from **File>Preferences** (Windows users) or **Options>Preferences** (Mac and Linux users).
- 3 Click **Physics Builder** and select the **Enable Physics Builder** check box. Click **OK**.
- 4 Select **New** from the **File** menu.
- 5 On the **New** page under **Physics**, click the **Physics Builder** button () to open the Physics Builder.

The COMSOL Desktop then contains a **Physics Builder** window with a tree similar to the **Model Builder**.

From the main Physics Builder toolbar or from the **Windows** menu (Linux and Mac), you can open [The Physics Builder Manager](#) where you can administer testing of Physics Builder files and deployment of entire Physics Builder packages.



The **Physics Builder** window displays the tree containing the sequence of all physics and building blocks within a file.

- To add new functionality, right-click a node in the tree and choose a functionality from the context menu or click a corresponding button in the Physics Builder toolbars.
- It is only possible to add new physics interfaces to the root of the tree. Each new node represents a new physics interface that can be chosen from the Model Wizard when the interface is complete.

THE PHYSICS BUILDER BRANCHES AND SUBBRANCHES

External Resources

The [External Resources](#) branch is useful to avoid reimplementing features, properties or components. [Import](#) previously created items stored in a different builder file. All items that you implement under the Building Blocks branch in a builder file can be used by any other builder file that imports it.

Building Blocks

Use the [Building Blocks](#) branch to create a library of [Components](#), [Properties](#), [Features](#), and [Multiphysics Couplings](#) that you can build physics interfaces (including multiphysics interfaces). The [Physics Builder Tools](#) chapter describes the features and subfeatures available in detail.

Definitions Library

The [Definitions Library](#) contains definitions of material property groups, physical quantities, and other definitions that are used by but are not part of a physics interface. There are these subbranches: **Physics Areas**, **Selections**, **Extra Dimensions**, and **Auxiliary Definitions**.

Migration


[Migration](#), or backward compatibility, has to be considered in situations when you make changes to your physics interface design but still want users of the interface to use COMSOL model files created in the old [Version](#) of the interface.

Documentation

The need for internal [Documentation](#) (comments about implementation and for simplifying extending and maintaining the implementation) and external documentation (user documentation and context help) varies. The Physics Builder includes tools for creating documentation for both internal and external documentation. See [Introduction to Comments and Documentation](#).

The Physics Builder Manager

Use [The Physics Builder Manager](#) to manage testing, compilation, and comparison of your Physics Builder files.

To open this window, click **Physics Builder Manager** () on the main toolbar (Windows) or, from the main menu, select **Windows>Physics Builder Manager** (Linux, Mac). On the toolbar, click again to close the window.

Saving and Opening Custom Physics Interfaces

Saving Physics Builder files works in the same way as ordinary model files (*.mph) except that it has the extension mphphb. Opening a file is also similar to opening MPH-files, but select **Physics Builder File (*.mphphb)** from the list of file types in the **Open** dialog box.



- See [The Physics Builder Manager](#) for more information about organizing development files.
- [Saving COMSOL Files](#) in the *COMSOL Multiphysics Reference Manual*

Designing the GUI Layout

The design of the GUI layout of a feature or a property is an important and sometimes complex task. You often have to compromise between a simple layout and flexibility in the functionality. Each [User Input](#) node often represent a GUI component (or *widget*), for example fields, combo boxes, and tables. Based on the declaration of a user input, there is often only one possible choice of GUI component. In situations where there are several possible choices, you have the option to choose. You always find such choices under the **GUI Options** section of a **User Input** node.

In contrast to user inputs, which controls what GUI components you see, the [User Input Group](#) node controls when and where to display the GUI components. As an example, the user input group can list the user inputs you want to see under a specific section. This is an option in the **GUI Options** section of a **User Input Group**.

In this section:

- [User Inputs and GUI Components](#)
- [User Input Group GUI Options](#)



Use this section in combination with the features described in the [User Inputs](#) section.

User Inputs and GUI Components

The settings under the **Declaration** section often determines what GUI component you see when the user input is visible.



Use this section in combination with the features described in the [User Inputs](#) section.

SINGLE-ARRAY INPUTS

The table below summarizes the behavior for single-array inputs (option **Array type** set to **Single**).

DIMENSION	ALLOWED VALUES	GUI COMPONENT
Scalar or 1×1	Any	field (text box)
Scalar or 1×1	From list	Combo box
Vector (3x1)	Not applicable	Table with 1–3 rows, depending on the option Vector component to display .
Matrix (3x3)	Not applicable	Table with 1–3 rows and 1–3 columns, depending on the option Matrix component to display . You also get a combo box for matrix symmetry.
Boolean	Not applicable	Check box. Note that there is a special node for creating Boolean inputs.
Custom	Not applicable	Table with rows and columns representing the specified dimension. If it represents a square matrix, you can specify a symmetry with the option Matrix symmetry for square matrix .
Changeable	Not applicable	Single column table with the possibility to add rows.

Depending on the dimension of the input, you get different options in the **GUI Options** section. You find the available options below:

- **Hide user input in GUI when inactive.** The logic controlling the user input determines that it is inactive, the input’s GUI component disappears from the layout. This is not necessary if the user input is a member of a user input group that can disappear.
- **Show no description.** Removes the label above the GUI component.
- **Show no symbol.** Removes the symbol to the left of the GUI component.
- **Add divider above the user input.** Places a horizontal line above the GUI component, possibly with a descriptive text.
- **Show no coordinate labels.** For spatial vectors, by default you get the coordinate labels in the left-most column. Selecting this option removes that column.
- **Vector components to display.** Controls what components of a spatial vector you want to display in non-3D geometries. You can choose between **All**, **In-plane**, and **Out-of-plane**.
- **Matrix components to display.** Same as above but for spatial matrices.
- **Matrix symmetry for square matrix.** For non-spatial, square matrices you can force a matrix symmetry with this option. The choices are **Diagonal**, **Symmetric**, **Anisotropic**,

and **Symmetric, fixed diagonal**, and they control the cells that the user can edit. For **Symmetric, fixed diagonal**, the diagonal is fixed to the default values, and the user can only edit the off-diagonal elements.

DOUBLE-ARRAY INPUTS

Double-array inputs are far more complex to design GUI components for, and some combinations are not supported. The table below summarizes the behavior for the supported double-array inputs (option **Array type** set to **Double**).

OUTER DIM.	INNER DIM.	ALLOWED VALUES	GUI COMPONENT
Vector (3x1)	Scalar or x	Any	Behaves as a single-array vector
Vector (3x1)	Scalar or x	From list	Several combo boxes when the input is a member to a special input group, see User Input Group GUI Options . Otherwise, behaves as a single-array vector with restrictions what you can enter in the table cells.
Matrix (3x3)	Scalar or x	Any	Behaves as a single-array matrix
Vector (3x1)	Boolean	Not applicable	Several check boxes when the input is a member to a special input group, see User Input Group GUI Options . Otherwise, behaves as a single-array vector with restrictions what you can enter in the table cells.



For double-array inputs, you might get an error when you try to use an unsupported combination. In other situations, especially when the inner dimension is a scalar, you can get a component, but with an unpractical behavior. For example, when the outer dimension is fixed but nonscalar, the inner dimension is scalar, and **Allowed values** is set to **From list**. Then the input behaves as the single-array version, but with restrictions on what you can enter in the table cells.

There are fewer GUI options for double-array inputs, but those supported are identical to the options for single-array inputs.



User Input

User Input Group GUI Options

You can use the [User Input Group](#) for two main purposes: controlling GUI layout and putting the same activation conditions on several user inputs. The latter is usually a consequence when implementing the first. Grouping of user inputs also makes it possible to define the section name and to create help contents for the section. The **GUI layout** option under the **GUI Options** section controls the behavior of a user input group. The available layouts are **Group members below each other** (the default), **Group members placed in a stack**, **Create a widget for each vector component**, **Radio buttons from first user input**, **others interleaved**, **Group members define columns in table**, or **Group members define a section**.



Use this section in combination with the features described in the [User Input Group](#) section.

GROUP MEMBERS BELOW EACH OTHER

The GUI components that each group member represents appear below each other. The member can be another group, so the entire layout of that group gets a spot in this sequence. The figure below shows a schematic drawing of this layout.

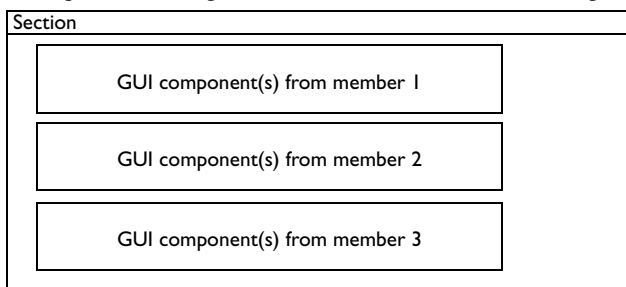


Figure 2-1: A schematic of the layout of the option “Group members below each other”.

If any of the user inputs or user input groups has the option **Hide user input in GUI when inactive** selected, it gets hidden when inactive. It is still present, and its presence can be noted because it occupies a small empty space in the layout. If you only have one hidden member like this, you hardly notice it, but if there are several such hidden members in a row, you get a clearly visible empty space. You should then use the option **Group members placed in a stack** (see below).

GROUP MEMBERS PLACED IN A STACK

The GUI components of each member are placed in separate sublayouts, called cards. Each card can appear and disappear as a unit, giving the effect that a part of the layout changes instantly. The activation condition on each member controls when its card appears or disappears. Each card can contain several GUI components and other cards depending on the type of member it corresponds to. In this way, you can create an advanced nested dynamic GUI. See below for a schematic drawing of this type of layout.

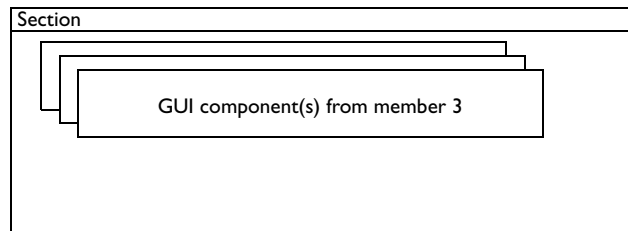


Figure 2-2: A schematic of the layout of the option “Group members placed in a stack”.

GROUP MEMBERS DEFINE A SECTION

The simplest and one of the most important GUI layouts is the section layout. You use it when you want specify what members that belong to a certain section. You specify the title of the section in the **Description** field.



The recommended way of creating a section is to use the **Section** node instead of the **User Input Group** node.

If you do not specify a section, there is a default section that can be good enough for simple layouts. There are several situations when the default section is never generated:

- When you want more than one section, you must specify all sections.
- When you have at least one **Constraint** node in your feature. The constraint usually adds a special section for weak constraints and constraint type selection, so you must specify all other sections as a section group.

As rule of thumb, always add a section if you do not see the user inputs you expect.



The section that constraint nodes usually adds is not always shown. You must show advanced physics options to see it.

CREATE A WIDGET FOR EACH VECTOR COMPONENT

You can use this layout if you wish to place a GUI component sequentially for each component of a vector-valued user input. The user input can either be a single-array vector or a double-array vector with a scalar or Boolean inner type.

A typical example is if you want to activate each vector component value with a check box. Then you create one double-array user input with the outer dimension set to vector and the inner dimension set to Boolean, and one single-array user input as a vector. Put both these user inputs as member to a group using this layout, and you get a layout like the screenshot below.

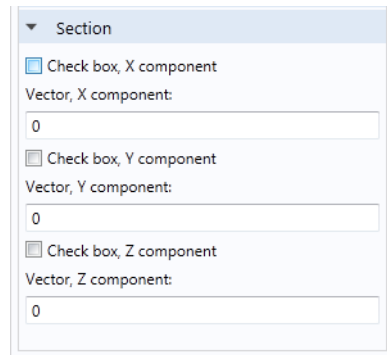


Figure 2-3: A screen shot of a window created with the layout option “Create a widget for each vector component.”

RADIO BUTTONS FROM FIRST USER INPUT, OTHERS INTERLEAVED

Use this option when you want two or more radio buttons (option buttons) that control the visibility of other user inputs or groups. The first user input must have a set of valid values, each one representing one radio button. The number of group members except the first one has to be equal to the number of allowed values in the first user input. Similar to the previous GUI layout, you get the GUI components of a group member after each radio button. It is also common that you activate the group members depending on the value of the radio-button input. See [Figure 2-4](#) that displays an example with two radio buttons. It needs three user inputs, where the first one has two allowed values.

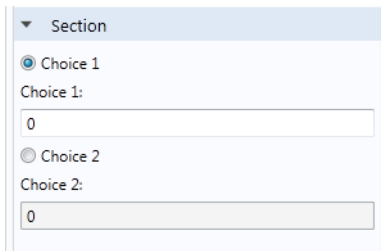


Figure 2-4: A screen shot of a window created with the layout option “Radio buttons from first user input, others interleaved.”

GROUP MEMBERS DEFINE COLUMNS IN A TABLE

Use this option when you want to combine several user inputs into a table GUI component, where each user input represents a column in the table. This requires that the user inputs are vectors and have the same dimension. In the **Table height** field you set the height of the table in pixels. You can control the behavior of the table through the check boxes listed in the table below:

CHECK BOX	DESCRIPTION
Automatically add new rows	Select if you always want an empty row below the entered ones
Rows can be added	Select to enable adding of rows
Rows can be deleted	Select to enable deleting of rows
Rows can be moved up and down	Select to enable row content to be movable

CHECK BOX	DESCRIPTION
Table data can be saved to file and loaded	Select to add toolbar buttons for saving and loading table content
Table can be cleared	Select to add a Clear button under a table to clear all data in that table.

The table columns get their headers from each user input if the **Table headers** list has the option **Use user input descriptions**. Choose **Specify** to enter them manually in the table that pops up below the list. The last table controls the settings for each column, where you specify the column settings in the corresponding row. The table below summarizes the available options.

OPTION	DESCRIPTION
Widths	The initial width of the column
Editable	Selected means that the user can edit the column
Variable	The column must represent unique and valid variable names
Expression	The values must be an expression without syntax errors
Synchronized	You get fields below the table for easier typing
New row	Control the default value for new rows, either when automatically added or when added through a toolbar button. The default array contains the initial values that the table is populated with when a feature is created. The value in the New row column is only used as default for new rows (representing new array elements in the user inputs).

ATTACH ACTIVATION ON GROUP MEMBERS

Selecting this option means that all activation conditions under this group will be attached to the list of activation conditions for all members. The attachment will be done when using the interface in the Model Builder and can be used to do modifications of existing user input activations that you either cannot access or want to do selectively for a certain feature sharing a component with other features.



User Input Group

Entering Names and Expressions

In this section:

- [Entering Names](#)
- [Using Customized Names and Descriptions](#)
- [Entering Names of Operators and Functions](#)
- [Adding a Delimiter to a String](#)
- [Tensor Parser](#)
-



Use this section in combination with the features described in the [Physics Builder Tools](#) chapter.

Entering Names

All variable names that you write in an expression are first assumed to be a variable defined by the physics interface, which means that it has a physics interface scope. If no variable is found with that scope, it checks the component scope and finally the root scope. If you want to access a variable in the root scope, but you are unsure if it exists in any other scope, enter the variable fully scoped, for example, `root.lambd` to access the eigenvalues from the solver.

You might also want to access the value of a user input in your equations without adding it as a variable. The syntax for this is to add `par.` before the input parameter name. For example, to access the input parameter `sigma` in an expression, type `par.sigma`. The `par` prefix is part of a name generation syntax that the builder interprets. This syntax is built up by a sequence of dot-separated items, where each position has a special meaning. The full syntax description can be defined by the following rule

$$[<prefix>].<identifier>.[<input>]^*.[<integer>]^*$$

All items within brackets mean that you do not have to specify them, and in some cases a default is used instead. An asterisk (*) means that you can write zero or several items. The `par` prefix in the mentioned above, is an example when the identifier position is a user input, and the value of that user input replaces the entire sequence. There are

other similar reserved prefixes for accessing different scopes and specifying operators. You find the complete list in the following table:

PREFIX	DESCRIPTION	EXAMPLE
phys	Replaced by the physics interface scope. Under a coupling feature the phys prefix can be appended with a coupling type; see Generic Multiphysics Coupling	phys.A => root.comp1.es.A
comp	Replaced by the model component scope.	comp.u => root.comp1.u
root	Used as is to define root scope.	root.h => root.h
coord	Identifies the coordinate and is used together with number-dot. Can be prefixed by s, m, g, or M corresponding to the coordinate in the spatial, material, geometry, or mesh frame, respectively.	coord.1 => X g.coord.2 => Yg
item	Replaced by a scope that represents the full path to the feature or property.	item.V0 => root.comp1.es.gnd1.V0
parent	Replaced by a scope that represents the full path of the parent to the feature or property.	parent.V0 => root.comp1.es.V0
dep	The subsequent identifier is a dependent variable, and the true name replaces the entire sequence.	dep.u => root.comp1.u2
order	The subsequent identifier is a dependent variable, and shape order of the variable replaces the entire sequence.	order.u => 2
par	The subsequent identifier is an input, and the value of that input replaces the entire sequence.	par.sigma => 12[S/m]
mat	The subsequent identifier is a material property, and the material property value, either from the material or the user defined, replaces the sequence.	mat.rho => root.comp1.mat2.def. rho(root.comp1.T)
arg	The subsequent identifier is a component parameter. The parser substitutes the argument with the parsed result of the value.	arg.C => phys.C2 => root.comp1.ht.C2

PREFIX	DESCRIPTION	EXAMPLE
map	The subsequent identifier is an extrusion operator of a pair or a periodic condition. map.nsign is a variable for the normal sign, defined as 1 on all boundaries. To change the normal direction on some boundaries, redefine this variable to be -1 to flip the normal direction.	map.src2dst(c1) => root.comp1.src2dst_p1(c1)
rot	The subsequent identifier defines a rotation of a vector or matrix variable in a periodic condition.	rot.src2dst(A) => {cos(30)*Ax+sin(30)*Ay, sin(30)*Ax-cos(30)*Ay}
minput	The subsequent identifier is a valid model input variable name, and the value of the model input parameter replaces the entire sequence.	minput.T => root.comp1.ht.T
entity	The subsequent identifier is a valid get method from the current feature or property. Valid methods are only those without arguments and that return a string value.	entity.tag => init1 entity.name => Initial value 1
loop	Modifies the variable name so it represents a unique name for the current pass in a loop.	loop.D => root.comp1.di.D_c1
dev	Replaced by the current device scope.	dev.v => root.comp1.cir.R1_v
sys	Usually replaced with the scope of the coordinate system currently selected in the Coordinate system list in the Settings window of the feature instance. In other cases it can represent more complex expressions; see The Input Base Vector System for more details.	sys.T => sys2.T

If the prefix is left out, it is assumed to be phys for variable names, but not for dependent variables, operators, and functions (see below). After the identifier there can be a trailing sequence of integers. This sequence represents indices of a tensor element. Assume that there is a 3-by-3 tensor A with physics interface scope, and that it is used in a 2D axisymmetric model where the coordinate names are r , ϕ , and z .
If you type

A.1.2

in a builder expression, it becomes

Arphi

in the 2D axisymmetric model. The standard naming convention for components of a vector or matrix is a base name concatenated with the coordinate names. You can override this naming convention using the [Component Settings](#) node.

Dependent variables are treated differently. Firstly, they always have component scope, so unscoped names get this scope. Secondly, the user can change their names, so you always specify them by their default name. The physics scope lookup has precedence over the default-name lookup of dependent variables, so if you want to use a dependent variable that has the same default name as the name of a variable, you must use the `dep` prefix.

Using Customized Names and Descriptions

In the [Component Settings](#) node, you can define custom names and descriptions by selecting different options in the **Create components by** list. The first option, **Appending coordinates to the name**, is the default behavior for spatial tensors that concatenate the tensor name with the coordinate name for each tensor component:

Axy

The option **Appending indices to the name**, concatenate the tensor name with the tensor index:

A12

This is the default for non-spatial tensors. Use the option **Specifying a template**, if you have a certain naming convention for the i :th component. For example, assume that you want to use the following names and descriptions for a velocity vector:

NAME	DESCRIPTION
x_vel	x-velocity
y_vel	y-velocity
z_vel	z-velocity

Then you specify the following template for the variable name

```
str.append(coord.i, _vel)
```

and for the description

```
#coord.i#-velocity
```

If you have a tensor with up to 4-indices, use the identifiers *j*, *m*, and *n* to access the other indices.

It is also possible to concatenate parts with `str.append` operator. The operator appends all its argument to generate the final component name. Assume that a feature has a user input called `Port` that has the value 2. The following template

```
str.append(phys.R,par.Port,par.Port)
```

then generates the following component name (`root.comp1.ph` is the physics scope)

```
root.comp1.ph.R22
```

The final option is **Specifying each component separately**. Here you type the name and description for each component in the table below the list. You can use the dot (`.`) and hash (`#`) symbols to use the coordinate names. You can implement the example above with the following component settings:

COMPONENT NAMES	COMPONENT DESCRIPTIONS
<code>str.append(coord.1,_vel)</code>	<code>#coord.1#-velocity</code>
<code>str.append(coord.2,_vel)</code>	<code>#coord.2#-velocity</code>
<code>str.append(coord.3,_vel)</code>	<code>#coord.3#-velocity</code>



Components

Entering Names of Operators and Functions

When you want to enter a name to an operator or function almost the same rules apply as for variable names. The only difference is the default scope. For variable names, the default scope is always the physics scope, represented by the `phys` prefix. When you declare a new operator or function, the default prefix is also `phys`, but not when you use the operator or function in an expression. Then the default is the `comp` prefix, which is interpreted as component scope. The reason is simply that it is most common that you declare new operators with physics scope, but not when you use a function. Then you often refer to functions that are unscoped (for example, `sin`, `cos`, `exp`, `gradient`, and `normalize`). In the Model Builder, all unscoped names are first

interpreted using component scope, then root scope, so it is possible to change the meaning of the function name `sin` if you want.



Operators and Functions

Adding a Delimiter to a String

The `str.delimited` operator creates a string by concatenating the arguments separated by the delimiter. You can use this operator in expressions in Variable Definitions, Components Settings, and so on. The delimiter can be any string, while the arguments can be string literals, variables, user inputs, or other commands (such as `entity.tag`). In the case of vector or matrix variables, each component is considered as a separate argument. The `str.delimited` operator has the following syntax:

```
str.delimited(<delimiter>, <arguments>...)
```



Use this section in combination with the features described in the [Physics Builder Tools](#) chapter.

Tensor Parser

All **Expression** fields supports tensor variables and operators for tensors. If you, for example, want the cross product between two vectors, simply type

$$A \times B$$

directly in the **Expression** field. The symbol for the cross product is among the standard mathematical symbols defined by the Unicode standard. Other special symbols used by expressions are the (inner) dot product, $A \cdot B$, and the nabla operator, ∇A . The system font must support the special symbols to display them properly; otherwise, the expression might not look correct. It is always possible to copy-paste them from an editor that supports Unicode input or directly from a Unicode character map.

There are also some functions that you can use to perform tensor operations — for example, the transpose of a matrix or the inverse of a matrix.



Use this section in combination with the features described in the [Physics Builder Tools](#) chapter.

The following table lists the operator symbols and operations that the tensor parser supports.

OPERATION	PRECEDENCE	EXAMPLE
Cross product	Same as multiply	$a \times b$ or <code>cross(a,b)</code>
Inner dot product	Same as multiply	$a \cdot b$ or <code>dot(a,b)</code>
Double dot product	Same as multiply	$a : b$
Gradient	Function	∇a or <code>gradient(a)</code>
Tangential gradient	Function	<code>gradientT(a)</code>
Divergence	Function	$\nabla \cdot a$ or <code>divergence(a)</code>
Curl	Function	$\nabla \times a$ or <code>curl(a)</code>
Tangential curl	Function	<code>curlT(a)</code>
Inverse of a matrix	Function	<code>inverse(a)</code>
Determinant of a matrix	Function	<code>determinant(a)</code>
Transpose of a matrix	Function	a^T or <code>transpose(a)</code>
Normalize a vector	Function	<code>normalize(a)</code>
Norm of a vector	Function	<code>norm(a)</code>
Sum over last index	Function	<code>sum(a)</code>
Deviatoric part of a tensor	Function	<code>deviatoric(a)</code>
Maximum element in a vector	Function	<code>maxElem(a)</code>

OPERATION	PRECEDENCE	EXAMPLE
Minimum element in a vector	Function	minElem(a)
Fill with variable to size	Function	array(a, {2,2}) = {{a,a},{a,a}} array(a, "3x1") = {a,a,a}
Specify elements of vector	Variable name	{1,2,3,4,5}
Specify elements of matrix	Variable name	{{11,12},{21,22}}
Expand elements to list of arguments	Variable name	Let r = {x,y,z} f(r.1..n) becomes f(x,y,z) g(r.1..2) becomes g(x,y)
Force symmetry in matrix	Function	Let M be a matrix that is symmetric, M = {{u*u, u*v, u*w}, {v*u, v*v, v*w}, {w*u, w*v, w*w}}. Symmetry cannot be detected because v*u is different than u*v by string comparison. The symmetric operator forces symmetry: symmetric(M) = {{u*u, u*v, u*w}, {u*v, v*v, v*w}, {u*w, v*w, w*w}}
Zero out in-plane components of a spatial vector	Function	Let r = {r, phi, z} in 2D axial symmetry zeroInPlane(r) = {0,0,z}
Zero out out-of-plane components of a spatial vector	Function	Let r = {r, phi, z} in 2D axial symmetry zeroOutOfPlane(r) = {r,0,z}
Multiply by volume integration factor	Function	integrand(a+b) becomes: (a+b)*2*pi*r in 2D axial symmetry (a+b)*ie1.detInvT for an infinite element domain

OPERATION	PRECEDENCE	EXAMPLE
Evaluate physical constants, global parameters, and units to numerical values. You can use this operator with conditional expressions to check if a parameter is larger than a certain value, for example.	Function	Let T0 be a global parameter set to 300K, evalConst(k_B_const*T0/e_const) becomes 0.025851997154882865 evalConst(1[μm]) becomes 1e-6 if the base unit is meter.
commaDerivative	Function	Let u be a vector-valued dependent variable with components u, v, and w and the coordinate names x, y, and z. Then commaDerivative(u) = { {ux, uy, uz}, {vx, vy, vz}, {wx, wy, wz} }

The *double dot product* is a summation over two indices:

$$\mathbf{a}:\mathbf{b} = a_{ij}b^{ij}$$

Unfortunately, there are two definitions of the double dot product, and the above is referred to the *Frobenius inner product* or the *colon product*. The other definition has flipped order for the indices in the second factor

$$\mathbf{a}:\mathbf{b} = a_{ij}b^{ji}$$

The former definition is used by the tensor parser.

The *gradient operator* can be suffixed with **s**, **m**, **g**, or **M** to specify in regard to which coordinate variables (spatial, material, geometry, or mesh frame, respectively) it should take its derivatives. Example $\nabla \cdot \mathbf{m} \cdot \mathbf{u}$ is the gradient of the variable u in the material frame.

Using Ctrl+Space to Access Expressions

Press Ctrl+Space in text fields for expressions to get access to lists of special operators, variables, prefixes, functions, and other expressions that the Physics Builder parser supports. The following list are available:

- **Operators:** Supported special mathematical operators.

- **Functions:** Various mathematical and logical functions supported by the Physics Builder.
- **Special Prefixes:** Special prefixes that can be used in the Physics Builder.
- **Coordinate Access:** Variables that are used to access coordinates.
- **Source Destination Access:** Functions and variables used in the context of a pair feature or periodic feature.
- **Entity Access:** Variables that access properties of the current model entity.
- **Coordinate System Access:** Variables that access the selected coordinate system.
- **Custom Physics Builder Functions:** Any tensor-valued functions declared under the **Auxiliary Definitions** branch also show up under the **Custom Physics Builder Functions** menu.

Using Coordinate Systems

When creating a feature or property, you can define two coordinate systems: the Base Vector System and the Input Base Vector System.

In this section:

- [The Base Vector System](#)
- [The Input Base Vector System](#)
- [Transformation Between Coordinate Systems](#)



Use this section in combination with the features described in the [Physics Builder Tools](#) chapter.

The Base Vector System

This is the system a feature declare its variables in. Typically, this only has an effect if the variable is a spatial tensor (for example, a vector with length 3). It also has an effect for weak form equations, where the base vector system can define the volume factor for the weak form integration. The most common choice is to use the coordinate system represented by the current frame used by the feature. In the **Base vector system** lists, this is the option **Frame system compatible with material type**.



[About Frames](#) in the *COMSOL Multiphysics Reference Manual*

The table below summarizes all possible options for the Base vector system list.

OPTION	DESCRIPTION
Frame system compatible with material type	Uses a coordinate system that represents the frame compatible with the selected material type for the feature.
Selected input coordinate system	This options activates a coordinate system selection list for a feature, where the user can choose between user-defined systems and a global system that corresponds to the feature's frame.
Spatial frame system	Uses the coordinate system for the spatial frame no matter what the feature's frame is.

OPTION	DESCRIPTION
Material frame system	Uses the coordinate system for the material frame.
Mesh frame system	Uses the coordinate system for the rarely used mesh frame.
Geometry frame system	Uses the coordinate system for the geometry frame.

The feature determines its frame from the **Frame type** list, which has the options **Material**, **Spatial**, or **Selectable by user**. The **Material** option corresponds to the material frame, and the **Spatial** (typically fluids) option corresponds to the spatial frame. For the **Selectable by user** option, the frame type depends on user choice or material setting during a Model Builder session.

When you select the base vector system for a feature, it acts as a default for all variables, user inputs, weak form equations, and constraints declared by the feature. If necessary, it is possible to override this default by changing the setting in the **Base vector system** list under the **Advanced** section of any of these nodes. Under the same **Advanced** section for variables, you can also set the tensor type, individual base vector system, and base vector type for each tensor index. In the **Tensor type** list, choose the type of vector: a **Normal tensor**, a **Tensor density**, or a **Tensor capacity**. Tensor densities and capacities are affected by the scaling of the unit volume during a change of base vector system. For nonscalar quantities, use the **Base vector system** column in the table to set individual base vector systems for each tensor index. In the **Type of base vector** column, set the type to **Covariant** or **Contravariant** for each index.



By default, all tensor indices are contravariant, and this setting is only important for nonscalar, spatial tensors in non-orthonormal coordinate systems.

The Input Base Vector System

This is the system used by all spatial (length 3) vector-valued and tensor-valued user inputs. The options in the **Input base vector system** list are the same as [The Base Vector System](#) list. When the settings differ between these two lists, everything a user enters for a user input, is automatically transformed to the system defined by the **Base vector system** list. The transformation matrices used by the transformation can be accessed through a special scope syntax, `sys.<variable>`. There are six variables defined by

a coordinate system that you can access using this scope. These are summarized in the table below:

VARIABLE	SYMBOL	DESCRIPTION	BASE VECTOR SYSTEM
T	T_i^j	Transformation matrix from public system to global system for contravariant tensors	i : public system j : global system
invT	$(T^{-1})_i^j$	Transformation matrix from global system to public system for contravariant tensors	i : global system j : public system
gSup	g^{ij}	Contravariant metric tensor for public system with respect to global system	i, j : public system
gSub	g_{ij}	Covariant metric tensor for public system with respect to global system	i, j : public system
detT	$ T_i^j $	Determinant of T, and also the volume of a unit cube in the public system measured in the global system	Not applicable
detInvT	$\frac{1}{ T_i^j }$	Determinant of invT, and also the volume of a unit cube in the global system measured in the public system	Not applicable

The public system of a coordinate system is the base vector system it defines and the global system is the base vector system the public system is defined with respect to. A global system is almost always also a frame system, whose base vectors represents the coordinates of a frame. For example, a rotated system performs a rotation of the base vectors of the global system to get the base vectors of the public system.

In some special situations, the global system of the selected coordinate system can differ from the global system of the base vector system used by the feature or property. In those cases, the transformation matrices include an extra transformation between the different global systems. Because the global systems also are frame systems, these extra transformations are usually called frame transformations. A frame transformation between the material frame and the spatial frame is given by the differentiation of the spatial coordinate with respect to the material coordinates or vice versa.

Transformation Between Coordinate Systems

All spatial vectors and matrices can transform as tensors when an operation involves two tensors defined in different coordinate systems. Consider the following example

$$D_n = \mathbf{n} \cdot \mathbf{D}$$

or in Einstein summation notation

$$D_n = n_i D^i$$

where subscripts indicate *covariant* indices and superscripts indicate *contravariant* indices. The type of index determines how a tensor transforms to a different coordinate system. A non-orthonormal coordinate system has two sets of base vectors, the covariant and contravariant base. A covariant tensor component use contravariant base vectors and a contravariant tensor component use covariant base vectors. For all orthonormal systems these two set of base vectors are identical. Now assume that D^i is given in a different coordinate system than n_i . To compute D_n properly, D^i first have to be transformed as a contravariant tensor

$$D^{i,x} = \frac{\partial x_i}{\partial u_j} D^{j,u}$$

where x_j is the i :th coordinate for the desired system, and u_i is the i :th coordinate for the original system. To separate tensor indices, they also include the coordinate name. If the tensor was covariant, the transformation would become

$$D_{i,x} = \frac{\partial u_j}{\partial x_i} D_{j,u}$$

These transformation are used whenever there exist several systems in an expression or variable assignment. The most common example is when you use an input coordinate system for your user inputs that differs from the base vector system in which the variables are stored. A material tensor from the material library, for example, can undergo a rotation to align its z -axis with the y -axis in the tensor variable used in the model. A coordinate system for rotation is always orthonormal, so in this case it does not matter if the tensors are covariant or contravariant.

Another situation when a variable might undergo an automatic conversion is if you try to perform a scalar dot product between to tensor of the same type — for example, two covariant tensors

$$D_n = n_i (g^{ij} D_j)$$

The expression parser performs a raise-index operation on D_j before taking the dot product. This is essentially a multiplication with the contravariant metric tensor, g^{ij} . The metric tensor is the identity matrix for all orthonormal systems.

REFERENCE

I. G. B. Arfken, H. J. Weber, *Mathematical Methods for Physicists*, Academic Press, 1995.


Specifying Selections

Selection Section Settings

Specifying the selection where a certain variable definition is valid works in the same way for all types of definitions throughout the Physics Builder. You find these settings under the **Selection** section of all nodes that support a selection.

It is not possible to give an absolute selection, because you do not know enough about the geometry that the physics interface is used in. Instead, set up the selection relative to selections that are known in the Model Builder.

In the **Selection** list specify what selection to start from. The bullets below explain the list options, assuming that the selection belongs to a variable, but this is also valid for all other types of nodes that support selections.

- **From parent.** The selection becomes identical to the selection of the feature. If the variable belongs to a property or a physics interfaces, the selection becomes identical to the selection of the physics interface.
- **Global.** The variable gets a global selection. This option can disable other settings, like shape selection, for example. A shape selection does not make sense for global selections because the only valid degree of freedom is an ODE variable.
- **From physics interface.** Only available for selection components. The selection is taken from the physics interface.
- **Source.** Only available for periodic condition features and pair features. The selection is identical to the source selection of the periodic condition or pair.
- **Destination.** Only available for periodic condition features and pair features. The selection is identical to the destination selection of the periodic condition or pair.
- **Operation.** Performs an operation between several selection components defined under the **Building Blocks** branch (). The supported operations are the same as for selections in the Model Builder; see [Visualization and Selection Tools](#) in the *COMSOL Multiphysics Reference Manual*.
- **From external resource.** Use this option to select a link to a selection definition from an external resource. To define the selection, choose an **Imported file** and a **Link**. The **Imported file** list is the list of **Import** nodes that has been defined and imported from the external resource.

- **Top level entities applicable to parent.** The selection become the top level entities applicable to the parent node (domains or boundaries, for example).
- **Operations on sibling-feature selections.** Searches for sibling features with a specific type in the list containing the current feature, or if the current entity is a property or physics interface, it searches the feature list under the physics interface. Then performs the selected operation on the selections of the found features.
- **From definitions library.** The selection refers to a selection component under the **Building Blocks** branch that defines the selection; see [Selection](#).

For all options except **Global**, you can also choose the output entity from the **Output entities** list. This list has the following options:

- **Adjacent boundaries.** The variable's selection contains the adjacent boundaries to the selection, which typically is a domain selection. If the selection is a boundary selection, this option returns the boundaries adjacent to the selection.
- **Adjacent domains.** The variable's selection contains the adjacent domains to the selection.
- **Adjacent edges.** The variable's selection contains the adjacent edges to the selection.
- **Adjacent points.** The variable's selection contains the adjacent points to the selection.
- **Mesh boundaries.** Specifies that the selection is of a special kind where the entities represents the boundaries of each mesh element in a domain selection.
- **Restricted to geometric entity types.** The selected entities undergo a filtering only including the entity types selected in the **Allowed entity types list**, such as **Exterior**, **Interior**, or **Symmetry axis** (in axial symmetry). See [Selection Terminology](#) for more details.
- **Restricted to frame type.** Restrict the use of the selected entities to the frame type selected from the **Frame type** list. A frame type can vary across the selected entities of a feature when the **Frame type** list of a feature uses the option **Selectable by user**; see [Using Coordinate Systems](#).
- **Base selection of extra dimension.** With this option, the output selection only contains the base selection.
- **Attached extra selection.** The output selection contains only the extra dimension that was attached to the production selection.

For the option **Adjacent boundaries** you get another option to restrict the output boundaries to certain conditions. Some restrictions only make sense if the original

selection (determined by the **Selection** list) is a domain selection. In the **Restrict to** list, you can choose among the following options:

- **All adjacent boundaries.** This option returns all adjacent boundaries, and this is the only option that makes sense for non-domain original selections.
- **Exterior boundaries to the domain selection.** All boundaries that only has one of the upside domain or downside domain belonging to the domain selection.
- **Interior boundaries to the domain selection.** The boundaries where the upside domain and downside domain both belong to the domain selection.
- **Exterior boundaries whose upside is in the domain selection.** Include exterior boundaries that has the upside domain in the domain selection.
- **Exterior boundaries whose downside is in the domain selection.** Include exterior boundaries that has the downside domain in the domain selection.

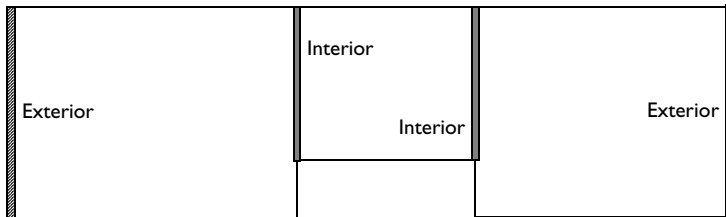


Figure 2-5: A schematic of a domain selection with highlighted exterior and interior boundaries. Note that some exterior boundaries are not highlighted.

Selection Terminology

A selection represents a set of entities on an entity dimension or geometric entity level. A boundary selection is a selection with the geometric entity level boundary. For a boundary selection in 3D the set of entities is face numbers and the entity dimension is 2. In 2D, a boundary selection has a set of edge numbers and the entity dimension is 1. There are no edge selection in 2D and 1D, and no point selection in 1D, because

there are redundant with the boundary selection. The table below summarizes the geometric entity levels and their entity dimensions.

LEVEL	DIMENSION 3D	DIMENSION 2D, 2D AXISYMMETRIC	DIMENSION 1D, 1D AXISYMMETRIC	DIMENSION 0D OR GLOBAL
Domain	3	2	1	N/A
Boundary	2	1	0	N/A
Edge	1	N/A	N/A	N/A
Point	0	0	N/A	N/A
Global	-1	-1	-1	-1

2. The geometric entity type, or just entity type, is a category in which each entity in a set belongs to. Usually, the physics interface selection defines in what entity types an entity in a set belongs to. Below is a short summary of all entity types available for features and selections.


ENTITY TYPE	APPLIES TO LEVEL(S)	DESCRIPTION
Active	All levels	Entities where the physics interface is active
Inactive	All levels	Entities where the physics interface is inactive
Geometry	All levels	Same as Active
Exterior	Boundary	Entities that are exterior to the physics interface selection
Interior	Boundary	Entities that are interior to the physics interface selection
Symmetry axis	Boundary, Edge, Point	Only for axial symmetry. The entities that lies on the symmetry axis (z-axis)
Pair	Boundary, Edge, Point	Not used by entities. For a Feature it means that the program creates an automatic version for pairs.
Source or destination	Boundary, Edge, Point	Not used by entities. For a Feature it means it only applies as a fallback feature for pairs
Exterior, neither source nor destination	Boundary	Not used by entities. For a Feature it means that it is excluded from the list of fallback features for pairs
Neither source nor destination	Edge, Point	Same as the previous row for edges and points


ENTITY TYPE	APPLIES TO LEVEL(S)	DESCRIPTION
Identity pair	Boundary, Edge, Point	For future use
Contact pair	Boundary, Edge, Point	For future use



Only a subset of these makes sense for a selection, and some are not used at all by the Physics Builder.

The Physics Builder Manager

With the **Physics Builder Manager** window () you manage testing, compilation, and comparison of your Physics Builder files. Testing is when you temporarily register one or more development files (*.mphpb) in your COMSOL session to fully test their physics interfaces in a real modeling environment. When you are satisfied with a collection of builder files, you can compile them into a builder archive.


To open this window, click **Physics Builder Manager** () on the main toolbar (Windows) or, from the main menu, select **Windows>Physics Builder Manager** (Linux, Mac). On the toolbar, click again to close the window.



In this section:

- [The Development Files](#)
- [Compiling an Archive](#)
- [Working with Builder Archives](#)
- [Searching in Archives](#)


Testing Custom Physics Interfaces

Physics Builder files can contain several physics declarations that you can access from the Model Wizard. These physics interfaces are identical to *built-in* physics interfaces (the physics interfaces shipped with COMSOL Multiphysics), with the difference that their functionality is specified by your Physics Builder file.

The COMSOL Multiphysics software searches and uses the Physics Builder files that are located under the **Development Files** branch ().



- 1 Click the **Physics Builder Manager** button () on the main Physics Builder toolbar, or (on Linux and Mac) select **Windows>Physics Builder Manager** () to open the window.

- 2 Under **Archive Browser**, right-click the **Development Files** node to add new Physics Builder files to the list.


To remove a builder file from the list, select it and then right-click it and choose **Remove Selected** () from the context menu.


COMSOL loads all physics interfaces listed in the **Development Files** node when a new session is started. If a Physics Builder file is added during a session, COMSOL loads it and updates the list of physics interfaces.

If the Physics Builder file is changed on the file system by another session, you have to manually reload it to activate these changes.

- 3 Click the **Register Development Files** toolbar button () in the **Physics Builder Manager** window to reload all physics interfaces listed in the **Development Files** branch (

The Development Files

The files listed are included in your COMSOL session. This means that your interfaces appear in the Model Wizard, so you can add them to your model and work with them like any other physics interface. You can also save model files (*.mph) that use your new interface. To add a development file, right-click the **Development Files** node () and choose **Add Builder File**. If you right-click any added development file, you can choose to remove it from the list or to open it. There is also an option to compact the archive. The compact operation removes all unnecessary data in the file to save space and simplify textual comparisons between different versions of a file.

Whenever you make changes to a builder file listed as a development files, you must click the **Register Development Files** toolbar button () to re-read all files into the current session.



If you save a model file (*.mph) that uses one of your new physics interfaces, you must make sure that the same physics interface is available when you open the file again.

Compiling an Archive

When your interface is finalized and you are ready to distribute it to others, you can compile all development files into a builder archive. Right-click the **Development Files** branch and choose **Compile to Archive Folder**. Either choose an empty folder or create

a new one. When the compilation is finished, the new archive folder can be found as a new branch under the **Archives** branch. The archive is a folder containing your source builder files, the compiled builder files, all files the builder files refer to (icons for example), the necessary Java code, and a set of language files for translation. The language files are ordinary text files where you can add a translation to all descriptions displayed for your interfaces. A language file has the following format:

```
##
# German language file
#
# Original description = Auto current calculation
deployment1.phys1.description = Automatische Stromberechnung
# Original description = Current domain
deployment1.phys1.feas1.description = Stromführender Bereich
```

All lines starting with a hash symbol (#) are comments. All files use the original description string by default, but you replace them when translating. The original description is always in the comment above the translation for reference. Do not change the tag on the left side of the equal sign. This is used by the COMSOL Multiphysics software to identify the description. The tag is a path to an entity within a builder file with the localization tag set to `deployment1` in the above example. You can change this tag in the **Root** window of the root node of a builder file. Enter the new tag in the **Localization tag** field, located in the **Physics Builder** section.


If you recompile an archive into to an existing archive, the compilation replaces all files except the language files. The compilation tries to merge the language files, by adding new descriptions, removing unused descriptions, and leave translated descriptions untouched. Unused or unreferenced files are kept in the archive.



Do not open a builder file from the **Compiled builder files** folder in an archive or add it to the development files. These files might contain file references that only work in a compressed archive (*.jar). Furthermore, they might also contain encrypted expressions that you cannot read or change.

Working with Builder Archives

Under the **Archives** node (📁) you find your compiled archives. You can add and remove archives manually from this list, but a compilation always adds the compiled archive. This list has several purposes: exporting archive as a plug-in, recompiling archives, and open the source files for editing. Once you compiled the development

files to a new archive, you should work with the source builder files in that archive, which are copies of the ones that you added to the development files. You find them under the **Source Builder Files** folder under the archive node. You can right-click any file under the **Source Builder Files** node and click  **Open Selected** to edit the file.

Right-click the archive node and choose **Compile Archive** to recompile the entire archive. This replaces all builder files under the **Compiled Builder Files**, adds new or replaces existing icons, and updates the language files as described in the previous section. To compile an individual file in an archive, right-click the node of that file and choose **Compile File**. Compiling individual files is a bit limited and sometimes it is necessary to do a full compilation of the archive to update everything properly. Here is a list of changes that require a full update:

- Adding a new **Physics Interface** node.
- Adding a new file that other files link to through the **External Resources** branch.
- Changing icons of a physics interface, adding or changing menus and menu items of a physics interface, and other changes that alters the `plugin.xml` of the archive.
- To fully update the language resources for translation.



The two latter issues are often not required to do simple testing of the functionality of the physics interface, so it is probably safe to compile a single file although you might not see the correct icon, for example.

Choose the option **Compact Archive** to reduce the file size of the builder files in the archive by removing unnecessary data. Note that this operation does change the files under the **Source Builder Files** node. A compacted file always undergoes a compact operation before future save operations. To turn this off, open the file and select the root node of the file. In the **Physics Builder** section of the **Root** window, clear the **Compact file during save** check box.

Compare the entire archive against an SVN repository by choosing **Compare with Repository**. A **Connect to SVN Repository** dialog box appears where you fill in the connection settings. In the **URL** field you enter the location in the SVN repository that contains the checked in archive (folder) with the same name as the selected archive. Also fill in the user credentials in the **Username** and **Password** fields. You can perform other comparisons between builder files in the **Archive Browser**:

- Select two archive nodes, right-click and choose **Compare Archives**.

- Select two builder file nodes from the **Development Files** node or from a **Source Builder File** node of an archive. Then right-click and choose **Compare Selected Files**.
- If there is an open builder file on the desktop, select one builder file, right-click and choose **Compare with Open Physics**. This compares the open file (local) with the selected file (remote).

COMSOL displays the result of the comparison in the **Comparison** section. A comparison displays the differences between a local file and a remote file. When comparing archives there are several such pairs of local files and remote files. The **Comparison** section contains a tree whose top nodes correspond to such a pair. The icon of the node tells if the files in a pair are equal () or not (). Expand the node to browse the differences between two files. For each selected node, you can view its attributes in the table below the tree, and the bottom table displays the currently selected pair of local and remote file.

You use the option **Export As Plug-in** to export the archive to a compressed archive (`*.jar`), when you want to include it into a COMSOL installation. The next step is to copy the compressed archive into the `plugins` folder of the COMSOL installation. To use the Run in Web Browser feature of the Application Builder for applications that use a physics interface created using the Physics Builder, the plug-ins also have to be placed in the `web/plugins` directory. Finally, you have to restart COMSOL before you can use the new plug-in.

In some system environments, the COMSOL installation folder can be write protected for ordinary users, so you cannot put the exported plug-in there without contacting the system administrator. There is an alternative location where you can put your compiled plug-ins. In your user home folder, COMSOL always creates a `.comsol` folder. Under this folder the alternative location is `<version>/archives`, where you replace `<version>` with the current version of COMSOL. Any compressed archive (with extension `.jar`) is loaded into COMSOL next time it starts.

To allow the physics interface to be used in applications running on COMSOL Server (see the *COMSOL Server Manual*), the compiled plug-ins should be placed in the `server/plugins` and `web/plugins` directories of the COMSOL Server installation directory, or in `<version>server/archives` (for example, `v52server/archives`).

Searching in Archives

The **Search in Archive** section presents a way to search through the physics builder files that you have placed under the **Archives** node in the **Archive Browser**. The kind of search

to perform is specified by the list with the options **Variables** (the default), **Node labels**, and **Override type**. The text field below the list box is where you enter the search query.






- When **Variables** is selected the search lists all nodes that declare, define, or contains a reference to the search query. The check boxes under the combo box are used to specify if the search results should include declarations, definitions, or references.
- When **Node labels** is selected the search lists all nodes whose node label begins with the characters that are typed into the search query field.
- The **Override type** option specifies that the search should list all nodes that make use of the override type that you have entered as the search query.

Physics Builder Tools

This chapter provides a description of the tools in the Physics Builder that you can use to create custom physics interfaces for specific applications.

- Building Blocks
- External Resources
- Components
- Properties
- Physics and Multiphysics Interfaces
- Features
- User Inputs
- Variables
- Equations
- Constraints
- Device Systems
- Operators and Functions
- Definitions Library
- Physics Areas
- Selections
- Extra Dimensions
- Auxiliary Definitions
- Mesh Defaults
- Study and Solver Defaults
- Result Defaults
- Migration
- Documentation
- Elements

Building Blocks




Under the root of the Physics Builder tree there is the **Building Blocks** () branch where you create a library of **Components** (), **Properties** (), **Features** (), and **Multiphysics Couplings** () that you can build physics interfaces (including multiphysics interfaces).




These items are not used in any physics interface until they are referenced from a link node (see [Component Link](#), [Property Link](#), [Feature Link](#), and [Multiphysics Couplings](#)).

Components

The **Components** branch () has the following items:


- **Component** (). A collection of user inputs, variables, equations, and constraints. Items in this branch are available to any feature or property as component links.
- **Physics Interface Component** (). A collection of nodes that define something specific that you need in several places or that group nodes together to avoid long lists of nodes under a physics interface. Items in this branch are available to any physics interface as physics interface component links.
- **Code Editor** (). Opens a text editor window for Java code, providing a possibility to enter coded methods in Java.

The components are available to all [Component Link](#) nodes. The selection components are available as references in other selection components or in any other item using selections (for example [Variable Definition](#) nodes and [Weak Form Equation](#) nodes). If the component link is in the same builder file, use **Local** in the **Link from** list of the component link node. The components in the **Components** branch of another Physics Builder file are also available, if included as an [Import](#) node under the [External Resources](#) branch ().



Components


Properties

In the **Properties** branch () you can add several **Property** nodes. The properties in this list are available to all **Property Link** nodes. If the property link is in the same builder file, use **Local** in the **Link from** list of the property link node. The properties in the **Properties** branch of another builder file are also available, if included as an **Import** node under the **External Resources** branch.



Properties


Features

In the **Features** branch () you can add several **Feature** nodes for defining physics features such as material models, boundary conditions, loads, and sources. The features in this list are available to all **Feature Link** nodes. If the feature link is in the same builder file, use **Local** in the **Link from** list of the feature link node. The features in the **Features** branch of another builder file are also available, if included as an **Import** node under the **External Resources** branch.



Features


Multiphysics Couplings

In the **Multiphysics Couplings** branch () you can add several **Coupling Feature** nodes. The coupling features in this list are available to all **Multiphysics Coupling** nodes. If the multiphysics coupling is in the same builder file, use **Local** in the **Link from** list. The coupling features in the **Multiphysics Couplings** branch of another builder file are also available if you include it as an **Import** node under the **External Resources** branch.



Physics and Multiphysics Interfaces

Code Editor

The **Code Editor** node () provides the possibility to enter coded methods in Java to perform tasks that you cannot accomplish with the nodes in the **Physics Builder** tree. A **Code Editor** node works like a **Component** node, so you include it through a **Component Link** node. Use of the code editor requires knowledge of the Java programming language and the COMSOL Java API. Note that adding Java code usually makes it much harder to find and solve problems with your physics interface, so only use it when necessary.

The following Java interfaces are supported by the **Code Editor** node:

- **VariableDefinitionProvider**. Defines expressions and selections for a set of declared variables.
- **UserInputProvider**. Defines dynamic allowed values and default values for lists.




Contact COMSOL support to learn more about the supported interfaces.

About Links

The Physics Builder include different types of links, such as Component Link, Property Link, Feature Link, and others. These links make it possible to define common building blocks and the use them in the physics interfaces that you create. In the **Settings** window for these link node, **None** is the default option in the **Link** list for new and reset links. If you use the link in a physics interface that you create, you must replace None with an actual link to a node under Building Blocks, for example; otherwise, an error occurs when running the physics interface.

Dependencies



The **Dependencies** window display information about dependencies for variables and user inputs for a number of nodes in the Physics Builder. For the selected node, these section display the declared and defined variables, variables it uses, user inputs it uses, and so on. To display the **Dependencies** window, right-click a node and choose **Dependencies** (). The **Dependencies** window then appears as a separate window next to the **Settings** window unless you close it. If not applicable, the **Dependencies** window is empty. Otherwise, it contains a group of sections that provide an overview of the

items that you have declared for the feature. The contents of these sections are read-only, and the sections are initially empty. The following sections are available:


- **Variable Declarations.** Lists all variable declarations found in the feature, excluding the ones declared through any component link. The table also provides information about the description, dimension, and physical quantity of the variables.
- **Variable Definitions.** Lists all variable definitions found in the feature, excluding the ones defined in component links. The table also specifies if the definition is an expression, parameter, or shape definition, and also displays the actual definition.
- **Necessary Variables.** Lists the variables found in expressions. This list corresponds to all the variables that must exist for this feature to work properly in all cases. Some of the variables can be declared by the feature itself, but others must be declared elsewhere.
- **Dependent Variable Definitions.** Lists the dependent variables defined by this feature with their names and physical quantity. All these definitions must have a corresponding declaration under the physics interface.
- **Necessary Property User Inputs.** Lists the user inputs read from properties found in this feature. These properties and user inputs must exist in any physics interface that uses this feature.
- **User Inputs.** Lists the added user inputs in this feature with their array type, dimension, and description, excluding any user inputs added by any component links.

These sections provide useful information about the feature — for example, how you can use a feature in different physics interfaces.

External Resources

To avoid reimplementing features, properties or components, you can use items stored in a different builder file. All items that you implement under the **Building Blocks** branch in a builder file, can be used by any other builder file that imports it. Under **External Resources** () you can add **Import** nodes () for importing other builder files.


Import

Use the **Import** node () to import other builder files. The **Settings** window contains the following section:

IMPORT FILE

In the **File** field, you enter the path to the builder file you want to import. As an alternative, you can also click **Browse** and choose a file from the system. With the **Import** button, you can re-import the file. This is necessary if you have changed the selected file from another COMSOL session.

Definitions Library

In the **Definitions Library** branch () custom material properties groups can be added and defined using available or additional material properties. You can also create physical properties and other definitions that are used but are not part of a physics interface.




There are these subbranches: [Physics Areas](#), [Selections](#), [Extra Dimensions](#), and [Auxiliary Definitions](#).

Components

In this section:

- [Creating Components](#)
- [Component](#)
- [Physics Interface Component](#)
- [Usage Condition](#)
- [Equation Display](#)
- [Component Link](#)
- [Extra Dimension Link](#)
- [1D Interval](#)
- [Multiple 1D Intervals](#)
- [2D Rectangle](#)


Creating Components

As an alternative to directly define variables, user inputs, and so forth, under a feature or property, it is possible to create a collection of such items. This collection is a **Component** (), which is added under [Building Blocks](#) () to the **Components** () branch in the Physics Builder tree.


It is convenient to use a Component when you want to reuse user inputs, for example, in several different features. Grouping variables into components can also give a better overview if you have a feature containing a lot of variables. You can, for example, collect all user inputs and groups used in a section, and use a [Component Link](#) in the feature or property that needs this section. A Component can contain the same items that a [Generic Feature](#) and [Property](#) can, with a few exceptions. For example:

- [Dependent Variable Definition](#)
- [Variable Declaration](#)
- [Variable Definition](#)
- [User Input](#)
- [User Input Group](#)
- [Material Property](#)
- [Feature Input](#)
- [Weak Form Equation](#)
- [Constraint](#)

Component

Use the **Component** node () to collect nodes that define something specific that are needed in several places, or to group nodes together to avoid long lists of nodes under a feature or property.

To add a **Component**:

- On the **Home** toolbar click the **Component** button ().
- Under **Building Blocks**, right-click **Components** and add it from the context menu.

To add a wide variety of features, right-click the **Component** node or click the buttons on the **Component**, **Model**, or **Physics Interface** toolbars. The available features are described throughout this chapter.



Variables, Equations, User Inputs, Usage Condition, Equation Display, Component Link, Extra Dimension Link, Integration Over Extra Dimension, Operators and Functions, Elements, and Device Systems.

The **Settings** window has a section to specify parameters.

For information about the **Dependencies** window's information about dependencies, see [Dependencies](#).

PARAMETERS


Specify parameters by filling in the columns **Name**, **Description**, and **Default expression**. A parameter expression can be changed for each component link that uses this component.

Select the **Loop parameter** check box to activate looping over the elements of a dependent variable. Enter the **Name**, **Description**, and **Default expression** in the corresponding columns of the table.



Entering Names and Expressions

Physics Interface Component

In the **Physics Interface Component** node () you can collect nodes that define something specific that you need in several places, or to group nodes together to avoid long lists of nodes under a physics interface.

To add a **Physics Interface Component**:

- On the **Home** toolbar click the **Physics Interface Component** button ()
- Under **Building Blocks**, right-click **Components** and add it from the context menu.

To add a wide variety of features, right-click the **Physics Interface Component** node or click the buttons on the **Home** or **Physics Interface** toolbars. The available features are described throughout this chapter.



Variables, Feature Link, Property Link, Physics Interface Component Link, Multiphysics Coupling, Usage Condition, Equation Display, Mesh Defaults, Study and Solver Defaults, Result Defaults, Auxiliary Settings (Physics Interface), Menu, and Menu Item.

The **Settings** window has a section to specify parameters.

For information about the **Dependencies** window's information about dependencies, see [Dependencies](#).


PARAMETERS

Specify parameters by filling in the columns **Name**, **Description**, and **Default expression**. A parameter expression can be changed for each component link that uses this component.



Entering Names and Expressions

Usage Condition

The **Usage Condition** () node puts a condition that enables or disables its children. You can use the condition in a variety of contexts — for example, for variable definitions under a feature or for solver and mesh defaults. The kind of conditions you can use differ between contexts because some conditions cannot be evaluated in all contexts.

In general, to add a **Usage Condition** right-click a node and add it from the context menu.



Component Link nodes can exist under a **Usage Condition** node with the limitation that the target **Component** node adds no user inputs, sections, or other user input groups. If it does, error message appears.

The **Settings** window has one section. The description covers all possible conditions, but some are not visible based on the context.

USAGE CONDITION

Select a **Condition: Explicit, And condition, or Or condition**. For **And condition** and **Or condition** define a usage condition that evaluates as a Boolean operation (*and* or *or*) between other usage conditions. Add usage conditions to the **Input condition** list. For any choice, select the **Invert condition** check box to invert the entire condition.

The following settings are for an **Explicit Condition**.

Restrict to Space Dimensions

Select the **Restrict to space dimension** check box to enable a condition on the geometry dimension used by the model in the Model Builder. Add any of the following: **0D**, **3D**, **2D**, **Axial symmetry (2D)**, **1D**, and **Axial symmetry (1D)**.

Restrict to Geometric Entity Levels

Select the **Restrict to geometric entity levels** check box to enable a condition on the geometric entity level of the context, which can be the entity level of a feature. The allowed levels are **Global**, **Domain**, **Boundary**, **Edge**, and **Point**.

For results and mesh defaults, the check box is called **Restrict to entity dimensions** and has the options **Volume**, **Surface**, **Line**, **Point**, and **Global**.

Restrict to Study Types

Select the **Restrict to study types** check box to enable a condition on the study type currently solved for. This is applicable for usage conditions under **Features**, **Properties**, **Study and Solver Defaults**, and **Result Defaults**. A common example is when you want to define the result of a time derivative such as:

```
timeDerivative(A)
```

in time-dependent study types but

```
iomega*A
```

in frequency-domain study types. The most important study types are **Stationary**, **Time Dependent**, **Frequency Domain**, **Eigenfrequency**, and **Eigenvalue**. There are also other alternatives, but some of these require additional licenses or modules.



Study and Study Step Types in the *COMSOL Multiphysics Reference Manual*

User Input

This section depends on user inputs in the parent feature, parent property, or some property. Select the **User input** check box to enter the following.

Choose an option from the **Specify user input** list: **By reference**, **By name**, or **In expression**.

If the usage condition is under a feature or property, which might contain other user inputs, choose **By reference** to directly refer to any of those user inputs by in the list. Then choose the **User input** and the **User input condition**. The options available depend on the user input referred to, but the condition can either check if the **User input is active**, or if the **User input has any of certain values**, in which case enter these in the **Values** table.

Select **By name** to enter a name in the **User input** text field. Choose an option from the **User input from** list: **Containing feature or property** (the default), **Parent feature**, **Study step**, or **Another property**. For **Another property** enter the **Property** that contains the user input in the field. Also choose the **User input condition** as described above.

For usage conditions under [Study and Solver Defaults](#), [Result Defaults](#), and [Mesh Defaults](#), the **By name** option is the only way to refer to a user input. Furthermore, they can only refer to user input under a property, so there is no such choice either. Instead, there is an option to choose the type of condition in the **Condition on** list. The option **User input in property** enables the usage condition on a user input under a property. With the option **Feature is active**, the usage condition is true if there exists an active feature of a certain type. You specify the type in the **Feature type** field. Select the **Condition is not fulfilled for undefined references** check box to if you want the condition to be treated as not fulfilled instead of throwing an error if the property is undefined.

Select **In expression** as a general tool that can evaluate an expression of relations and Boolean operators that are entered in the **Condition** text field. It also supports some special functions and names, summarized in the following table:

TABLE 3-1: VALID SYNTAX IN THE CONDITION FIELD

SYNTAX	DESCRIPTION
<cond 1> && <cond 2>	Logical <i>and</i> between conditions.
<cond 1> <cond 2>	Logical <i>or</i> between conditions.
!<cond>	Logical <i>not</i> of a condition.
<value 1> == <value 2>	True if values are equal.
<value 1> != <value 2>	True if values are different.
isActive(<input>)	Active status of the given input.
contains(<array>,<value>)	True if the value exist in the given array.
{<value 1>, <value 2>, ...}	Array of values.
'<string>'	String value.
[par.]<input name>	Value from an input in the current feature or property. The par prefix is optional.
[par.]<property>.<input name>	Value from an input in property. The par prefix is optional.
[par]..<input name>	Value from an input in the parent feature. The par prefix is optional.
arg.<argument name>	Value from an argument evaluation.
entity.sdim	Returns the space dimension as an integer.
entity.edim	Returns the geometric entity level for the current physics feature as an integer.
entity.isAxisymmetric	True if the current geometry is axisymmetric (2D or 1D),
<int 1> + <int 2>*<int 3>	Simple integer expression (supports: +, -, *, /, and %).
(<cond 1> <cond 2>) && <cond 3>	Use parentheses to override precedence.
isStudyStep(<step 1>, <step 2>)	True if the current study step solved for is either a <step 1> study step or a <step 2> study step ('Stationary' or 'Transient', for example. The identifier of the study step is the same one used when creating new study steps from the external API.

There are some special rules for these expressions that differ from ordinary tensor expressions:

- The `par` prefix is the default prefix and can be omitted in some situations. An input named `par` have to be accessed with `par.par`.
- All string values have to be typed within quotation marks ('') unless they are numbers. A number within quotation marks is different from the number itself (for example, `'1' == 1` is false).
- A Boolean input can act as a condition that returns true or false, and can be used directly in logical expressions. Boolean inputs use the values 0 and 1 for false and true, respectively, so a Boolean input as a condition is equivalent to the expression `<input> == 1`.
- The operator `isActive` is only allowed in Usage Condition nodes. Using the operator in another context results in an error.
- The only allowed prefixes are `par`, `arg`, and `entity`. All other prefixes are not recognized and most likely cause an unknown input error.



Complex expressions with several inputs may result in poor performance for updates of the user interface due to long chains of event handling. A complex enable-disable logic might also confuse the user.

The **Require input is active** check box is selected by default. It is only applicable when specifying a user input to check by reference or by name, not for expressions. When selected, the activation condition is only true if the checked user input is also active as decided by its activation conditions. For expressions, you can achieve the equivalent logic using the `isActive` operator.

Select the **Invert condition** check box to invert (negate) the defined condition.

Equation Display

With the **Equation Display** (Δu) node you can enter pretty-print equations in LaTeX that show up in the **Equations** section of a physics interface or feature in the Model Builder.

To add an **Equation Display**, first add a node where it is available, for example, components, physics interfaces, multiphysics interfaces, features, or properties, then right-click the node and choose it from the context menu.



You can also add an **Equation Display** under the **Auxiliary Definitions** branch. This is the button available on the **Home** toolbar. See [Equation Display \(Auxiliary Definitions\)](#).




The **Settings** window has the following sections:

DECLARATION

If you select the **Allow named references to equation** check box, the **Name** field will be the name used to reference to this equation display from other equation displays using the `\symbref` command. If you clear this check box, the tag of the equation display will be used. Apart from named references, the automatic or given name is also used when you use equation displays in user input groups. In this case it is important that the name is not in conflict with other equation display names in the same physics feature or physics property also used in input groups. See also [References in Equation Expressions](#).

EQUATION

Enter the LaTeX-encoded expressions in the **Enter equation in LaTeX syntax** field. There are tools you can use to get help entering specific LaTeX commands.

- Press Ctrl+Space to get lists of predefined operations to choose from.
- Click the **Add Expression** () or **Replace Expression** () toolbar buttons for the same list of operations.
- Click the **Add Expression** toolbar button to concatenate expressions to the entered expression, and **Replace Expression** to overwrite.
- See a preview of the entered expression under the **Equation preview**.
- Use the **Refresh equation preview** button () to update this preview to force it to be up to date with the expression entered in the **Enter equation in LaTeX syntax** field.

References in Equation Expressions

It can be useful to reuse parts of equations or combine multiple equation expressions into one equation. This is achieved using the syntax

```
\symbref (eq)
```

in an expression. This inserts the equation expression from the equation display node with the name `eq`. This referenced node can be local (that is, defined in the same feature or property as the referee), or it can be defined under the [Definitions Library](#) branch. It is useful to access equations from the **Definitions Library** for a file that has been imported under the [External Resources](#) node. In this case, the name of the equation should be prefixed with the tag of the import node, for example:

```
\symbref(imp1.eq)
```

It is also possible to insert the name of a dependent variable into an equation expression. If the default name of a dependent variable is `u`, then use the following syntax to access the current user-defined name of that dependent variable:

```
\symbref(dep.u)
```



See [Mathematical Symbols and Special Characters](#) in the *COMSOL Multiphysics Reference Manual* for information about available LaTeX symbols and commands.



GUI OPTIONS

This section controls how the equation is displayed in the user interface. To include an image under the equation select the **Include image below equation** check box and enter a file path to an **Image file**. Select the **Exclude from equation section** check box if the equation should not be displayed in the standard **Equation** section.




This option is useful when the **Equation Display** node is created only to be displayed in a user input group.

Component Link

Use a **Component Link** () to include all items defined within a component under the **Components** branch () as if they were part of the feature containing the link; see [Components](#).

To add a **Component Link**, first add a [Component](#), then:

- On the **Component** toolbar click the **Component Link** button (.
- Under **Components** right-click **Component** and add it from the **Component** node's context menu.

The **Settings** window has the following additional section:

SOURCE COMPONENT

In the **Links from** list, choose where to look for a certain property. Available options are:


- **Building blocks.** Lets you choose among the components listed under **Components** in the **Building Blocks** branch. Choose the component from the **Link** list.
- **External resources.** Lets you choose among the components listed in an imported builder file under the **External Resources** branch. Choose the file from the **Imported file** list. The **Link** list contains all components found in the **Building Blocks>Component** branch of the selected file.

When the source component has parameters declared, these are shown in the **Component parameters** table. The value in the **Expression** column changes the parameter value for this particular instance of the link. Other links to the same source component can use a different expression. If the source component uses a loop parameter, the **Loop parameter** table appears. Enter the name of the dependent variable in the **Expression** column.



Entering Names and Expressions

Extra Dimension Link

Use an **Extra Dimension Link** to include all items defined within an extra dimension defined in the **Definitions Library**, for example, under the **Components** branch () as if they were part of the feature containing the link.

To add an **Extra Dimension Link**, first add a **Component**, then right-click **Component** and add it from the context menu's **Links** submenu.

SOURCE EXTRA DIMENSION

In the **Links from** list, choose where to look for a certain property. Available options are:

- **Definitions library.** Lets you choose among the extra dimension nodes defined under **Definitions Library>Extra Dimensions**. Choose the extra dimension node from the **Link** list.
- **External resources.** Lets you choose among the extra dimensions listed in an imported builder file under the **External Resources** branch. Choose the file from the

Imported file list. The **Link** list contains all extra dimension nodes found in the selected file.

When the source extra dimensions has parameters declared, these are shown in the **Extra dimensions parameters** table. The value in the **Expression** column changes the parameter value for this particular instance of the link. Other links to the same source extra dimensions can use a different expression. If the source extra dimensions uses a loop parameter, the **Loop parameter** table appears. Enter the name of the dependent variable in the **Expression** column.

ATTACHMENT SELECTION

Choose an option from the **Selection** list for the attachment selection to use for the linked extra dimension: **From parent** (the default), **Global**, **Operation**, **From definitions library**, **Top level entities applicable to parent**, or **Operation on sibling-feature selections**.

For any choice, except **Global**, choose the **Output entities**: **Selected entities** (the default), **Adjacent domains**, **Adjacent boundaries**, **Adjacent edges**, **Adjacent points**, **Mesh boundaries**, **Adjacent edges**, **Restricted to geometric entity types**, or **Restricted to frame type**.

- For **Adjacent boundaries**, select an option from the **Restrict to** list: **All adjacent boundaries**, **Exterior boundaries to the domain selection**, **Interior boundaries to the domain selection**, **Exterior boundaries whose up side is in the domain selection**, or **Exterior boundaries whose down side is in the domain selection**.
- For **Adjacent edges** or **Adjacent points**, select an option from the **Restrict to** list: **All adjacent entities**, **Exterior entities to the selection**, or **Interior entities to the selection**.
- For **Restricted to geometric entity types** choose the **Allowed entity types** in the table, **Interior** or **Exterior**.
- For **Restricted to frame type** choose a **Frame type**: **Material** (the default), **Mesh**, **Geometry**, or **Spatial**.

Operation

For **Operation**, choose the **Operation type**: **Union** (the default), **Intersection**, **Difference**, or **Complement**. Then define the **Input selections**, and for Difference the **Selections to subtract**. Select options from the **Output entities** list as defined above.

From definitions library

For **From definitions library**, choose an option from the **Link** list and select options from the **Output entities** list as defined above.

Operation on sibling-feature selections

For **Operation on sibling-feature selections**, choose the **Operation type**: **Union** (the default), **Intersection**, **Difference**, or **Complement**. Then define the **Input feature types**, and, for **Difference**, the **Feature types to subtract**. Select options from the **Output entities** list as defined above. Instead of writing the feature ID of the physics feature to use in the selection, you can also write a path of IDs. The first ID is then the physics ID followed by one or several feature IDs. Regular expressions are accepted to match the IDs against a pattern. If the current entity is a coupling feature, the path syntax also accepts a coupling type instead of the physics ID. The coupling type must be preceded with the multiphysics prefix (*mph.*) and uses the coupling feature's selected physics of this coupling type. Here are some examples:

- **FeatureB**: Looks for the selections of features with ID **FeatureB**.
- **InterfaceA/FeatureB/FeatureC**: Looks for the selections under **FeatureB** with ID **FeatureC**. The interface ID must be **InterfaceA**.
- **\w+/FeatureB/Feature[A-C]**: Looks for the selections under **FeatureB** with IDs that start with **Feature** and end with any of the letters A, B, or C. The interface ID can be any nonempty sequence of word characters.
- **mph.TypeA/Feature[A-C]**: Looks for selections of features under the interface of coupling type **TypeA** in a coupling feature. The features must have IDs that start with **Feature** and end with any of the letters A, B, or C.

Note that a physics feature can only match interface IDs of the physics that it belongs to. A coupling feature can only match interfaces that are part of its selected interfaces.




- [Entering Names and Expressions](#)
 - [Integration Over Extra Dimension](#)
 - [Using Extra Dimensions](#) in the *COMSOL Multiphysics Reference Manual*
-

Properties

In this section:

- [Property](#)
- [Property Link](#)

Property

A **Property** () contains user inputs and variables that are important to the entire physics interface. A property instance always exists in only one instance for a physics interface. Any variables or equations defined by a property typically inherit the selection of the physics interface, although it is possible to change this.

To add a **Property**:

- Under **Building Blocks**, right-click the **Properties** node and select it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes and select it from the context menu.

Right-click the **Property** node to add many other features from the context menu. The **Settings** window has the following sections:

IDENTIFIER


The text you write in the **Description** field is only used as the default section description. If you create one or several manual sections using the [User Input Group](#), the description in the **Description** field is unused. The **Type** is a unique string that identifies the property, which must be unique among all properties supported by a physics interface.

RESTRICTIONS

In the **Allowed space dimensions** list you can define what space dimensions this particular property can be used in. The option **Same as parent** (the default option) means that the feature supports the same space dimensions as the physics interface. If you want to control the space dimensions manually, choose **Customized** from the list. You can then select from a list of all space dimensions.

You can also impose a special restriction on study types for this property. If you try to solve a problem for a property that does not support the current study type, it does not add any contributions to the model. Choose **Customized** from the **Allowed study types** list to control the supported study types manually by selecting from a list of study types. The option **Same as parent** (the default option) means that the property supports the same study types as the physics interface.

Property Link

The **Property Link** () refers to a common definition of a [Property](#).

To add the **Property Link** node right-click a **Physics Interface** or **Multiphysics Interface** nodes and select it from the context menu. The **Settings** window has the following section:

SOURCE PROPERTY

In the **Links from** list you choose where to look for a certain property. Available options are:


- **Building blocks.** Lets you choose among the properties listed under the **Properties** branch in the **Building Blocks** branch. You choose the property from the **Link** list.
- **Built in.** Lets you choose among built-in properties that are available to the Physics Builder. In the **Package** list you choose the main resource to use properties from. For each resource, you have a list of properties in the **Link** list to choose from. You can only choose among the currently published properties.
- **External resources.** Lets you choose among the properties listed in an imported builder file under the **External Resources** branch. You choose the file from the **Imported file** list. The **Link** list contains all properties found in the **Building Blocks>Properties** branch of the selected file.

GUI Options

Select the **Include in Model Wizard** check box to include as a property section in the Model Wizard.

For information about the **Dependencies** window's information about dependencies, see [Dependencies](#).

Tensor-Valued Function

The **Tensor-Valued Function** node () adds a possibility to create functions with tensor-valued arguments and output. It is very similar to the scalar analytical function in the Model Builder.

You can also add this node under the **Auxiliary Definitions** branch. Doing so puts the function in a more global context similar to how declarations of new physical quantities work.

SETTINGS

Enter a name for the tensor-valued function in the **Function name** field. To use this function in an expression, type `phb.<function name>`. However, if the function is declared under the **Auxiliary Definitions** branch, it is available in all Physics Builder files, which means that it will be an error to declare a function that already exists. It is therefore recommended to use an extra scope level for your functions. For example, use the name of the Physics Builder archive it belongs to, so if your archive is named `myarchive`, the function name could be set to `myarchive.myfunc`. In expressions, you can then access the function with `phb.myarchive.myfunc`.

Add the arguments to the function in the **Argument** column. An argument can have an arbitrary dimension, so it depends on the actual argument passed to the function. To use a specific dimension, select the check box in the **Force dimension** column, and enter the desired dimension in the **Dimension** column (as `2x2x3`, for example, for a 2-by-2-by-3 tensor dimension). This last column is ignored when the **Force dimension** check box is cleared.


Specify the output with the **Specify output** list, which has the options **From expression** and **Specify size and template**:



- With **From expression** the output evaluates the tensor expression in the **Expression** field. The dimension of the output depends on the expression.
- Use the **Specify size and template** option when you want to control the size of the output and enter the expression of each component in the output. Such an expression should always evaluate to a scalar, and it supports the index variables i, j, k , and l , which represent the index in the output tensor currently evaluated. If an argument has the same dimension as the output it is also possible to use these in index variables to pick up an element of the argument; for example, use `arg.i.j` to get the i :th row and j :th column in the argument tensor named `arg`.

Physics and Multiphysics Interfaces

In the Model Builder you can add physics interfaces and physics features in the Model Wizard. Depending on your license, the Model Wizard can contain different physics interfaces grouped in different physics branches. The physics interfaces shipped with COMSOL Multiphysics are referred to as *built-in physics interfaces*.

With the Physics Builder you can create new physics interfaces that show up in the Model Wizard, either in an existing physics branch or in a new physics branch.

Under each **Physics Interface** and **Multiphysics Interface**, the following can be added under **Building Blocks** ():

- A feature or a link to a feature in the [Features](#) branch ().
- A property or a link to a property in the [Properties](#) branch ().

The following can also be added:

- A [Dependent Variable Declaration](#) for the variable used by the physics.



This node does not add a dependent variable to the physics interface, it just declares that it exists.

-
- A [Variable Declaration](#) with [Variable Definition](#) as child nodes, where the definitions are expressions in terms of other declared variables. A declared variable can also be made available for plotting and results evaluation (see [Variables](#)).
 - Other nodes to define default values, equation displays, and more.





In this section:

- [Creating a Physics Interface or a Multiphysics Interface](#)
- [Physics Interface](#)
- [Multiphysics Interface](#)
- [Contained Interface](#)
- [Physics Interface Component Link](#)
- [Auxiliary Settings \(Physics Interface\)](#)
- [Disable Allowed Study Types](#)

- [Menu](#)
- [Menu Item](#)

Creating a Physics Interface or a Multiphysics Interface

These steps outline how to build a new physics interface with the Physics Builder:

- 1 Define the physics by formulating domain equations, domain sources, boundary conditions, and boundary sources, which represent the available nodes in the physics interface. Depending on the physics, equations, conditions, and sources for edges and points can also be added.
- 2 Identify the physical quantity or quantities to solve for (the dependent variables).
- 3 List the parameter settings that have to be available for the entire physics interface, which are the parameters that do not depend on the selection of geometric entities (geometric entities are domains, boundaries, edges, or points). Organize similar such parameters in groups representing the properties of the physics interface.
- 4 Add a new [Physics Interface](#) or [Multiphysics Interface](#) to the tree.
- 5 Enter information for the physics interface into the **Settings** window, initially in the **Identifiers**, **Restrictions**, and **Settings** sections.
- 6 Add the dependent variable declarations for the variables to solve for.
- 7 Add the features that the physics interface requires. All features can be added directly under the physics interface. Alternatively, add them to **Building Blocks>Features** () and then add a link to the feature in the physics interface. By placing features in the **Building Blocks** branch () , these features can be reused by linking from several physics interfaces.
- 8 Add the properties to the physics interface in the same way as the features.
- 9 Test the physics interface in some of its supported space dimensions using one of the options on the **Preview** menu () on the **Physics Interface** toolbar — **Show Preview for 3D** () , for example.


This preview automatically sets up a simple modeling environment with a predefined geometry and an instance of the physics interface. Experiment with the **Settings** window for all features, the GUI logic, and investigate the generated equations in the equation view. To test a physics interface more thoroughly, use [The Physics Builder Manager](#).


- 10 Add [Study and Solver Defaults](#), [Mesh Defaults](#), [Result Defaults](#), and [Documentation](#) to make the physics interface more intuitive and easier to use.

Physics Interface

In the Model Builder you can add physics interfaces and physics features in the Model Wizard. Depending on your license, the Model Wizard can contain different physics interfaces grouped in different physics branches. The physics interfaces shipped with COMSOL are referred to as *built-in physics interfaces*.

With the Physics Builder you can create new physics interfaces that show up in the Model Wizard, either in an existing physics branch or in a new physics branch.

To add a **Physics Interface** () and create a single physics interface:

- On the **Home** or **Physics Interface** toolbar, click the **Add Physics Interface** button (), or
- Right-click the **Root** node (the top node) and choose **Physics Interface**.

You can add features to the **Physics Interface** node in these ways:


- On the **Physics Interface** toolbar, click the available buttons, or
- Right-click the **Physics Interface** node and choose features from the context menu.




For example, see [Features](#), [User Inputs](#), [Properties](#), [Mesh Defaults](#), [Study and Solver Defaults](#), [Result Defaults](#).


The **Settings** window has the following sections:

PHYSICS AREA

This section is initially collapsed. It contains a tree view of all physics areas (fluid flow, heat transfer, and so forth) and sub-areas available from built-in resources, areas defined in the current builder file, and areas defined in any imported file under the **External Resources** branch (). See [Physics Areas](#) to learn about adding physics areas.

To put a physics interface under a physics area in the tree, select the relevant physics area node and then click the **Set as Parent** button () below the tree. Or right-click the physics node and choose **Set as Parent** from the submenu. You can find the currently selected category under the **Parent area** divider.

IDENTIFIERS

The text written in the **Description** field is the text COMSOL Multiphysics displays for the physics interface in the Model Wizard. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.

Select the **Type** check box to define a unique string to identify the physics interface. The string should not conflict with other names for the physics interfaces present in the Model Wizard.

The entry in the **Default name and tag** field is used to generate the scope of all variables that the physics interface adds in the Model Builder. It also defines the prefix for the tag of all newly created physics interfaces in the Model Builder. The tag of a physics interface is only important for references to a created interface in model files for Java.




In the Model Wizard and for the physics instance in the Model Builder there is an icon displayed for the particular physics interface. **Browse** to an image file to add the **Icon** to display for the physics interface if the default icon is not applicable.

RESTRICTIONS

The **Allowed space dimensions** list specifies the geometry dimensions that the physics interface supports: **3D**, **2D**, **1D**, **Axial symmetry (2D)**, **Axial symmetry (1D)**, and **0D**. Click the **Add** button (**+**) to open the **Allowed space dimensions** list



Choose **0D** to create a physics interface with a global scope (for example, a system of ODEs). The default is to support all space dimensions except **0D**. Delete items from the list in cases where not all space dimensions are applicable.

The **Allowed study types** includes the study types that the physics interface can create equations for. The most important alternatives are **Stationary**, **Time-dependent**, **Frequency domain**, **Eigenfrequency**, and **Eigenvalue**. Click the **Add** button (**+**) to open the **Allowed study types** list to choose other study types. Some of these studies are only for specific physics interfaces. Use the **Move Up** , **Move Down** , and **Delete**  buttons under the table to organize the list.



[Study and Study Step Types](#) in the *COMSOL Multiphysics Reference Manual*

SETTINGS

From the **Top geometric entity level** list, choose the top level for the governing equations of the physics interface: **Global**, **Domain** (the default), **Boundary**, **Edge**, or

Point. Domain is the most common, which means that the top level is the same as the geometry dimension.



It is also possible to define shell and wire interfaces that have dependent variables and equations defined on entity levels lower than the geometry dimension. For a **Shell** interface choose **Boundary**. The governing equations are then defined on faces in a 3D geometry or lines (edges) in a 2D geometry.

Select a **Default frame** to choose how the physics interface behaves together with mesh deformation. Select **Spatial, Material** (the default), **Geometry frame**, or **Mesh**.



Most interfaces use either **Spatial** or **Material**. Typically a spatial frame is used for fluid mechanics and other Eulerian-based physics, whereas the material frame is used for solid mechanics and other Lagrangian-based physics. It is not recommended to use the **Mesh** frame.

Select the **Deformed mesh** check box to allow this physics interface to control frame motion, similarly to the **Moving Mesh** or **Deformed Geometry** interfaces. When this check box is selected, the **Moving Frame Domain Condition** and **Moving Frame Boundary Condition** nodes can be used in physics feature and physics properties to specify the frame motion. The frame control settings for this physics interface will also be available in study steps.

When the check box is selected,

- Select the **Moving frame**, controlled by this physics. If **Spatial** is selected, the interface will control the spatial frame and use the material frame as reference frame (similarly to the Moving Mesh interface). If **Material** is selected, the interface will control the material frame and use the geometry frame as reference frame (similarly to the Deformed Geometry interface).
- The **Geometry shape order** setting controls the order of polynomials used for representing the geometry shape in the moving frame. Select **Same as first dependent variable** (the default) or **Custom**. For **Custom**, enter an **Order**. The default is 2.

GUI OPTION

Select an option from the **Hide interface in the Model Wizard**: **No** (the default) or **Yes**. This can be useful if you want to define an interface that is only used in another multiphysics interface, and does not make sense to use as a standalone interface.

Enter a **List order weight in Model Wizard**. The default is **2**. (the higher the weight, the lower position the Model Wizard gets in the tree of physics interfaces).


OVERRIDE RULE

This section summarizes the override rules defined by all features of the interface. If two features uses different override rules, you can fill in the table with rules between override types in different override rules. See [Override Rule](#).


DEFAULT FEATURES

This section has no user input. It contains a list of all default features that you declare for the physics interface and what geometric entity level and domain type they exist on. When you create a new physics interface in the Model Builder, COMSOL Multiphysics always adds the default features in this list to the new physics interface. See [Features](#).

Multiphysics Interface

A **Multiphysics Interface** () is a combination of other physics, behaving like one single physics interface. The multiphysics interface inherits all features and properties that all the contained physics interfaces have, which are added through [Contained Interface](#) nodes. It is possible to remove features that do not work in a multiphysics context.

To add a **Multiphysics Interface** and to create a multiphysics interface that connects and add couplings between several physics interfaces:

- On the **Home** or **Physics Interface** toolbar, click the **Add Multiphysics Interface** button () , or
- Right-click the **Root** node (the top node) and choose **Multiphysics Interface**.

You can add features to the **Multiphysics Interface** node in these ways:

- On the **Physics Interface** toolbar, click the available buttons, or
- Right-click the **Multiphysics Interface** node and choose features from the context menu.



For example, see [Features](#), [User Inputs](#), [Properties](#), [Mesh Defaults](#), [Study and Solver Defaults](#), [Result Defaults](#).

The **Settings** window for the multiphysics interface is similar to that for a single physics interface but with some settings removed (because it is inherited from the contained interfaces).

See the [Physics Interface](#) node for these settings: [Physics Area](#), [Identifiers](#), and [GUI Option](#). Also see [Override Rule](#).

RESTRICTIONS

The default **Intersect space dimensions** is **Contained interfaces**, which restricts the space dimensions by intersecting the allowed space dimensions from the contained interfaces. Choose **Custom dimensions and interfaces** to add extra restrictions to the space dimensions.

From the **Allowed study types** list choose **Intersection of interface types** (the default) or **Union of interface types** to control how to combine the study-type restrictions from the contained interfaces. Select **Customized** to set the study types manually.

SETTINGS

Choose the **Default frame: Material, Spatial, Geometry frame, or Mesh**. These have the same meaning as for the [Physics Interface](#) node. The top geometric entity level for the multiphysics interface is the maximum level among the contained interfaces.


GUI OPTIONS

In addition to the settings also available for a Physics Interface node, there is an **Exclude equation display from contained interfaces** check box. By default, the equation displays of the contained physics interfaces are shown in the multiphysics interface node. If you select this check box, the equations are not shown (allowing users to define their own equations).

DEFAULT FEATURES

Similar to the [Default Features](#) section of the [Physics Interface](#) node, this section only lists the features flagged as default features for the multiphysics interface. In this list, you do not see any default features added by the [Contained Interface](#) node, only the default features added directly under the multiphysics interface.

Contained Interface

The **Contained Interface** () is actually a link node, similar to the [Feature Link](#) and [Property Link](#) nodes. The difference is that you do not choose an item from the **Building Blocks** branch but among the available physics interfaces.

Right-click the **Multiphysics Interface** node to add this feature from the context menu.

The **Settings** window has the following sections:

SOURCE INTERFACE

In the **Links from** list choose where to look for a certain interface:

- **Local interfaces** (the default). Choose from the interfaces defined in the same file. Choose the interface from the **Link** list.
- **Built in**. Choose from built-in physics interfaces available to the Physics Builder. Select the main resource (COMSOL product) from the **Package** list. For each resource, choose an interface from the **Link** list. Only the currently published interfaces are available.
- **External resources**. Choose from the interfaces found in an imported builder file under the **External Resources** branch. Choose the file from the **Imported file** list. The **Link** list contains all interfaces found in the selected file.

INTERFACE FEATURES

From the **Add default features from interface** list, choose **Default features except those below** to add the same default features as the contained interface. Choose **Customized** to select the default features manually using the buttons under the list. The list can be empty if you do not want to use any default features from this interface.

In the **Remove features** list, add features from the contained interface that you do not want in the multiphysics interface. Make sure that you do not remove a feature that you use as a default feature.



Contained Interface (Predefined Multiphysics)

Physics Interface Component Link

Use a **Physics Interface Component Link** () to include all items defined within a component under the **Building Blocks>Components** branch () as if they were part of the physics interface containing the link; see [Components](#).

To add a **Physics Interface Component Link** right-click the **Physics Interface** or **Multiphysics Interface** nodes and select it from the context menu.


The **Settings** window has the following section in addition to the sections described for [Physics Interface](#):

SOURCE COMPONENT

In the **Links from** list choose where to look for a certain component:

- **Building blocks** (the default). Lets you choose among the components listed under the **Components** branch in the **Building Blocks** branch. You choose the component from the **Link** list.
- **Built in**. Lets you choose among built-in features that are available to the Physics Builder. In the **Package** list you choose the main resource (COMSOL product) to use features from. For each resource, you have a list of features in the **Link** list to choose from. You can only choose among the currently published features.
- **External resources**. Lets you choose among the physics interface components listed in an imported builder file under the **External Resources** branch. You choose the file from the **Imported file** list. The **Link** list contains all components found in the **Building Blocks>Component** branch of the selected file.

Auxiliary Settings (Physics Interface)

A physics interface can have an **Auxiliary Settings** node () that contains a collection of settings for its physics interface that you usually do not need to change.

To add an **Auxiliary Settings** node, right-click the **Physics Interface** or **Multiphysics Interface** nodes and select it from the context menu. Also right-click the **Auxiliary Settings** node to add a [Disable Allowed Study Types](#) node.

The **Settings** window includes the following sections:

PROPERTY DEFAULTS

This section contains a table with the four columns: **Property name**, **Input name**, **Default value**, and **Read only and hidden**. The purpose of this table is to specify default values for the user inputs in properties of the physics interface. Of course the user inputs already have default values specified in their own features. So this functionality is a way to redefine those default values. A situation where this is useful is when two physics interfaces make use of the same property through a [Property Link](#) node. Then this setting makes it possible to have different default values of the user inputs defined under the property for the two physics interfaces.

- **Property name** is the type of the property.

- **Input name** is the name of the user input.
- **Default value** is the value that the user input should be set to.
- **Read only and hidden** ensures that after the value of the user input is set to the default value, it becomes hidden and its value cannot change.

PHYSICS SELECTION

Click to select either of the **Not applicable on infinite element domain** and **Not applicable on perfectly matched layer domain** check boxes to make sure that the physics interface cannot be applied on a domain of the particular type.



[Infinite Elements, Perfectly Matched Layers, and Absorbing Layers](#) in the *COMSOL Multiphysics Reference Manual*

REMOVE STANDARD FEATURES


In this section it is possible to select what standard features to remove from the context menu of the physics interface. Possible standard features are the following, which you can add by clicking the **Add** button (**+**) and selecting them from the dialog box that opens:

- Weak Constraints,
- Weak Contribution
- Pointwise Constraint
- Global Equations,
- Axial Symmetry
- Weak Contribution on Mesh Boundaries,
- Global Constraint
- Discretization
- Continuity


Remove standard features that are not applicable for a physics interface.

Select the **Only hide features from context menus** check box to only remove them from the user interface's context menu but make it possible to open existing MPH-files that include any of the removed features.

Disable Allowed Study Types

Right-click the **Auxiliary Settings** node to add a **Disable Allowed Study Types** node () to disable some allowed study types from some space dimensions, for example.

STUDY TYPES

To add study types to disable, click the **Add** button  and then select from the **Disabled study types** list in the **Add** dialog box that opens.

RESTRICTIONS


Select the **Restrict to space dimensions** check box and use the buttons under the table to add or edit the list.

Select the **User input** check box to define the **Condition on**.

- For **User input in property** (the default) enter a **Property** and a **User input**. Choose the **User input condition: User input is active** (the default) or **User input has any of certain values**. as needed enter values in the table.
- For **Feature is active**, enter a **Feature type**.

Select the **Condition is not fulfilled for undefined references** check box to if you want the condition to be treated as not fulfilled instead of throwing an error if the property is undefined.


Menu

Use a **Menu** () node to customize the context menu of the physics interface. This particular node adds a submenu.

To add a **Menu** node, right-click the **Physics Interface** or **Multiphysics Interface** nodes and add it from the context menu. Also right-click the **Menu** node to add a **Menu Item** node or another **Menu** node, which acts as a submenu.

In the **Settings** window for **Menu**, enter a **Title** of the menu in the text field.

Menu Item

Use an **Menu Item** () node to customize the context menu of the physics interface. This particular node adds a menu item that a user can select to perform a certain task.

Right-click the **Menu** node to add a **Menu Item** node. The **Settings** window has the following section:


ACTION REFERENCE

Select an **Action Type**: **Create feature** (the default) or **Internal action tag**.


Use **Create feature** to place the action to add a new feature under a different menu than the one chosen by the program. Enter the type of feature in the **Type** text field. Select the **Restrict to geometric entity levels** check box to restrict the menu item. If the feature is applicable to other entity levels than those restricted here, it displays in its default location in the context menu.

Use **Internal action tag** to add a general action given by its tag that you type in the **Action tag** text field. This option is mainly for internal use, and there is no list of available action tags.

Physics Interface — Preview

This **Physics Interface** node () is a preview of the physics interface that you have created in the Physics Builder. You can right-click this node to add and test all features that you have defined for the physics interface. The **Settings** window for the physics interface show the applicable settings and properties that you have defined for the physics interface. Typically it includes a section for selections, such as a **Domain Selection** section, and an **Equation** section.

Multiphysics Interface — Preview

This **Multiphysics Interface** node () is a preview of the multiphysics interface that you have created in the Physics Builder. You can right-click this node to add and test all features that you have defined for the multiphysics interface. The **Settings** window for the multiphysics interface show the applicable settings and properties that you have defined.

Features

A feature commonly contains most of the variables and equations that a specific physics interface requires. In the Model Builder, zero or more instances of a feature can exist as a child node to a physics interface instance or another feature instance. Except for global features, they always have a selection on a specific geometry dimension (domains, boundaries, edges, or points).



Typically the governing equations of a problem are defined for a feature at the domain level and constraints and flux conditions for features are defined at the boundary level.

In this section:

- [Generic Feature](#)
- [Domain Condition](#)
- [Boundary Condition](#)
- [Global Feature](#)
- [Domain Feature](#)
- [Boundary Feature](#)
- [Edge Feature](#)
- [Point Feature](#)
- [Pair Feature](#)
- [Contact Pair Feature](#)
- [Device Model Feature](#)
- [Moving Frame Domain Condition](#)
- [Moving Frame Boundary Condition](#)
- [Periodic Feature](#)
- [Feature Link](#)
- [Multiphysics Feature](#)
- [Multiphysics Coupling](#)
- [Generic Multiphysics Coupling](#)
- [Global Multiphysics Coupling](#)
- [Domain Multiphysics Coupling](#)
- [Boundary Multiphysics Coupling](#)
- [Edge Multiphysics Coupling](#)
- [Point Multiphysics Coupling](#)
- [Coupling Type Contribution](#)
- [Contained Feature](#)
- [Auxiliary Settings \(Feature Nodes\)](#)
- [Auxiliary Settings \(Multiphysics Couplings\)](#)
- [Geometric Nonlinearity](#)
- [Physics Symbol](#)

Generic Feature


To add a **Generic Feature** node (

- On the **Physics Interface** toolbar, select it from the **More Features** menu, or
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

For information about the **Dependencies** window's information about dependencies, see [Dependencies](#).

The **Settings** window for a **Generic Feature** has the following sections:

IDENTIFIERS

The text entered in the **Description** field is the text COMSOL displays for the feature in the Model Builder. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.

The **Type** is a unique string that identifies a feature and it must be unique among all features supported by a physics interface. The **Default name and tag** field is used to generate the tag of newly created instances in the Model Builder.

RESTRICTIONS

From the **Allowed space dimensions** list, define what space dimension this particular feature can be used with. **Same as parent** (the default) means that the feature supports the same space dimensions as the parent instance, which can either be a physics interface or another feature. Select **Customized** to control the space dimensions manually.

Special restrictions on study types can also be controlled for this feature. If you try to solve a problem for a feature that does not support the current study type, it does not add any contributions to the model. Choose **Customized** from the **Allowed study types** list to control the supported study types manually. The option **Same as parent** (the default) means that the feature supports the same study types as the parent.

SELECTION SETTINGS

Supported Geometric Entity Levels

The **Supported geometric entity levels** list specifies the level of the selection that the feature uses when adding all its contributions like variables, equations, and constraints. The choices in this list have a slightly different meaning compared to the top level you

choose in the **Top geometric entity level** list of the physics interface settings (see [Physics and Multiphysics Interfaces](#)). The choices **Same as top level (Domain condition)** and **One level below top level (Boundary condition)** are relative to the top level of the interface. If the top level of the interface is **Domain**, choosing **One level below top level (Boundary Condition)** means that the feature contains a boundary condition to the governing equations. These boundary conditions then live on faces in 3D and lines (edges) in 2D. On the other hand, if the top level is **Boundary**, choosing **One level below top level (Boundary Condition)** still means that the feature contain boundary conditions, but the conditions now live on lines (edges) in 3D and points in 2D. For such *shell interfaces*, the governing equations are defined on faces in 3D and lines (edges) in 2D. The boundary condition to a line, for example, is a point. The choices **Same as parent level** and **One level below parent level** make the feature's selection settings contain entities of the same geometric entity level or one level below (boundaries, if the parent level is domains, for example), respectively. The choices **Edge** and **Point** always refer to the geometric entities edge and point no matter what the top level is. For more details on geometric entity levels, see [Selection Terminology](#).

Finally, if your interface has **Global** as **Top geometric entity level**, the relative choices, **Same as top level (Domain condition)** and **One level below top level (Boundary condition)**, refers to the geometry dimension the global interface belongs to. If you add it to a 3D geometry, the entity levels becomes domains and faces. A global interface can also live on a global model (no geometry), and in this case it does not make sense to use anything else than the **Global** option. It is therefore recommended that you restrict the space dimensions of features that use any of the other options.

Applicable Entities

It is possible to limit the types of entities where a feature can be active. For example, a boundary condition can be limited to only interior boundaries. All other selected boundaries then get marked as *not applicable*. For more complex situations, you can use conditions on domain types to decide if a specific boundary is applicable or not.

Select a form of **Applicable entities** from the list: **From entity types** (the default) or **From sequence**.

- If **From entity types** is selected, select one or more of the options available when you click the **Add** button (**+**) add them to the list of applicable entities. **Exterior** and **Interior** boundaries are selected by default. Use the buttons under the list to organize as needed.
- If **From sequence** is selected, select a **Selection filter sequence: Locally defined** (the default) or **Imported from external resource**. Select a sequence to link to from the

Link list. Click the **Go to Source** button  as needed. If the sequence is **Imported from external resource** also select an **Imported file**.





For more details on entity types, see [Selection Terminology](#).

Override Rule

Some features cannot exist on the same selection (for example a boundary), and COMSOL Multiphysics uses override rules to decide how the selection of one feature overrides another feature's selection.

Select an **Override rule:** **Built in** (the default), **Locally defined**, or **Imported from external resource**.

- If **Built in** is selected there are four options in the **Override type** list: **Exclusive**, **Contributing**, **Override features of same type**, and **Never overridden**:
 - **Exclusive** means that the feature should override all other features except those that are of the type **Never overridden**. An exclusive feature is overridden by other exclusive features.
 - **Contributing** means that the feature should not override other features. The exception is for entity levels below the top entity level of the physics, where features of any type override the default features.
 - **Override features of same type** means that the feature only overrides other features of the same feature type.
 - **Never overridden** ensures that a feature is never overridden and does not override any other feature.
- If **Locally defined** is selected, also select a **Link** from the list. Click the **Go to Source** button  as needed.
- If **Imported from external resource** is selected, select an **Imported file** from the list. Click the **Go to Source** button  as needed. Then select a **Link** from the list.

Override Type

The **Override type** list specifies if the feature is **Exclusive** or **Contributing** to other features. In the Model Builder, an *exclusive feature* (such as constraints and fixed values) replaces all previous feature instances for intersecting selections whereas a *contributing feature* (such as loads and sources) adds to other contributing features sharing the same selection. You also select to only **Override features of the same type** or to let the feature be **Never overridden**.

COORDINATE SYSTEMS

In the **Input base vector system** list, you choose what coordinate system all user inputs and material parameters are given in. This is related to the frame type and is the coordinate systems to use for spatial vectors and matrices.

The **Base vector system** list specify the coordinate system used by the equations and variables declared by this feature. If any of these lists has the selection **Selected input coordinate system**, the user gets an option to choose coordinate system in the **Settings** window of the feature instance in the Model Builder.

The **Frame type** list specifies if the equations assumes that they live on the material or spatial frame. The options are **Material** (the default), **Spatial**, and **Selectable by user**. The latter means that the frame type can be controlled when using the feature instance. The instance then gets a **Material type** list with the options **Solid**, **Non-solid**, or **From material**. The feature uses the material frame for the solids, and the spatial frame for non-solids (typically a fluid such as a liquid or a gas).



- [Using Coordinate Systems](#)
 - [Transformation Between Coordinate Systems](#)
-

PREFERENCES

Select the **Singleton feature** check box if the physics interface only allows a single instance of the feature.

A feature instance in the Model Builder can have a section called **Model Inputs**, which shows up when you select the **Include model inputs** check box. A model input is an input argument to material parameters when they depend on a quantity (for example, if the density depends on temperature).

For features under a physics interface there is an **Add as default feature** check box. Select this check box if you want the feature to be a default feature. When you specify default features, specify the geometric entity level in the **Default geometric entity level** list. Newly created interfaces then add the feature on this level. The **Default entity types** specifies the entity types the default feature is added on.

In the **Advanced preferences** table, you can specify special options for the default feature. By default, default features have a selection over all domains (it cannot be changed), and you can neither remove or disable the feature. Select the check box columns — **Unlock selection**, **Clear selection**, **Deactivable**, and **Removable** — to alter this default behavior. The last column, **Lists order weight**, is a preference to control the

order of the default features. The standard order is domain features first, then boundary features, and so on until the point features followed by the global features. The automatically added initial value features always show up last in the list. The program uses an integer when sorting the default features. The integer depends on the geometric entity dimension and the list order weight using the following formula:


$$100 * \langle \text{entity dimension} \rangle + \langle \text{list order weight} \rangle$$

Do small adjustments (<100) to control order within an entity dimension, and large adjustments to put a default feature first or last in the list. Initial value features has a default weight of -1000 to appear last.

GUI OPTIONS



This section is only available for a **Generic Feature** node added under a **Physics Interface** or **Multiphysics Interface** node.

Select the **Hide feature from context menus** check box to remove the possibility to add new features of this type. This can be useful when a feature must be kept for backward compatibility reasons, but a user cannot add them in new models. When selected, this option also enables hiding with respect to the option **Advanced Physics Options** under the **Show** () menu in the Physics Builder. This is the only option there that represent a category in the **Category for hiding feature** list, which has the options **Always hidden** and **Advanced physics options**. Select **Always hidden** (the default) to hide it permanently, or select **Advanced physics options** to make it possible for users to show it by selecting **Advanced Physics Options** from the **Show** menu.


PARAMETERS




This section is only available for a **Generic Feature** node added under **Building Blocks>Features**.

Specify parameters by filling in the columns **Name**, **Description**, and **Default expression**.

Domain Condition


The **Domain Condition** node () defines a feature that can only exist (have selections) on the same geometric entity level as the physics.

To add a **Domain Condition**:


- On the **Physics Interface** toolbar, click the **Domain Condition** button ().
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it does not include the **Supported geometric entity levels** list.

Boundary Condition


The **Boundary Condition** node () defines a feature that can only exist (have selections) on one level below the geometric entity level of the physics.

To add a **Boundary Condition**:

- On the **Physics Interface** toolbar, click the **Boundary Condition** button ().
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it does not include the **Supported geometric entity levels** list.

Global Feature


The **Global Feature** node () defines a feature that can only exist (have selections) on the global geometric entity level (entire geometry), which means that the **Settings** window for the feature instance does not contain any selection list.

To add a **Global Feature**:

- On the **Physics Interface** toolbar, select it from the **More Features** menu.
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it do not include the **Supported geometric entity levels** list.

Domain Feature


The **Domain Feature** node () defines a feature that can only exist (have selections) on the domain geometric entity level of the current space dimension, which represents volumes in 3D, surfaces in 2D, and lines in 1D.

To add a **Domain Feature**:

- On the **Physics Interface** toolbar, select it from the **More Features** menu.
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it does not include the **Supported geometric entity levels** list.

Boundary Feature


The **Boundary Feature** node () defines a feature that can only exist (have selections) on the boundary geometric entity level of the current space dimension, which represents surfaces in 3D, lines in 2D, and points in 1D.

To add a **Boundary Feature**:

- On the **Physics Interface** toolbar, select it from the **More Features** menu.
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it does not include the **Supported geometric entity levels** list.

Edge Feature

The **Edge Feature** node () defines a feature that can only exist (have selections) on the edge geometric entity level, which represents lines in all space dimensions. This feature is normally only applicable in 3D (the default restriction), but can optionally be allowed in lower dimensions in special situations.


To add an **Edge Feature**:

- On the **Physics Interface** toolbar, select it from the **More Features** menu.

- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it does not include the **Supported geometric entity levels** list.

Point Feature


The **Point Feature** node () defines a feature that can only exist (have selections) on the point geometric entity level, which represents points in all space dimensions. This feature is normally only applicable in 3D, 2D, and 2D axial symmetry (the default restriction), but can optionally be allowed in other dimensions in special situations.

To add a **Point Feature**:

- On the **Physics Interface** toolbar, select it from the **More Features** menu.
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is almost identical to the [Generic Feature](#), except that it does not include the **Supported geometric entity levels** list.

Pair Feature

A **Pair Feature** () is a special feature that defines conditions on pairs. A pair has a source and a destination side, and a pair condition typically connects the dependent variables between them using pointwise constraints.



The most common pair condition is the continuity pair condition, which simply makes the dependent variables equal on both sides. All physics interfaces gets this pair condition by default, so you do not have to add it.

This feature is useful to define other types of pair conditions. The major difference between a pair feature and an ordinary feature lies in subnodes that define selections.

For pair features, there are two extra options in the **Selection** list of the **Selection** section; **Source** and **Destination**. With these options you can specify to use the source or destination boundaries in the condition of a pair feature; see [Specifying Selections](#) for

more information.

- To add a **Pair Feature** under the **Building Blocks** branch, **Features** node: On the **Physics Interface** toolbar, select it from the **More Features** menu, or right-click **Features** and add it from the context menu.
- To add a **Pair Feature** under the **Physics Interface** or **Multiphysics Interface**, Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

Except for a few excluded sections, the **Settings** window is similar to the [Generic Feature](#) node.

SELECTION SETTINGS

See [Selection Settings](#) for **Generic Feature**. For a pair feature, you do not have to specify the geometric entity level, because it is always boundary level. The same is true for the domain type setting, because all pairs belong to a special domain type, called **Pair**. Ordinary features that support this domain type also appear in a pair version among the physics interface's pair conditions.

All pair features also have a special rule for overriding selections. They are always exclusive to all non-pair features preceding the pair in the list. For all features, pairs and non-pairs that lie below the pair in the list, contribute with the pair condition. As a result, you cannot set the override type for pair features.


PREFERENCES

See [Preferences](#). It is not possible to add a pair feature as a default feature to a physics interface.



- [Identifiers](#) for the [Generic Feature](#) node
 - [Identity and Contact Pairs](#) in the *COMSOL Multiphysics Reference Manual*
-

Contact Pair Feature

A **Contact Pair Feature** () node is almost identical to the **Pair Feature** node. The difference is only visible for the feature instance in the Model Builder. A pair feature of

the contact pair type can only select contact pairs. See [Pair Feature](#) for more information.




Using this feature also forces the **Include geometric nonlinearity** switch to be selected in study steps, which introduces extra license requirements.

To add a **Contact Pair Feature**:

- On the **Physics Interface** toolbar, select it from the **More Features** menu.
- Under **Building Blocks**, right-click **Features** and add it from the context menu.
- Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

Device Model Feature

A **Device Model Feature** node () is a combination of a [Generic Feature](#) node and a [Device Model](#) node. It behaves identically to a feature node with additional functionality to make it work as a device model. The device model gets a type identical to the tag of the feature instance.

- To add a **Device Model Feature** under the **Building Blocks** branch, **Features** node: On the **Physics Interface** toolbar, select it from the **More Features** menu, or right-click **Features** and add it from the context menu.
- To add a **Device Model Feature** under the **Physics Interface** or **Multiphysics Interface**, Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Devices** submenu.



You can also add a [Device Feature](#) to this node. It is a special device of the type that the device model feature defines, and it is placed at the same level as the device model feature in the device system hierarchy. See [Device Systems](#) for more information.

Moving Frame Domain Condition

If the physics interface controls the deformation of the material or the spatial frame, use this node to specify how the frame is deformed on a certain selection. The physics interface controls the frame deformation if the **Deformed mesh** check box in the physics interface node is selected.

The **Moving Frame Domain Condition** node ([u,v,w](#)) allows specifying two types of frame deformation, using the **Frame motion** list box:

- When **Free** (the default) is selected, the coordinates of the moving frame in the chosen selection will be added as dependent variables for the problem. Specify a **Smoothing equation** to use to govern the motion of the coordinates, one of **Laplace** (the default), **Hyperelastic**, **Yeoh**, or **Winslow**. See to the documentation for the Moving Mesh interface for more details on each smoothing equation. Specify also an expression for the **Initial values for displacement** as a 3-components vector. The initial values must not depend on the coordinates of the moving frame (but can depend on the coordinates of the reference frame). In order for the model to be solvable, appropriate **Moving Frame Boundary Conditions** must be applied on the boundaries of **Free** domain. The **Free** frame motion is only supported for top-entity-level selections (domain conditions).
- When **Prescribed displacement** is selected, the frame motion is determined by the **Expression** entered (a 3-component vector). The expression cannot depend on the coordinates of the moving frame, but it can depend on the coordinates of the reference frame. The **Prescribed displacement** condition can be applied on selections of all entity level. For example, it can be used to prescribe the motion of boundaries or edges, in addition to domains.

In the **Selection** section, specify the selection on which the condition is to be applied. **Free** frame motion is only supported for top-entity-level selections (domain condition).

Only one entity in the entire model can be responsible for the frame motion on a certain selection. Multiple **Moving Frame Domain Condition** in the same physics applied on the same selection will cause an error when trying to solve the model.

The node supports the **Input Dependency** subnode to provide more flexibility in the setup.

Moving Frame Boundary Condition

If the physics interface controls the deformation of the material or the spatial frame, use the **Moving Frame Boundary Condition** node ([u,v,w](#)) to specify the boundary conditions for domains with Free frame motion. See the [Moving Frame Domain Condition](#) section for details. The physics interface controls the frame deformation if the **Deformed mesh** check box in the physics interface node is selected.


The available boundary conditions are:

- **Displacement** (the default): Specify the **Displacement vector** at the boundary as a 3-component vector.
- **Componentwise displacement**: Choose which components of the displacement to constrain by using the **Constrain displacement in direction 1/2/3** check boxes. Then specify the expressions for the appropriate **Displacement component 1/2/3**. Use the **Input Dependency** subnode to make these specification dependent on other user inputs.
- **Normal displacement**: Specify the expression of the normal displacement of the boundary. A positive displacement is in the direction of the normal from downside (`root.dn`).
- **Velocity** (the default): Specify the **Velocity vector** at the boundary as a 3-component vector. This condition is only applicable for **Transient** studies.
- **Componentwise velocity**: Choose which components of the velocity to constrain by using the **Constrain velocity in direction 1/2/3** check boxes. Then specify the expressions for the appropriate **Velocity component 1/2/3**. Use the **Input Dependency** subnode to make these specification dependent on other user inputs. This condition is only applicable for **Transient** studies.
- **Normal velocity**: Specify the expression of the normal displacement of the boundary. A positive displacement is in the direction of the normal from downside (`root.dn`). This condition is only applicable for **Transient** studies. The **Normal velocity** condition can also support **Boundary smoothing** by selecting the check box.

Weak constraints can be used for certain conditions, according to the value selected in the **Use weak constraint** list. If **Yes** is selected, weak constraints will be always used. If **No** is selected, weak constraints will never be used. If **From constraints settings** is selected (the default), the settings in the feature's **Constraint Settings Section** will determine if weak constraints are to be used, defaulting to **No** if the section does not exist.

Use the **Selection** section to specify the selection on which to apply the boundary condition. The selection must be adjacent to a selection with **Free** frame motion, and (except for very particular cases) boundary conditions must be specified for all **Free** domain selections in order to obtain a well-posed problem.

Periodic Feature

A **Periodic Feature** () is a special boundary condition that couples, usually by pointwise constraints, dependent variables on a source side to a destination side to create a periodic boundary condition, for example. It has similarities with the [Pair Feature](#).

The major difference between a periodic feature and an ordinary feature lies in subnodes that define selections. For periodic features, there are two extra options in the **Selection** list of the **Selection** section: **Source** and **Destination**. With these options you can specify to use the source or destination boundaries in the condition of a periodic feature; see [Specifying Selections](#) for more information.

- To add a **Periodic Feature** under the **Building Blocks** branch, **Features** node: On the **Physics Interface** toolbar, select it from the **More Features** menu, or
- To add a **Periodic Feature** under the **Physics Interface** or **Multiphysics Interface**, Right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the **Features** submenu.

The **Settings** window is similar to the [Generic Feature Settings](#) window.



For information about the settings in the [Identifiers](#) and [GUI Options](#) sections, see the [Generic Feature](#) node.

SELECTION SETTINGS

Except for a few excluded settings, the sections are described for the [Generic Feature](#) node; see [Selection Settings](#). For a periodic feature, you do not have to specify the geometric entity level because it is always boundary level. The same is true for the entity type setting because the only setting that makes sense is periodic conditions on exterior boundaries.

PREFERENCES


The **Allow simple periodic deduction of edges** check box is enabled if you have added **Edge** to the **Supported geometric entity levels** list in the **Selection Settings** section. Select that

check box to use the simple method for deduction of a periodic boundary without taking into account the rotation between source and destination.



For the other settings see [Preferences](#), for the [Generic Feature](#) node.

Feature Link

The **Feature Link** () refers to a common definition of a feature (for example, under the **Building Blocks** branch).

To add a **Feature Link** node, right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the context menu.

SOURCE FEATURE

Choose where to look for a certain feature from the **Links from** list. Select:


- **Building blocks** to choose from the features listed under the **Features** branch in the **Building Blocks** branch. You choose the feature from the **Link** list.
- **Built in** to choose among built-in features that are available to the Physics Builder. In the **Package** list you choose the main resource to use features from. For each resource, you have a list of features in the **Link** list to choose from. You can only choose among the currently published features.
- **External resources** to choose among the features found in an imported builder file under the **External Resources** branch. You choose the file from the **Imported file** list. The **Link** list contains all features found in the **Building Blocks>Features** branch of the selected file.

DEFAULT FEATURES

Select the **Add as default feature** check box to define the **Default geometric entity level**, **Default entity types**, and **Advanced preferences**.

See the [Generic Feature](#) node for the rest of the settings.

Multiphysics Feature

A **Multiphysics Feature** () is a combination of other features that behave like one feature. The multiphysics feature inherits all variables and equations that all contained features has, which you add through **Contained Feature** nodes. You can right-click the

Multiphysics Feature node to add most of the same features as for the [Multiphysics Interface](#), including the [Contained Feature](#) node.

To add a **Multiphysics Feature**, right-click the **Multiphysics Interface** node to add this from the context menu.

The **Settings** window for the multiphysics feature is similar to a feature, but with some settings removed as they are inherited from the contained features. The entire **Restrictions** section is excluded because the allowed space dimensions and allowed study types get their values from an intersection of the values in the contained features.



- [Identifiers](#) section for the [Generic Feature](#) node.
-

SETTINGS

Almost identical to the [Selection Settings](#) section of the [Generic Feature](#) node. For a multiphysics feature you cannot change the supported geometric entity levels, the domain type, or the override type. The contained features determine these settings. The entity level and domain type get their values from an intersection-like operation, and as override type you get the most strict one. The most strict override type is the first one in the **Override type** list.

PREFERENCES

Almost identical to the [Preferences](#) section of the [Generic Feature](#) node, except that you cannot change the model input setting. A multiphysics feature includes model inputs if any of the contained features does.

Multiphysics Coupling

You can add **Multiphysics Coupling** nodes () under a **Physics Interface** or **Multiphysics Interface** node and refer to a multiphysics coupling under the **Building Blocks** branch.

To add a **Multiphysics Coupling**, right-click the **Physics Interface** or **Multiphysics Interface** nodes to add this from the context menu.

This node contains settings for specifying which coupling type the physics interface should be in regard to a coupling feature:

COUPLING FEATURE LINK

In this section you specify a coupling feature that the parent physics interface can be coupled to by selecting it from the **Link** list. The coupling feature can be linked from the same file as the physics interface (by choosing the default **Building blocks** from the **Links from** list), from a built-in package (by choosing **Built in** from the **Links from** list), or from an external file (by choosing **External resources** from the **Links from** list).


MULTIPHYSICS COUPLING

From the **Coupling type** list, you specify which coupling type this physics interface is with regard to the coupling that is selected in the **Coupling Feature Link** section. The **Coupling type** list contains the coupling types that you have defined in the multiphysics coupling that this node links to.

OVERRIDE RULE SETTINGS

In this section, an override type is specified for the coupling feature by first using the **Override rule** list to determine which override rules to use — **Built in** (the default), **Locally defined**, or **Imported from external resource** — and then select a type from the **Override type** list (see [Override Rule](#) for more information). The coupling feature acts as if it were added last in the list of physics features under the physics interface, and overrides the physics features according to this override type. Note that in the coupling feature node itself an override type is also specified. That override type determines how the coupling feature should override other coupling features.

Generic Multiphysics Coupling

To add a **Generic Multiphysics Coupling** node () , under **Building Blocks**, right-click the **Multiphysics Couplings** branch node to add this from the context menu. You can right-click the **Generic Multiphysics Coupling** node to add many features.

Coupling features have mostly the same subnodes as ordinary features. However, there is one difference when working with variable expressions under a coupling feature.

Under a coupling feature there is no parent physics interface, so the normal `phys` prefix is useless (unless it is used under a [Coupling Type Contribution](#) node). Instead there is the possibility to append the coupling type to the physics scope; see the **Couplings** section below. So inside a coupling feature that couples to a physics of coupling type A, the prefix `phys.A` works in the same way as the prefix `phys` would under the physics interface of type A.

See [Generic Feature](#) for information about the **Identifiers**, **Restrictions**, **Selection Settings** (see also below), **Coordinate Systems**, and **Preferences** sections.

COUPLINGS


This section has a table that defines the names and descriptions of the coupling types that the coupling feature requires. The coupling feature is not available under the **Multiphysics** branch in the **Model Builder** unless at least one physics interface of each coupling type has been added to the same model component. To define the coupling type of a physics interface in regard to a coupling feature, see [Multiphysics Coupling](#).

The coupling feature instance in the **Model Builder** has a section with a title given by the description of the coupling feature. This section contains one list for each coupling type. The lists are used to pick the physics interfaces that the coupling feature is coupled to. The descriptions of the lists are given by the descriptions of the coupling types, defined in the second column of the table.

SELECTION SETTINGS

This section works in the same way as the same section for a [Generic Feature](#), but there is one extra thing to keep in mind when specifying the override type of a coupling feature. The override type specified under the **Selection Settings** section is only used to determine the override behavior of the coupling feature in regard to other coupling features in the **Multiphysics** branch. A coupling feature can also override the physics features of the physics interfaces that it is coupled to. However, the way in which that is done is determined by another override type, which you specify in the [Multiphysics Coupling](#) node.


Global Multiphysics Coupling

The **Global Multiphysics Coupling** node () defines a multiphysics coupling that can only exist (have selections) on the global geometric entity level (entire model), which means that the **Settings** window for the feature instance does not contain a selection list.

To add the **Global Multiphysics Coupling** node, under **Building Blocks**, right-click the **Multiphysics Couplings** branch node to add this from the context menu. You can right-click the **Global Multiphysics Coupling** node to add many features.

See [Generic Multiphysics Coupling](#) for the settings.


Domain Multiphysics Coupling

The **Domain Multiphysics Coupling** node () defines a coupling feature that can only exist (have selections) on the domain geometric entity level of the current space dimension, which represents volumes in 3D, surfaces in 2D, and lines in 1D.

To add the **Domain Multiphysics Coupling** node, under **Building Blocks**, right-click the **Multiphysics Couplings** branch node to add this from the context menu. You can right-click the **Domain Multiphysics Coupling** node to add many features.

The **Settings** window is almost identical to the [Generic Multiphysics Coupling](#) node, except that it does not include the **Supported geometric entity levels** list.


Boundary Multiphysics Coupling

The **Boundary Multiphysics Coupling** node () defines a coupling feature that can only exist (have selections) on the boundary geometric entity level of the current space dimension, which represents surfaces in 3D, lines in 2D, and points in 1D.

To add the **Boundary Multiphysics Coupling** node, under **Building Blocks**, right-click the **Multiphysics Couplings** branch node to add this from the context menu. You can right-click the **Boundary Multiphysics Coupling** node to add many features.

The **Settings** window is almost identical to the [Generic Multiphysics Coupling](#) node, except that it does not include the **Supported geometric entity levels** list.


Edge Multiphysics Coupling

The **Edge Multiphysics Coupling** node () defines a coupling feature that can only exist (have selections) on the edge geometric entity level of the current space dimension, which represents lines in all space dimensions. This feature is normally only applicable in 3D (the default restriction), but it can optionally be applicable in lower dimensions in special situations.

To add the **Edge Multiphysics Coupling** node, under **Building Blocks**, right-click the **Multiphysics Couplings** branch node to add this from the context menu. You can right-click the **Edge Multiphysics Coupling** node to add many features.

The **Settings** window is almost identical to the [Generic Multiphysics Coupling](#) node, except that it does not include the **Supported geometric entity levels** list.


Point Multiphysics Coupling

The **Point Multiphysics Coupling** node () defines a coupling feature that can only exist (have selections) on the point geometric entity level of the current space dimension, which represents points in all space dimensions. This feature is normally only applicable in 3D, 2D, and 2D axial symmetry (the default restriction), but it can optionally be applicable in other dimensions in special situations.


To add the **Point Multiphysics Coupling** node, under **Building Blocks**, right-click the **Multiphysics Couplings** branch node to add this from the context menu. You can right-click the **Point Multiphysics Coupling** node to add many features.

The **Settings** window is almost identical to the [Generic Multiphysics Coupling](#) node, except that it does not include the **Supported geometric entity levels** list.


Coupling Type Contribution

The **Coupling Type Contribution** node () , which can be added as a subnode to any of the multiphysics coupling nodes, defines a physics interface context in which you can add variable declarations, variable definitions, equations, and so forth. The contributions behave as if they were added by a physics feature under the physics interface. All contributions that do not belong to any context use the multiphysics coupling as context. The context decides the default scope of variables and function names, for example. The **Settings** window contains the following section:

RESTRICT TO COUPLING TYPE

To the **Restrict to coupling type** list, add the coupling type that the contributions should be restricted to using the  button.

Contained Feature


Right-click the [Multiphysics Feature](#) node to add the **Contained Feature** node () and to specify the features that a multiphysics feature inherits from. This is a link node,

similar to the [Feature Link](#) node, the difference being that the **Settings** window only contains the [Source Feature](#) section.



If you use a multiphysics interface that links to a built-in interface, you can choose the features of that built-in interface as a contained feature. Use such links with care, because not all built-in features support the automatic GUI generation that the Physics Builder uses.

Auxiliary Settings (Feature Nodes)

A feature can have one **Auxiliary Settings** node () that contains a collection of various settings for its parent feature that you usually do not need to change.

Auxiliary Settings nodes can be added to the parent **Features** nodes under the **Building Blocks** branch, or to the features added under a **Physics Interface** or **Multiphysics Interface**. Right-click the **Auxiliary Settings** node to add [Geometric Nonlinearity](#), [Physics Symbol](#), and [Usage Condition](#) subnodes.

The **Settings** window has the following sections:

GUI ATTRIBUTES

Enter a valid image file name in the **Icon** field to use a different icon for the feature. If this field is empty, the feature uses a default icon, which is different depending on the feature's entity level.

HARMONIC PERTURBATION

Select the **Perturbation method** from the list that has the following options:

- **No perturbation support** (the default).
- **Can act as contribution**. Enables an option so that a user can choose if a feature instance can act either as a stationary contribution or as a harmonic contribution. This option only makes sense for contributing features.
- **Can add contribution as child**. Adds a subfeature to this feature that adds the harmonic contribution. This is typically used for exclusive features. From the **Harmonic contribution feature** list, choose **Automatic** generation of the subfeature or link to a feature with the **From link** option. The automatically generated subfeature adds harmonic contributions for all variables in the parent feature that are defined under a user input and that have the preference property **Interpret as right-hand-side** selected.

VISIBILITY OF MODEL INPUTS

This setting requires that the **Include model input** check box in the **Feature** node's **Preference** section is selected. A model input is by default only shown when a material requests it. In the **Always visible** list, add the model inputs that ignore the materials so that they should always appear in the **Settings** window of the feature instance.

LOAD- AND CONSTRAINT GROUPING

This section has the settings **Enable load groups** and **Enable constraint groups** which can be selected to make the physics feature compatible with the **Load cases** functionality of COMSOL Multiphysics. When load- or constraint groups are enabled for a feature it gets extra menu-items so that the user can select under which group it is active. To make the physics feature useful under load grouping the variables that should be affected by the grouping need to have the **Include in load groups** setting selected in the **Variable declaration** node. Constraints are by default affected by the constraint grouping. If a particular constraint should not be affected by the grouping, then select the **Exclude from constraint grouping** setting in the **Constraint** node.

FEATURE SELECTION

This section contains settings that affect the entities on which the physics feature is applicable. The setting in the **Take selection from** list controls the maximum selection from which the feature can take its selection. That is the selection that is further restricted by the **Applicable entities** setting of the physics feature. The default option, **Parent**, means that the selection should be taken from the physics or in the case of a subfeature from the parent feature. **Physics** means that the selection should be taken from the physics even in the case of a subfeature. The last option, **Geometry**, indicates that the feature should take its selection from the entire geometry.


The other settings are the **Not applicable on infinite element domain** and **Not applicable on perfectly matched layer domain** check box. Select any of these check boxes to make sure that the physics feature cannot be applied on a domain of the particular type.

INPUT COORDINATE SYSTEM

When any of the base vector systems choices of the parent physics feature are **Selected input coordinate system**, this section has settings that control the input coordinate system. First, there is a **Coordinate section** list with the options **Add coordinate system section** (the default) and **Use parent's coordinate systems section**. The latter choice disables all other settings and uses the **Coordinate System Selection** section from the parent if there is any. The **Add coordinate system section** option adds a **Coordinate System Selection** section to the physics feature. It also enables a couple of other settings

that control this section. Select the **Only allow orthonormal systems** check box to only allow the user to choose between orthonormal systems when selecting the input coordinate system of the feature. In the **Filtering of systems** list, choose between the options **Allow boundary systems**, **No boundary systems**, and **Only boundary systems**. Note that the filtering only affects features that live on the boundary geometric entity level.





Auxiliary Settings (Multiphysics Couplings)

The various [Multiphysics Coupling](#) features can have one **Auxiliary Settings** node () each.


GUI ATTRIBUTES

Enter a valid image file name in the **Icon** field to use a different icon for the feature. If this field is empty, the feature uses a default icon, which is different depending on the feature's entity level.

VISIBILITY OF MODEL INPUTS

The **Always visible** list of model inputs makes it possible to select a list of model inputs to always show up in the coupling features if the **Include model inputs** check box is selected in the feature node's **Preferences** section. Click the **Add** button () to choose one or more model inputs from the **Always visible** list in the **Add** window that opens. Use the **Move Up**  , **Move Down**  , and **Delete**  buttons under the table to organize the list if needed.

Geometric Nonlinearity

The **Geometric Nonlinearity** node () is a subnode to the **Auxiliary Settings** node (see [Auxiliary Settings \(Feature Nodes\)](#)) and contains a setting for indicating whether the parent physics feature adds geometric nonlinearity to the problem.

GEOMETRIC NONLINEARITY

Select an option:

- **Never** (the default). This option is the same as not adding the **Geometric nonlinearity** node at all; that is, the feature does not make the model geometrically nonlinear.
- **On request**. Adding the physics feature to a model enables the geometric nonlinearity option in the study steps so that the user can choose if the model should be regarded as geometrically nonlinear. The feature must always generate linear or nonlinear equations depending on the current study step setting. Note that other

features with the geometric nonlinearity setting set to **Always** can take control over the setting, enforcing geometric nonlinearity.


- **Always.** Adding the physics feature to a model makes the geometric nonlinearity option in the study steps visible but selected and disabled so that the model is always regarded as geometrically nonlinear.

There are two parts that need to be implemented when making a physics interface compatible with geometric nonlinearity. The physics features must indicate if they listen to the geometric nonlinearity setting in the study steps, which then must be visible, or if they always add geometric nonlinearity to the problem, thus taking control of the study step setting. This part is done by using the **Geometric Nonlinearity** node.

The other part is to make physics features generate linear or nonlinear equations depending on the geometric nonlinearity setting in the current study step. This is done by placing the relevant variables and equations under a **Usage Condition** node. Typically the features that need support for both linear and nonlinear equations are the ones that have **Geometric nonlinearity** set to **On request**.

To set up a usage condition that is true when the geometric nonlinearity option is selected in the current study step, configure it in the following way: Select the **User input** check box, set **Specify user input** to **By name**, set **User input from** to **Study step**, type `geometricNonlinearity` in the **User input name** field, and finally add “on” as the activating value.

Physics Symbol

The **Physics Symbol** node () is a subnode to the **Auxiliary Settings** node (see [Auxiliary Settings \(Feature Nodes\)](#)) and contains settings that specify a symbol to be shown on the geometry in the **Graphics** window. The symbol displays by default on the selection of the physics feature.

PHYSICS SYMBOL

Enter a valid image filename in the **Icon** field to specify the image that should be used for the symbol. Next to the symbol a few extra decorations can be displayed. From the **Decoration** list choose an option:

- **None** (the default). No decoration should be used.
- **Selected input coordinate system.** A symbol indicating which coordinate system the user has selected for the physics feature.

- **Line to point.** A line from the physics feature's selection to a point that can either be specified by entering coordinates or by selecting **Center of mass of selection** from the **Point** list. To specify coordinates select **Specify coordinates** from the **Point** list and then enter the coordinates in the **Coordinate** field; for example, you can use $\{0, 0, 1\}$ or `par.someVectorInput`. The physics symbol is displayed at the given coordinate.
- **Plane.** You can define one or more planes (in 3D) or lines (in 2D) for visualization in the Graphics window. Each of plane's information is a line in the table, and it contains:
 - In the **Coordinates** column, the coordinates of 3 points not aligned in the 3D plane (3D) or of 2 points in the line (2D) that you want to show. In 3D, the format is $\{A.1, A.2, A.3\}$, $\{B.1, B.2, B.3\}$, $\{C.1, C.2, C.3\}$, and in 2D, the format is $\{A.1, A.2\}$, $\{B.1, B.2\}$.
 - In the **Color [RGBA]** column, the color of the plane or the line in the format of values from 0 to 255 for the red (R), green (G), and blue (B) colors and for the alpha channel (A), which specifies the opacity (for example, $\{222, 49, 99, 128\}$). The alpha value is 0 for a fully transparent object and 255 for a fully opaque object.
- **Normal vector.** An arrow indicating a surface normal. The direction of the surface normal can be reversed by selecting the **Reverse direction** check box.
- **Tangent vector.** An arrow indicating a tangent vector. The direction of the tangent vector can be reversed by selecting the **Reverse direction** check box.

Select the **Only display symbol if feature is selected** check box to make the physics symbol visible only when the physics feature is selected.

SELECTION

Choose an option from the **Selection** list: **From parent** (the default), **Global**, **Operation**, **From definitions library**, **Top level entities applicable to parent**, or **Operation on sibling-feature selections**.

For any choice, except **Global**, choose the **Output entities: Selected entities** (the default), **Adjacent domains**, **Adjacent boundaries**, **Adjacent edges**, **Adjacent points**, **Mesh boundaries**, **Adjacent edges**, **Restricted to geometric entity types**, or **Restricted to frame type**.

- For **Adjacent boundaries**, select an option from the **Restrict to** list: **All adjacent boundaries**, **Exterior boundaries to the domain selection**, **Interior boundaries to the domain selection**, **Exterior boundaries whose up side is in the domain selection**, or **Exterior boundaries whose down side is in the domain selection**.

- For **Adjacent edges** or **Adjacent points**, select an option from the **Restrict to** list: **All adjacent entities**, **Exterior entities to the selection**, or **Interior entities to the selection**.
- For **Restricted to geometric entity types** choose the **Allowed entity types** in the table, **Interior** or **Exterior**.
- For **Restricted to frame type** choose a **Frame type**: **Material** (the default), **Mesh**, **Geometry**, or **Spatial**.

Operation

For **Operation**, choose the **Operation type**: **Union** (the default), **Intersection**, **Difference**, or **Complement**. Then define the **Input selections**, and for **Difference** the **Selections to subtract**. Select options from the **Output entities** list as defined above.

From definitions library

For **From definitions library**, choose an option from the **Link** list and select options from the **Output entities** list as defined above.

Operation on sibling-feature selections

For **Operation on sibling-feature selections**, choose the **Operation type**: **Union** (the default), **Intersection**, **Difference**, or **Complement**. Then define the **Input feature types**, and, for **Difference**, the **Feature types to subtract**. Select options from the **Output entities** list as defined above. Instead of writing the feature ID of the physics feature to use in the selection, you can also write a path of IDs. The first ID is then the physics ID followed by one or several feature IDs. Regular expressions are accepted to match the IDs against a pattern. If the current entity is a coupling feature, the path syntax also accepts a coupling type instead of the physics ID. The coupling type must be preceded with the multiphysics prefix (mph.) and uses the coupling feature's selected physics of this coupling type. Here are some examples:

- **FeatureB**: Looks for the selections of features with ID **FeatureB**.
- **InterfaceA/FeatureB/FeatureC**: Looks for the selections under **FeatureB** with ID **FeatureC**. The interface ID must be **InterfaceA**.
- **\w+/FeatureB/Feature[A-C]**: Looks for the selections under **FeatureB** with IDs that start with **Feature** and end with any of the letters A, B, or C. The interface ID can be any nonempty sequence of word characters.
- **mph.TypeA/Feature[A-C]**: Looks for selections of features under the interface of coupling type **TypeA** in a coupling feature. The features must have IDs that start with **Feature** and end with any of the letters A, B, or C.

Note that a physics feature can only match interface IDs of the physics that it belongs to. A coupling feature can only match interfaces that are part of its selected interfaces.

User Inputs

In this section:

- [Creating User Inputs](#)
- [User Input](#)
- [Selectable Input](#)
- [Selection Input](#)
- [Boolean Input](#)
- [User Input Group](#)
- [Text Label](#)
- [Buttons](#)
- [Section](#)
- [Constraint Settings Section](#)
- [Material Property](#)
- [Material List](#)
- [Feature Input](#)
- [Activation Condition](#)
- [Additional Requirement](#)
- [Allowed Values](#)
- [Activating Allowed Values](#)
- [Button](#)
- [Integer Values Check](#)
- [Real Values Check](#)
- [Regular Expression Check](#)
- [Named Group Members](#)



Use this section in combination with the [Designing the GUI Layout](#) section.

Creating User Inputs

A **User Input** is a parameter that shows up in the **Settings** window of the feature instance in the Model Builder. A user input always represent data storage in a COMSOL Multiphysics model (MPH-file) built with the Model Builder. You can choose to hide a user input from the **Settings** window, and then the user input only represents a data storage.



One important difference between user inputs and variables is that all variables created by the Physics Builder are not saved in the MPH-file, but they can be recreated from the stored data (essentially user inputs) in an MPH-file together with the Physics Builder file (extension `.mphphb`).

The value of a variable, on the other hand, can be read when solving or from result features in the Model Builder. This is not possible with user inputs. If you want to access a value of a user input, you must first declare a variable and define its value equal to the value of the user input. It is possible to add a variable definition to a user input node, which represents a variable declaration and a variable definition. The declaration takes the information from the user input, and the definition gives the declared variable the value of the user input.

STANDARD USER INPUTS

There are these types of standard user inputs:

- **User Input.** The general user input that supports all settings and sizes. It is a bit more complex to setup due to the number of supported settings.
- **Selectable Input.** A user input that a user only can set to a predefined set of allowed values. It does not support units or array types. The typical GUI component is a combo box (or drop-down menu).
- **Boolean Input.** A user input that only allows two values, 0 and 1 (represents false and true). It does not support units or array types. The typical GUI component is a check box.

SPECIAL VERSIONS OF USER INPUT

There are also special versions of the user input, which add predefined declarations of one or several user inputs with a special relation, for example. These special user inputs are summarized below:

- **Material Property.** Defines two user inputs: one for selecting the source of the material data and the other for specifying a user-defined value.
- **Material List.** Defines a user input that contains a material selection.
- **Feature Input.** Defines two user inputs: one for selecting the source of the feature input data and the other for specifying a user-defined value.
- **Selection Input.** Defines an extra selection of geometrical entities for the feature to select entities on different domains when coupling them to the feature, for example

GUI OPTIONS SETTINGS


A user input also needs information about its appearance in the user interface. It is possible to control how it appears, when it appears, and where it appears in the **Settings** window of a feature or property. You control these settings from the **GUI Options**

section of each user input node and through a special activation condition node; see [Activation Condition](#).




Also see [Additional Requirement](#), [Allowed Values](#), [Activating Allowed Values](#), [Named Group Members](#), [Integer Values Check](#), [Regular Expression Check](#), [Variable Definition](#), and [Component Settings](#).

User Input

A **User Input** () for a feature or property becomes a data storage editable by the user for the corresponding instance in the Model Builder (see [User Inputs](#) for an overview).

To add a **User Input**, first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **User Input** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

The **Settings** window of a user input contains the following sections:

DECLARATION

In the **Declaration** section you specify the name of the parameter in the **Input name** field. The description, symbol, physical quantity, and SI unit of the user input work identical to variable declaration; see [Variable Declaration](#) and [Dependent Variable Declaration](#).

The dimension and default value of the user input is a bit different compared to variables. For scalars, vectors, and matrices you set the desired dimension in the **Dimension** list. Use the **Single** array type for the **Array type** list, which also is the default. When you need user inputs that are vectors of vectors, or vector of matrices, choose **Double** in the **Array type** list. You can then choose the outer dimension with the **Outer dimension** list and the inner dimension with the **Inner dimension** list. The **Inner dimension** list is identical to the **Dimension** list, but the **Outer dimension** list has fewer options. For all double-array user inputs, the default value refers to the inner dimension and fills up the outer dimension with identical defaults.

When the user input is a scalar, there is an option to define a set of allowed values. In the **Allowed values** list, you can choose **Any** or **From list**. If you choose **From list**, a table shows up that you can fill the with allowed values. The **Default value** list only contains

the values entered in the **Allowed values** list. If you choose **Any** in the **Allowed values** list, you specify the default in the **Default value** field.

User inputs of the type **Boolean** is a special case of a scalar with two allowed values, internally represented with 0 and 1. The obvious control type for this user input is a check box that either can be cleared (0) or selected (1). To make the check box selected by default, select the **Default selected** check box.

When you want vectors where each element must be in a set of allowed values or is a Boolean, you use a double-array user input. Set the outer dimension as desired, and choose **Scalar** or **Boolean** from the **Inner dimension** list. If you choose **Scalar**, you can set the allowed values as described above.

RESTRICTIONS

In the **Allowed space dimensions** list you can define what space dimensions this particular user input can be used in. The option **Same as parent** (the default option) means that the user input supports the same space dimensions as the parent feature or property. If you want to control the space dimensions manually, choose **Customized** from the list. You can then select the allowed space dimensions from a list of all space dimensions.



If the user input does not support a space dimension, it becomes completely removed from the feature or property, not just hidden from the user interface. If you then try to access it in a variable definition, for example, an error message appears.

ADVANCED

This section contains advanced options that you do not have to change in most cases. In the **Input base vector system** list, you can override the input base vector system specified by the parent (for example, a feature or property) by choosing something else than the default option **Same as parent**. See [Using Coordinate Systems](#) and [Transformation Between Coordinate Systems](#) for more information about selecting base vector systems.

GUI OPTIONS


If you want the user input to disappear when it is inactive, select the **Hide user input in GUI when inactive** check box. Select the **Show no description** check box to hide the text label containing the description above the GUI component of the user input. Similarly, you can hide the symbol from the GUI by selecting the **Show no symbol** check box. As

a final option, it is possible to add a divider above the GUI component and any description text label. The divider is a horizontal line with an optional descriptive text. Select the **Add divider above the user input** check box to add the divider. When selected, you can enter the divider text in the **Text** field.




See [Designing the GUI Layout](#) for more information

Selectable Input

A **Selectable Input** node () is a special version of the **User Input** node that defines a data storage that can only contain values from a list. This node is equivalent to a **User Input** node with a scalar or scalar inner dimension and with the **Allowed values** list set to **From list**. The main difference is that the selectable input defines its allowed values through **Allowed Values** subnodes. (see [User Inputs](#) for an overview).

To add a **Selectable Input**, first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Selectable Input** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.


The **Settings** window of a selectable input has the following sections:

DECLARATION

Almost the same as for the [User Input](#). The **Default value** list can either be the **First valid value** (the default) or **Custom default**. The latter allows you to enter an arbitrary default value as long as it is among the allowed values defined for the input.

See [User Input](#) for the rest of the settings.

Selection Input

A **Selection Input** node () adds an extra a selection of geometrical entities to the parent feature. Use such extra selections to show a selection in the graphics rendering or to select entities on different domains when coupling them to this feature, for example.

To add a **Selection Input**, first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

The **Settings** window of a selection input has the following sections:

DECLARATION

In the **Declaration** section you specify the name and a description of the selection input in the **Input name** and **Description** fields.

You also specify the entity level from the **Geometric entity level** list. The default is the same level as the parent feature. The choices **Same as top level (Domain condition)** and **One level below top level (Boundary condition)** are relative to the top level of the interface. If the top level of the interface is **Domain**, choosing **One level below top level (Boundary Condition)** means that the selection input applies to boundaries. If the top level is **Boundary**, choosing **One level below top level (Boundary Condition)**, the selections are for lines (edges) in 3D and points in 2D. The choices **Same as parent level** and **One level below parent level** make the selections apply to entities of the same geometric entity level or one level below (boundaries, if the parent level is domains, for example), respectively.

Initially, the selection can be cleared or include all entities for the selected geometric entity level. Choose **Cleared** (the default) or **All entities** from the **Initial selection** list.

RESTRICTIONS


In the **Allowed space dimensions** list you can define what space dimensions this particular selection input can be used in. The option **Same as parent** (the default option) means that the selection input supports the same space dimensions as the parent feature or property. If you want to control the space dimensions manually, choose **Customized** from the list. You can then select the allowed space dimensions from a list of all space dimensions.

SELECTION SETTINGS

Applicable Entities

It is possible to limit the types of entities where the selection can be active. For example, a boundary selection can be limited to only interior boundaries. All other selected boundaries then get marked as *not applicable*. For more complex situations, you can use conditions on domain types to decide if a specific boundary is applicable or not.

Select a form of **Applicable entities** from the list: **From entity types** (the default) or **From sequence**.

- If **From entity types** is selected, select one or more of the options available when you click the **Add** button (**+**) add them to the list of applicable entities. **Exterior** and **Interior** boundaries are selected by default. Use the buttons under the list to organize as needed.
- If **From sequence** is selected, select a **Selection filter sequence: Locally defined** (the default) or **Imported from external resource**. Select a sequence to link to from the **Link** list. Click the **Go to Source** button  as needed. If the sequence is **Imported from external resource** also select an **Imported file**.





For more details on entity types, see [Selection Terminology](#).

Override Rule

Some features cannot exist on the same selection (for example a boundary), and COMSOL Multiphysics uses override rules to decide how the selection of one feature overrides another feature's selection.

Select an **Override rule**: **Built in** (the default), **Locally defined**, **Imported from external resource**, or **None**.

- If **Built in** is selected there are four options in the **Override type** list: **Exclusive**, **Contributing**, **Override features of same type**, and **Never overridden**:
 - **Exclusive** means that the feature should override all other features except those that are of the type **Never overridden**. An exclusive feature is overridden by other exclusive features.
 - **Contributing** means that the feature should not override other features. The exception is for entity levels below the top entity level of the physics, where features of any type override the default features.
 - **Override features of same type** means that the feature only overrides other features of the same feature type.
 - **Never overridden** ensures that a feature is never overridden and does not override any other feature.
- If **Locally defined** is selected, also select a **Link** from the list. Click the **Go to Source** button  as needed.

- If **Imported from external resource** is selected, select an **Imported file** from the list. Click the **Go to Source** button  as needed. Then select a **Link** from the list.
- In **None** is selected, no override rule is used.

Override Type

The **Override type** list specifies if the feature is **Exclusive** or **Contributing** to other features. In the Model Builder, an *exclusive feature* (such as constraints and fixed values) replaces all previous feature instances for intersecting selections, whereas a *contributing feature* (such as loads and sources) adds to other contributing features sharing the same selection. You also select to only **Override features of the same type** or to let the feature be **Never overridden**.

GUI OPTIONS


In this section you can specify some options for the user interface.

To lock the selection so that it cannot be modified, select the **Lock selection** check box.


By default, the section is initially expanded. Select the **Section is initially collapsed** check box if it should be collapsed when first displaying the Settings window.

To hide the selection in the Settings window, select the **Hide in GUI** check box. Even if the selection is hidden from the Settings window, you can choose to still show the rendering of the selected entities in the user interface by selecting the **Show rendering when hidden** check box.

Boolean Input

A **Boolean Input** node () is a special version of the **User Input** node that defines a data storage that can only contain the values 0 or 1. This node is equivalent to a **User Input** node of the type Boolean.

To add a **Boolean Input**, first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Boolean Input** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.


The **Settings** window of a Boolean input contains the following sections:

DECLARATION


Almost the same as for the [User Input](#). Select the **Default selected** check box get the value 1 by default.

See [User Input](#) for the rest of the settings.

User Input Group

The **User Input Group** node () is a collection of other user inputs, user input groups, and material parameters. Use it to organize the GUI layout of the feature and property. This can be a complex task, and you can read more about the details of this process in [Designing the GUI Layout](#).

To add a **User Input Group**, first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **User Input Group** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.


The **Settings** window has the following sections:

DECLARATION

The **Group name** field cannot be in conflict with any other User Input or User Input Group. In the **Description** field enter a text to display for the group. For groups that represent a section, this description becomes the title of the section. You can pass the component's arguments to the description. The argument name must be placed between two # characters. For example, if you have an argument `arg.param = material`, the string `#Plot_in the _arg.param#_frame` will be displayed as `Plot in the material frame`.



For other groups, the description pops up above the location of the group (if visible) and usually looks strange. Clear the **Description** field to avoid this effect, which is the default behavior.

In the **Group members** list you add members to the group. Click **Add** () to get a list of all possible members to add.

GUI OPTIONS


Use the **GUI Layout** list to choose how to organize the group members in the window. The options are **Group members below each other** (the default), **Group members placed in a stack**, **Create a widget for each vector component**, **Radio buttons from first user input**, **others interleaved**, **Group members define columns in table**, **Group members define a section**, or **Attach activation on group members**.

If you choose **Group members define a section** for example, the group members are placed sequentially in a section in the window. Depending on the choice you made in the **GUI Layout** list, different options appear beneath the list.




See [User Input Group GUI Options](#) for more information

Text Label

The **Text Label** node () adds a static text to a Settings window for a feature or property. The text can provide explanations on how to use the feature or other information.

To add a **Text Label**, first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **User Input Group** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

The **Settings** window has the following section:


TEXT FORM

In the **Name** field, enter a name for the text label.


Select the **Multiline text** check box if you want the text label to include more than one line of text.

In the **Text** field, enter the text to display in the feature's Settings window.

Buttons

The **Buttons** node () adds a row of buttons to the Setting windows of a feature or physics interface of the Model Builder.

To add a **Buttons** node, first add a feature node or a property node, for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Buttons** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.


The **Settings** window has the following sections:




BUTTON

Add a **Name** for the buttons group.

From the **Action type** list, you choose which type of action to associate with the buttons: Choose **Button handler from code editor** (the default) to define the action handler from a [Code Editor](#) node (see [Code Editor](#)), or choose **Internal action tag** to use an action manager and let the buttons refer to this action by using an internal tag that you enter in the **Action tag** field.

ACTIONS


You add actions using the **Add** button () under the table. If the **Action type** is set to **Button handler from code editor**, you enter a **Description**, the filename for the **Icon** to use, and choose an existing **Code Editor** as the handler (or **None**) in the **Handler** column. If the **Action type** is set to **Internal action tag**, you enter a **Description** and the filename for the **Icon** to use for the action. If the **Icon** field is empty, the button becomes a text button; otherwise the button shows the image in the file specified in the **Icon** field.

Use the **Move Up** , **Move Down** , and **Delete**  buttons under the table to organize the list.


GUI OPTIONS

If you want the buttons to disappear when they are inactive, select the **Hide user input in GUI when inactive** check box. It is also possible to add a divider above the GUI component and any row of buttons. The divider is a horizontal line with an optional descriptive text. Select the **Add divider above the material property** check box to add the divider. When selected, you can enter the divider text in the **Text** field.

Section

A **Section** node () is a special version of the **User Input Group** node that defines a section. This node is equivalent to a **User Input Group** node where **GUI Layout** list has the option **Group members define a section**.


To add a **Section**, first add a feature node or property node (for example, a **Generic Feature**, **Domain Condition**, or **Device Model Feature**), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Section** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

The **Settings** window has the following sections:

DECLARATION

The **Group name** field cannot be in conflict with any other section, User Input or User Input Group. In the **Description** field, enter a text for the section title. You can pass the component's arguments to the description. The argument name must be placed between two # characters. For example, if you have an argument `arg.param = material`, the string `#Plot_in the _#arg.param#_frame` will be displayed as `Plot` in the `material` frame.

In the **Group members** list add members to display under this section, in the order they appear in the list. Click **Add** () to get a list of all possible members to add.


GUI OPTIONS

In the **Category for hiding section** list, choose among possible options to hide the section with respect to the global display settings **Advanced physics options**, **Discretization**, and **Stabilization**. The option **None** (the default) disables the hiding of this section. Select the **Section is initially collapsed** to collapse the section by default.




Designing the GUI Layout

Constraint Settings Section

A **Constraint Settings Section** node () is a special section that a feature with constraints uses. A feature always have this section when there is at least one constraint.

Add this node when you want to include constraint settings although there is no constraints for the feature, or if you want to modify the content of the section. The section always contains a selectable input that allows a user to select the type of constraint. It also contain a check box for weak constraints when necessary. Similar to the [Selectable Input](#) node, [Allowed Values](#) subnodes defines the allowed values for the selectable input of this section. There can only be one **Constraint Settings Section** node for each physics feature.

To add a **Constraint Settings Section** first add a feature node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Constraint Settings Section** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

The **Settings** window has the following sections:

SETTINGS FOR APPLYING REACTION TERMS

Under **Supported options defined by child nodes**, the **Default value** list can either be the **First valid value** (the default) or **Custom default**. The latter allows you to enter an arbitrary default value as long as it is among the allowed values defined for the input.

CONSTRAINT METHOD


The **Default method** list controls the default value for the constraint method setting in the Constraint Settings section for physics features that adds constraints. The options are **Elemental** (the default) or **Nodal**.

GUI OPTIONS


In the **Category for hiding section** list, choose among possible options to hide the section with respect to the global display settings **Advanced physics options** (the default), **Discretization**, and **Stabilization**. The option **None** disables the hiding of this section; for **None**, select the **Section is initially collapsed** check box to make the section initially appear as collapsed instead of expanded.



Material Property

The **Material Property** node () is a special case of a **User Input** that also supports linking with material properties from the **Materials** node in the Model Builder. A material property node is actually a collection of two user inputs: one list with the options **From material** or **User defined** and one property value field for user-defined values.

To add a **Material Property** first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Material Property** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.



Also see [Material Property \(Auxiliary Definitions\)](#), which is a subnode of the [Material Property Group](#) node under **Auxiliary Definitions**.

The **Settings** window has the following sections:


DECLARATION

The material property can either be a basic material property or a property that belongs to a predefined material property group. Choose a **Property type**: **Basic material property** (the default), **Built in predefined material property**, **Locally defined material property**, or **Imported material property from external resources**.

- A **Basic material property** belongs to a predefined set of quantities that are commonly used in material libraries.
 - Enter a valid **Property name**, **Description**, and **Symbol (LaTeX encoded)**. See [Variable Declaration](#).
 - Choose a **Physical quantity**. The quantity identifies the corresponding property in the material libraries. A property in the material library has a certain dimension, but it is possible to use a smaller dimension for a material property in a feature.
 - Choose the **Dimension**, which can only contain **Scalar** and **Custom** for a scalar material parameter.
 - Enter and choose [Default Values](#).

- For **Built in predefined material property**, choose a predefined **Material property group** from the list. Then choose a **Material property** and enter and choose [Default Values](#).
- For **Locally defined material property**, choose a **Material model** from the list. Then choose a **Material property** and choose [Default Values](#).
- For **Imported material property from external resources**, choose an **Imported file** from the list. Then choose a **Link** and a **Material property**. Then choose [Default Values](#).



Select **Locally defined** from the top of the **Physical quantity** list to use one of the locally defined physical quantities, which you choose from the **Link** list. Click the **Go to Source** button () to move to the [Physical Quantity](#) node for the selected local physical quantity.

Default Values

In the **Default value for user defined** field, which can be a table, enter the default value for the user-defined option.

Use the **Default value** list to set the default for the feature instance. Choose **From material** (the default), **User defined**, **First allowed value**, or **Custom default**. **First allowed value** takes the first active value from the values specified by the **Allowed Values** child nodes. Use the **Custom default** option to enter a value. This value must always be active for the list in the feature instance.

OPTIONS

In the Model Builder most material properties get their values from the material assigned to the geometric entities overlapping with the selection of the feature instance. You can add a [Material List](#) to the feature, which allows you to choose among all materials added to a model. Select the **Couple to material list** check box to specify that this property gets linked to the material selected in the list. Then choose the list to couple against in the **Material list**, either by entering a name in the **Name** field or by choosing a particular material list reference.

For a **Basic material property**, the **Pick material property from property groups in selected materials** check box is available. Select this check box to make the material property

take its values from the property groups of the material that is selected in the material list.



For a **Basic material property** with the **Physical quantity** set to **Diffusion coefficient**, the values that you can choose from are the property groups of the material selected in the material list that define the output property for the diffusion coefficient.

In addition to the options **From material** and **User defined**, you can add extra items to the list in the feature instance with **Allowed Values** child nodes. An alternative option is also entering them directly to the **Extra list items** table.

RESTRICTIONS

See [User Input](#) for information about this setting.


GUI OPTIONS

If you want the material property to disappear when it is inactive, select the **Hide user input in GUI when inactive** check box. Select the **Show no description** check box to hide the text label containing the description above the GUI component of the property. Similarly, you can hide the symbol from the GUI by selecting the **Show no symbol** check box. As a final option, it is possible to add a divider above the GUI component and any description text label. The divider is a horizontal line with an optional descriptive text. Select the **Add divider above the material property** check box to add the divider. When selected, you can enter the divider text in the **Text** field.




- [Designing the GUI Layout](#)
- [Materials](#) in the *COMSOL Multiphysics Reference Manual*

External Material List

In the Model Builder most material properties get their values from the material assigned to the geometric entities overlapping with the selection of the feature instance. As an option, add a **External Material List** node ()

This node lists all external materials that satisfy the list of the input and output variables to the external socket. The list of socket inputs and socket outputs must be defined as subnodes of this node (see [Socket Input](#) and [Socket Output](#)).

To add an **External Material List** first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **External Material List** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

EXTERNAL MATERIAL LIST

Specify the name of the external material list in the **Name** field, and the description in the **Description** field.

ADVANCED

By default, the integration order is determined automatically. To set an integration order manually, select **Customized** instead of **Automatic** from the **Integration order** list and then add an order as a positive integer in the **Order** field (default: 2).


GUI OPTIONS

If you want the material list to disappear when it is inactive, select the **Hide user input in GUI when inactive** check box. Select the **Show no description** check box to hide the text label containing the description above the GUI component of the material list. Similarly, you can hide the symbol from the GUI by selecting the **Show no symbol** check box. As a final option, it is possible to add a divider above the GUI component and any description text label. The divider is a horizontal line with an optional descriptive text. Select the **Add divider above the material list** check box to add the divider. When selected, you can enter the divider text in the **Text** field.



- [Designing the GUI Layout](#)
 - [Materials](#) in the *COMSOL Multiphysics Reference Manual*
-

Socket Input

Right-click an **External Material List** node to add a **Socket Input** subnode () where you can define an input for the external material.

DEFINITION

The **Variable name** field is inherited from the parent **External Material List** and shows the name to use to define a value for the socket input.

From the **Socket input quantities** list, choose the physical quantity of the socket input.

From the **Type** list is typically preset and unavailable.

In the **Expression** field, enter an expression for the input quantity that defines the value to send to the external material.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES

See [Preferences](#) for the Variable Definitions node.

PROTECTION

Set preferences that enable protection of entered expressions. Select the **Hide expression in equation view** check box to remove the definition to display in the **Equation View** node, which is a subnode to a physics feature in the Model Builder. This disables any possibility to alter the expression; it also makes it harder to read the expression.

To further complicate reading of the expression, you can select the **Encrypt expression** check box. This turns on an encryption of the expression in the saved model file and when accessing the expression in a model file for Java code. It also encrypts the tensor expression when you compile the archive (see [Compiling an Archive](#)), so the expression in a distributed builder file (*.mphphb) cannot be read.

Socket Output

Right-click an **External Material List** node to add a **Socket Output** subnode (**a=**) where you can defined an output for the external material.

DEFINITION

Define the **Variable name** field for the name of the variable that the external material will send its output to.

From the **Socket output quantities** list, choose the physical quantity of the socket output.

The **Type** list is typically preset and unavailable.

In the **Expression** field, enter an expression for the output quantity that defines the value to send from the external material.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES


See [Preferences](#) for the Variable Definitions node.

PROTECTION


Set preferences that enable protection of entered expressions. Select the **Hide expression in equation view** check box to remove the definition to display in the **Equation View** node, which is a subnode to a physics feature in the Model Builder. This disables any possibility to alter the expression; it also makes it harder to read the expression.

To further complicate reading of the expression, you can select the **Encrypt expression** check box. This turns on an encryption of the expression in the saved model file and when accessing the expression in a model file for Java code. It also encrypts the tensor expression when you compile the archive (see [Compiling an Archive](#)), so the expression in a distributed builder file (*.mphphb) cannot be read.

Material List

In the Model Builder most material properties get their values from the material assigned to the geometric entities overlapping with the selection of the feature instance. As an option, add a **Material List** node () . Any material property coupled to a material list makes it possible to choose among all materials added to a model. See [Material Property](#) for information how to couple a property to a list. For a feature instance in the Model Builder, a list displays with the first option **Domain material** followed by all materials. The **Domain material** option is identical to the original behavior without a list. By choosing a specific material, all material properties coupled to this list get the property values stored in that material independently of any selection.

To add a **Material List** first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Material List** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

MATERIAL LIST

Specify the name of the material list in the **Name** field, and the description in the **Description** field. You can pass the component's arguments to the description. The argument name must be placed between two # characters. For example, if you have an argument `arg.param = material`, the string `#Plot_in the _#arg.param#_frame` will be displayed as `Plot in the material frame`.

Similar to the [Material Property](#) and [Feature Input](#) nodes, you can add extra list items apart from the automatically added ones (**Domain materials** and list of materials in this case). You can also remove the **Domain materials** option from the list by clearing the **Allow domain material** check box.


GUI OPTIONS

If you want the material list to disappear when it is inactive, select the **Hide user input in GUI when inactive** check box. Select the **Show no description** check box to hide the text label containing the description above the GUI component of the material list. Similarly, you can hide the symbol from the GUI by selecting the **Show no symbol** check box. As a final option, it is possible to add a divider above the GUI component and any description text label. The divider is a horizontal line with an optional descriptive text. Select the **Add divider above the material list** check box to add the divider. When selected, you can enter the divider text in the **Text** field.




- [Designing the GUI Layout](#)
 - [Materials](#) in the *COMSOL Multiphysics Reference Manual*
-

Feature Input

The **Feature Input** node () is a special case of a **User Input** that also supports linking with any announced variable in the Model Builder. A feature input is matched against an announced variable by matching against a physical quantity that you have to set for both items. See [Variable Declaration](#) for information about how to set the physical quantity and the announce preference for variables. Similar to the [Material Property](#), the feature input is a collection of two GUI components: one list with the matching variables plus a **User defined** option and one field for the user-defined value.

To add a **Feature Input** first add a feature node or property node (for example, a [Generic Feature](#), [Domain Condition](#), or [Device Model Feature](#)), then:

- From the contextual toolbar (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**), click the **Feature Input** () button.
- Right-click the feature node (in this example, **Generic Feature**, **Domain Condition**, or **Device Model Feature**) and select it from the **Inputs** submenu.

DECLARATION

The declaration of a feature input is similar to that of a material property, where you specify the name, description, symbol, physical quantity, and dimension; see [Material Property](#) for further details. An announced quantity has a certain dimension, but it is possible to use a smaller dimension for a feature input in a feature. You set this dimension with the **Dimension** list, which can only contain **Scalar** and **Custom** for a scalar feature input.

OPTIONS

In addition to the announced variables and **User defined**, you can add extra items to the list in the feature instance with **Allowed Values** child nodes. An alternative option is also entering them directly to the **Extra list items** table.

Select the **Use as model input** check box to use this feature input as a model input. The feature input name will start with `minput_` and continue with the physical quantity field name. The feature input will then be placed in the model input section in the user interface and will be used as model input by the materials.

A feature input supports several levels of matching that you choose from the **Feature input match type** list. You can find a brief explanation of the matching options below:

- **Always match.** If there is any matching announced variable, always use the first one found.
- **Synchronized.** Match to a matching announced variable, if the provider to that variable fulfills the following conditions (see below for an example):
 - It has a feature input of the quantity chosen in the **Synchronized physical quantity** list.
 - That feature input chooses an announced variable from the provider owning the synchronized feature input.
- **Always match within physics interface.** If there is any matching announced variable by the current physics interface, use the first one found.

- **Model input match.** Also matches within the physics interface, but you can disable the matching through an interface setting.
- **Never match.** Never do any automatic matching. The default option is **User defined**, but you can always manually choose among the found announced variables.

For all the options except the **Synchronized** option, it is possible to set a regular expression in the **Match tag filter (Regular expression)** combined text field and list. The predefined options here are **None** (the default), and all physical quantity field names. **None** is equivalent to an empty tag and should be unless it is necessary to limit matching in some sense. Use the regular expression if the input should accept several match tags; use, for example, the expression `^$|explicit` to either match the default empty tag or an announced variable using the tag `explicit`. There are several resources on the Internet that explain the exact syntax of regular expressions.

Use the **Synchronized** option in situations when the matching depends on another feature input in the provider (typically a physics feature) of an announced variable. For example, assume feature “A” has a feature input with the **Never match** option that picks up variables with the quantity *electric potential*. Feature “B” announces an electric potential without a tag and has a synchronized feature input that should pick up a current variable only from the feature that picks up an electric potential from feature “B.” To make feature “A” a possible match, it has to announce a current variable with an **Electric potential** match tag. The synchronized feature input uses the **Synchronized** option but also needs the **Electric potential** option in the **Synchronized physical quantity** list. When the user selects the electric potential from a physics feature of type “B” in the input list of a physics feature of type “A”, the input (often hidden) in physics feature of type “B” automatically picks up (or auto-match) the electric current from physics feature “A”.

GUI OPTIONS


If you want the feature input to disappear when it is inactive, select the **Hide user input in GUI when inactive** check box. Select the **Show no description** check box to hide the text label containing the description above the GUI component of the input. Similarly, you can hide the symbol from the GUI by selecting the **Show no symbol** check box. As a final option, it is possible to add a divider above the GUI component and any description text label. The divider is a horizontal line with an optional descriptive text.

Select the **Add divider above the feature input** check box to add the divider. When selected, you can enter the divider text in the **Text** field.



- [Designing the GUI Layout](#)
- [Materials](#) in the *COMSOL Multiphysics Reference Manual*

Activation Condition

Use an **Activation Condition** node () to specify the conditions to dynamically enable or disable an user input, material property, feature input, and material list. The condition can depend on user inputs in the parent feature or property, or any other user input in a property within the physics interface. Conditions can either be true if the other user input has a specified value or if it is active. Right-click any feature node to add this subnode. Also right-click the **Activation Condition** node to add an [Additional Requirement](#) subnode. The **Settings** window has the following section:

ACTIVATION CONDITION

The activation condition can depend on user inputs in the parent feature, parent property, or some property. If the user input is under a feature or property, which can contain other user inputs, you can directly refer to any of those user inputs by choosing **By reference** in the **Specify user input** list. You then choose the user input from the **User input** list, and finally add or enter the values that activates the user input in the **Activating values** list. This list differs depending on the type of user input you refer to.

The other option in the **Specify user input** list is **By name**, which means that you enter the name in the **User input** field. The name is entered in the **Input name** field of the user input you want to refer to. For the **By name** option, you also have the list **User input from**, where you can choose if the user input is under the parent feature (**This feature** option), or under a property in the physics interface (**Property** option). With the **Property** option, you must also enter the type of the property that contains the user input in the **Property** field. For Boolean user inputs, the values **Selected** and **Not selected** you enter as 0 and 1.

Select an **In expression**. This is a general tool that can evaluate an expression of relations and Boolean operators. It also supports some special functions and names, see the section about [Usage Condition](#) for more details. You can select the **Invert condition** check box to invert the entire condition. The **Require input is active** check box is selected by default. It is only applicable when specifying a user input to check by reference or by name, not for expressions. When selected, the activation condition is

only true if the checked user input is also active as decided by its activation conditions. For expressions, you can achieve the equivalent logic using the `isActive` operator.

Additional Requirement

An additional requirement to an activation condition is a requirement that has to be fulfilled otherwise the activation condition cannot be fulfilled.

Right-click an **Activation Condition** node to add this subnode.

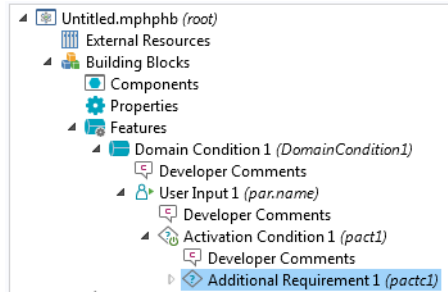



Figure 3-1: A submode of the Activation Condition node for additional requirements.

Allowed Values

Use the **Allowed Values** node () to specify the allowed values of the parent node, the parent node, which can be either of the types **Selectable Input**, **Material Property**, or **Feature Input**. The allowed values can also be dynamic with respect to a condition. The condition can depend on user inputs in the parent feature or property, or any other user input in a property within the physics interface. Conditions can either be true if the other user input has a specified value or if it is active. This node is also a child of a **Constraint Settings Section** node, controlling the allowed values of the selectable input included in that section. The **Settings** window of the node contains the following sections:

ALLOWED VALUES

Enter the allowed values in the **Allowed values** table that this node contribute to the parent node. Enter the value in the first column and the description in the second column. For allowed values under a **Constraint Settings Section**, there is also a third column where you specify if the value allows weak constraints.


ACTIVATION CONDITION

Enable the condition by selecting the **Use condition** check box. The activation condition can depend on user inputs in the parent feature, parent property, or some property. If the user input is under a feature or property, which can contain other user inputs, you can directly refer to any of those user inputs by choosing **By reference** in the **Specify user input** list. You then choose the user input from the **User input** list, and finally add or enter the values that activates the user input in the **Activating values** list. This list differs depending on the user input you refer to.

Select the **Invert condition on input values** check box to invert (negate) the condition.

The **Require input is active** check box is selected by default. It is only applicable when specifying a user input to check by reference or by name, not for expressions. When selected, the activation condition is only true if the checked user input is also active as decided by its activation conditions. For expressions, you can achieve the equivalent logic using the `isActive` operator.

Activating Allowed Values

Use the **Activating Allowed Values** node () to specify the conditions to dynamically allow values from the list of allowed values specified in the parent **User Input** node. The condition can depend on user inputs in the parent feature or property, or any other user input in a property within the physics interface. Conditions can either be true if the other user input has a specified value or if it is active. The **Settings** window has the following sections:

ALLOWED VALUES

Add the allowed values from the parent node to the **Allowed values** list that this node can activate. When the condition specified in the **Activation Condition** section is true, the allowed values for the parent user input contain the allowed values specified in the list.

ACTIVATION CONDITION


The activation condition can depend on user inputs in the parent feature, parent property, or some property. If the user input is under a feature or property, which can contain other user inputs, you can directly refer to any of those user inputs by choosing **By reference** in the **Specify user input** list. You then choose the user input from the **User input** list, and finally add or enter the values that activates the user input in the **Activating values** list. This list differs depending on the user input you refer to.

The last option in the list is **In expression**. This is a very general tool that can evaluate an expression of relations and Boolean operators. It also supports some special functions and names, see the section about [Usage Condition](#) for more details.

Select the **Invert condition on input values** check box to invert (negate) the condition.

The **Require input is active** check box is selected by default. It is only applicable when specifying a user input to check by reference or by name, not for expressions. When selected, the activation condition is only true if the checked user input is also active as decided by its activation conditions. For expressions, you can achieve the equivalent logic using the `isActive` operator.

Button

The **Button** node () adds a text or image button beside an user input or selectable input.

To add a **Button** node, right-click a **User Input** or **Selectable Input** node and select it from the context menu.

The **Settings** window has the following section:

ACTION


Enter a **Description** for the button. If you want to use an image button, select the **Icon** check box and enter the name of the image file in the associated text field.

If the **Action type** is set to **Button handler from code editor** (the default), you determine the source using the Links from list. Available options are:

- **Building blocks**. Lets you choose among the **Code Editor** node under the **Components** branch in the **Building Blocks** branch. You choose the code editor from the **Link** list.
- **Built in**. Lets you choose among built-in code editors that are available to the Physics Builder. In the **Package** list you choose the main resource to use a code editor from (or **None**). For each resource, you have a list of code editors in the **Link** list to choose from. You can only choose among the currently published code editors. The list shows **Not active** if no links are available.
- **External resources**. Lets you choose among the code editors listed in an imported builder file under the **External Resources** branch. You choose the file from the **Imported file** list. The **Link** list contains all code editors found in the **Building Blocks>Components** branch of the selected file.

If the **Action type** is set to **Internal action tag**, you enter a tag for the action in the **Action tag** field.

Integer Values Check


Add the **Integer Values Check** node () under a **User Input** node to force users to enter integer values for the input. An error message notifies the user that an illegal value was entered. The **Settings** window has the following section:

BOUNDS

From the **Use min value** and **Use max value** lists, choose one of the following bounds:

- **Unbounded** (the default).
- **Open bound (<)** or **Open bound (>)**: Use a minimum value or maximum value that is smaller than the value in the **Min value** field or larger than the value in the **Max value** field, respectively.
- **Closed bound (<=)** or **Closed bound (>=)**: Use a minimum value or maximum value that is smaller than or equal to the value in the **Min value** field or larger than or equal to the value in the **Max value** field, respectively.

Real Values Check


Add the **Real Values Check** node () under a **User Input** node to force users to enter real values for the input. An error message notifies the user that an illegal value was entered. The **Settings** window has the following section:

BOUNDS

From the **Use min value** and **Use max value** lists, choose one of the following bounds:

- **Unbounded** (the default).
- **Open bound (<)** or **Open bound (>)**: Use a minimum value or maximum value that is smaller than the value in the **Min value** field or larger than the value in the **Max value** field, respectively.
- **Closed bound (<=)** or **Closed bound (>=)**: Use a minimum value or maximum value that is smaller than or equal to the value in the **Min value** field or larger than or equal to the value in the **Max value** field, respectively.

Regular Expression Check

Add the **Regular Expression Check** node () under a **User Input** node to force users to enter values that only match a certain pattern. An error message notifies the user that an illegal value was entered. The **Settings** window has the following section:


REGULAR EXPRESSION

Enter the pattern in the **Regular expression** text field. Regular expressions supports a wide variety of string matching functionality, and the table below only list a few examples:

EXPRESSION	MATCHES
\w	A word character.
\d	A digit.
.	Any character.
[a-z]	Characters between a and z.
[^abc]	Not the characters a, b, and c.
\w*	Zero or more word characters.
\d+	One or more digits.

When the match fails for an entered value, an error message displays. Enter the error message to display in the **Error message** text field.

Named Group Members

Add the **Named Group Member** node () under a **User Input Group** or a **Section** to include group members by name and not by reference. Note that all named members are sensitive to name changes, so always use references if possible. The named group members always appear after the members specified in the parent node. The **Settings** window has the following section:

NAMED GROUP MEMBERS

List all group members by their names in the **Group members** column.

Variables

In this section:

- [Creating Variables](#)
- [Variables for Degrees of Freedoms](#)
- [Variable Declaration](#)
- [Variable Definition](#)
- [Dependent Variable Definition](#)
- [Dependent Variable Declaration](#)
- [Discretization](#)
- [Initial Values](#)
- [Hide in GUI](#)
- [Disable in Solvers](#)
- [Degree of Freedom Initialization](#)
- [Component Settings](#)
- [Frame Shape](#)
- [ODE States Selection](#)

Creating Variables

A variable is essentially defined by an expression of other variables, dependent variables, user input parameters, or entered values. You can use variables for many important purposes, including the following:

- To add a quantity for plotting and results evaluation.
- To simplify the writing and readability of expressions.
- To make evaluation of long expressions more efficient, especially if an expression occurs in several places.

All variables in the Physics Builder are tensors that are first declared and then defined. You typically do the declaration once in a central place.



It is possible to do several declarations for the same variable, but then it is important that they are consistent.

An example of two inconsistent declarations is when they have different dimensions, and this gives an error message when you try the physics interface in the Model Builder. A declaration does not specify anything about the value of a variable. This is what you use the variable definition for. A variable definition declares the expression or shape function for the variable, and where (a selection) the definition is valid.

Variables for Degrees of Freedoms


There are two methods to create variables to solve for (dependent variables), usually referred to as the unknowns or degrees of freedom (DOF). You can either use a dependent variable or a variable with a shape definition. The dependent variable can only be declared by an interface using the [Dependent Variable Declaration](#) under the physics interface. All dependent variables need a shape function definition to add the DOFs that the solver solves for. For this use the [Dependent Variable Definition](#) under a feature or property. If you are unsure what you need, use the default Lagrange shape function.

The other method to create DOFs is to add a shape definition for a variable declared by a [Variable Declaration](#). In contrast to the dependent variable, this variable does not show up in the Model Wizard or in the Settings window for the physics interface instance in the Model Builder. This means that the user cannot change the name of this variable. Another minor difference is that it is possible to define the scope of the variable, where the default is physics scope (`<model name>.<physics name>.<variable name>`). The dependent variable always has a component scope.






Entering Names and Expressions

Variable Declaration

A **Variable Declaration** () specifies a variety of properties for a variable. You can also right-click to add and define [Variable Definition](#), [Component Settings](#), and [Disable in Solvers](#) subnodes.

To add a **Variable Declaration** first add a **Component, Feature, Property, Physics Interface,** or **Multiphysics Interface** node then:

- From the contextual toolbar for **Component, Feature, Property, Physics Interface,** or **Multiphysics Interface,** click the **Variable Declaration** () button. For example, add a **Component** node under **Building Blocks>Components**. Then on the **Component I** toolbar, click **Variable Declaration** (). Or
- Right-click the **Component, Feature, Property,** or **Physics Interface** node and select it from the **Variables** submenu.

To find the definitions of the variable, click the **Find Declarations of this Variable** button () on the **Settings** window, or click the node and press F7, or right-click the node and choose **Search>Find Definitions**.

The **Settings** window has the following sections:

DECLARATION

Enter a **Variable name** and a **Description** to include text for the variable when shown in analysis and variable listings. You can pass the component's arguments to the description. The argument name must be placed between two # characters. For example, if you have an argument `arg.param = material`, the string `#Plot_in the _#arg.param#_frame` will be displayed as `Plot in the material frame`.

Enter a LaTeX-encoded string in the **Symbol (LaTeX encoded)** field to define a symbol (`\mu`, for example, to display the Greek letter μ).

Select a **Dimension: Scalar, Vector (3x1), Matrix (3x3), or Custom**. For **Custom**, you can specify a nonstandard dimension as an $M \times N [\times K \times L]$ array, where M , N , K , and L are integers, and K and L are optional (for example, $3 \times 3 \times 3$ if you need a tensor of rank 3 with indices of dimension 3).



Do not use tensors of rank higher than 4 due to the rapid increase in memory usage.

For **Custom** you can enter two special formats for dynamically setting the size:

- Enter `par.<name>` or `arg.<name>`, where `<name>` is a valid user input or argument, to use the size specification stored in the value of the user input named `<name>`. If the value of the user input is `3x4`, this variable is a 3-by-4 matrix. Make sure that the user input only can contain valid size strings ($N \times M \dots$) before using this format.
- Enter `size(<prefix>.<name>)`, where `<prefix>` is `par` or `arg`. It tries to evaluate the size of the given argument to the size operator. In case of `<prefix> = par`, this returns the size of the referenced user input. This can also be used to use the size of a different variable, although this usage is not recommended. The variable must be declared before this size evaluation, which cannot be guaranteed. All unknown variables return a scalar size.

The **Physical quantity** list defines the unit of the variable and is the same as a [Dependent Variable Declaration](#).



If you have defined a customized [Physical Quantity](#), choose **Locally defined** from the **Physical quantity** list and then choose the customized option from the **Link** list.

If there is no physical quantity defined, enter an SI unit for the dependent variable in the **SI unit** field. It is possible, in some contexts, to use arguments and values of user inputs to define the unit; this way you can enable dynamic units from arguments or other user inputs. There is also an operator, `evalUnit`, that you can use to parse units of known variables, typically dependent variables (example, `evalUnit(dep.u)`).



[Entering Names and Expressions](#) you can find more detailed information about how you can customize variable names with scopes and parameter values.

PREFERENCES

In this section, specify options about the variable.

Operation Between Multiple Definitions

Select an option from the **Operation between multiple definitions** list: **Replace**, **Add**, **Multiply**, **Logical or**, or **Logical and**.

Any variable definition for this variable uses the selected operator to combine multiple variable definitions of the same variable. You typically use the option **Add** when several contributing features add contributions to the same variable (heat sources, for example).

Interpret as Right-Hand Side

Select the **Interpret as right-hand side** check box if you use the variable in any right-hand side of an expression. Such variables get an extra factor

$$e^{i \cdot \text{phase}}$$

that enables plotting and animations of the entire harmonic cycle in frequency-domain simulations. It is also used to identify right-hand sides for harmonic perturbation features; see [Harmonic Perturbation](#).



Include in Load Groups

Select the **Include in load groups** check box if the variable should be affected by load grouping. To enable load groups for a feature add an **Auxiliary Settings** node and select the setting **Enable load groups**.

Matrix Symmetry for Square Matrix

If the matrix is square (for example, if you select **Matrix (3x3)** from the **Dimension** list), you can force a matrix symmetry with the options in the **Matrix symmetry for square matrix** list. The choices are **Diagonal**, **Symmetric**, and **Anisotropic** (the default), and controls the cells that the user can edit. This option also adds a check to the data field, so the user cannot enter a matrix structure that is more complex than the selected option. For example, choosing **Symmetric** does not allow an anisotropic matrix but it allows both isotropic and diagonal matrices.

Show in Plot Menu

Select the **Show in plot menu** check box (the default) if you want the variable to show up when a user clicks the **Insert expression** () and **Replace expression** () buttons in any of the **Results** nodes during a Model Builder session. Edit the **Menu** field to group the variable into a submenu. The default setting is to place the variable directly under a the menu of the physics interface.

Announce Variable to Feature Inputs

Select the **Announce variable to feature inputs** check box so the variable notifies its existence to all physics interfaces. The variable uses the selected physical quantity as an identifier. Any feature input parameter with the same physical quantity can pick up an announced variable, so that the variable can be used by the physics interface that the feature input belongs to.

An example of a feature input is any of the model input parameters in the **Model Inputs** section of a physics feature instance in the Model Builder. It is also possible to supply an extra **Match tag** to the announced variable. The predefined options here are **None** (the default), and all physical quantity field names. **None** is equivalent to an empty tag and should be used unless it is necessary to limit the matching in some sense. In such cases, enter an arbitrary but relevant tag — for example, `relative` or `absolute` to distinguish between pressure fluctuations and absolute pressure levels. You typically need physical quantity options for synchronized matching.

ADVANCED

This section contains advanced options that you do not have to change in most cases. In the **Base vector system** list you can override the base vector system specified by the

parent (for example, a feature or property) by choosing something other than the option **Same as parent**.

For 3-by-1 (vector) and 3-by-3 (matrix) tensors, the **Frame information rule** list contains the following options:

- **Automatic** (the default), which automatically adds frame information if there are split frames in the model.
- **No frame information**, which never appends any frame information for a variable.
- **Spatial frame information**, which adds spatial frame information.
- **Material frame information**, which adds material frame information.



- [Using Coordinate Systems](#)
 - [Transformation Between Coordinate Systems](#)
-


Variable Definition

A **Variable Definition** (**a=**) specifies the expression or shape of a variable and where the definition is valid.

A variable definition can exist in several places and has a slightly different user interface depending on where you use it. For example, it is not necessary to specify a variable name if the definition is a subnode to a variable declaration or user input. For a user input it is not even necessary to specify an expression because it is always set to the value entered by the user input (see [User Inputs](#)).

To add a **Variable Definition**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component I** toolbar, click **Variable Definition (a=)**.
- Right-click the **Component** node and select it from the **Variables** submenu.

To find the definitions of the variable, click the **Find Declarations of this Variable** button () on the **Settings** window, or click the node and press F7, or right-click the node and choose **Search>Find Definitions**.

The **Settings** window has the following sections:

DEFINITION

For a standalone definition (not being a subnode to a declaration, for example), enter the **Variable name**. The **Variable name** follows the rules described in [Entering Names and Expressions](#) and must match the name of a variable declaration somewhere in the same physics interface.



For a subnode definition, there is no **Variable name** field, because it always uses the same name as the parent.

Select a **Type**: **Expression**, **Shape function**, or **Available**. For **Expression**, enter an **Expression** using the rules in [Entering Names and Expressions](#).

For **Shape function** choose a **Shape function** and **Shape order**.



The shape function selection is unavailable if you use a global selection for this definition. In that case, the only type of degree of freedom (DOF) that makes sense is the ODE variable, which is the only available global DOF.



[Shape Function Variables](#) in the *COMSOL Multiphysics Reference Manual*

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES

Select the **Zero out components** check box to enforce a symmetry to spatial vectors and matrices.

If you choose **Zero out-of-plane**, the out-of-plane component is set to zero in the space dimensions 1D, 2D, 1D axial symmetry, and 2D axial symmetry. The out-of-plane component in 2D axial symmetry is the second component (the ϕ -component).

The **Zero in-plane** setting does the opposite and has no effect in 3D, for scalars, or non-spatial tensors (length other than 3).

If you choose **Property dependent**, the you can choose to zero out some components for specific values of some property values.

For each row in the table of property values, choose **Zero out-of-plane**, **Zero in-plane**, or **Keep all components** from the list under **Components to zero out** for the corresponding property value under **Property value**.

From the **Setting for other property values** list, choose **Zero out-of-plane**, **Zero in-plane**, or **Keep all components** to control how other property values are zeroed out.

Select the **Use the setting above for undefined references** check box to use the default value even when the property does not exist.

PROTECTION

Select preferences that enable protection of entered expressions. Select the **Hide expression in equation view** check box to remove the definition to display in the **Equation View** node, which is a subnode to a physics feature in the Model Builder. This disables any possibility to alter the expression; it also makes it harder to read the expression.

To further complicate reading of the expression, you can select the **Encrypt expression** check box. This turns on an encryption of the expression in the saved model file and when accessing the expression in a model file for Java code. It also encrypts the tensor expression when you compile the archive (see [Compiling an Archive](#)), so the expression in a distributed builder file (*.mphphb) cannot be read.

ADVANCED

This section is available when **Shape function** is selected as the **Type**.

Select the **Create a slit for the selected shape** check box to remove a shape function from the selection.



You typically use this for slit boundary conditions, where you want to allow a jump in the shape variable across a boundary.

Select the **Declare shapes for all frames** check box to ensure that the spatial derivatives of the shape functions exist for all existing frames. The declaration of frame-specific time derivatives of the shape functions is disabled when this check box is selected. These have to be declared manually for necessary frames.

The **Solver field type** list specifies the type of degree of freedom the shape function declares. There are few cases when this setting needs to be something different than

Normal. The option **Control** is used for optimization and sensitivity problems, whereas the **Discrete** and **Quadrature** options are used for solver events; see [Event](#).

Select a **Value type**: **Complex** or **Real**.

Dependent Variable Definition

The **Dependent Variable Definition** node ([u,v,w](#)) is used for the definition of a dependent variable, which means you specify the shape function to use and where that shape function is to be used.

To add a **Dependent Variable Definition**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component I** toolbar, click **Dependent Variable Definition** ([u,v,w](#)), or
- Right-click the **Component** node and select it from the **Variables** submenu.

The **Settings** window has the following sections:

DEFINITION

Select a **Dependent variable reference**: **Use physical quantity** (the default), **Use physical quantity + tag**, or **Variable name**.



- If you keep the default **Use physical quantity**, you must first specify the name of the dependent variable in its **Settings** window, which is selected from the **Physical quantity** list.
- For **Use physical quantity + tag** you can specify an arbitrary unique tag. Enter the tag in the **Unique tag** field. This tag is added to the end of the name of the **Physical quantity** chosen. For example, if you choose **Area (m²)** from the list and enter house in the **Unique tag** field, the name for the node (in brackets) in the Physics Builder changes to areahouse.
- For **Variable name**, enter a **Variable name** in the text field.




It is important to have a matching dependent variable setting under the physics interface; otherwise you get an error.

For **Use physical quantity** (the default), **Use physical quantity + tag**, choose a **Physical quantity** from the list. In addition to the predefined physical quantities you can use

locally defined physical quantities or physical quantities imported from an external resource:

- Select **Locally defined** from the top of the **Physical quantity** list to use one of the locally defined physical quantities, which you choose from the **Link** list. Click the **Go to Source** button () to move to the **Physical Quantity** node for the selected local physical quantity.
- Select **Imported from external resource** from the **Physical quantity** list to use physical quantities from another imported Physics Builder file, which you choose from the **Imported file** list. Click the **Go to Source** button () to move to the **Import** node for the imported Physics Builder file.

All dependent variables need a shape function declaration to add the unknowns that the solver solves for. You choose it from the **Shape function** list. If you are unsure what you need, use the default Lagrange shape function.


To find the definitions of the variable, click the **Find Declarations of this Variable** button () on the **Settings** window, or click the node and press F7, or right-click the node and choose **Search>Find Definitions**.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

For the rest of the settings, see the [Preferences](#) and [Advanced](#) sections for [Variable Definition](#).

Dependent Variable Declaration

A **Dependent Variable Declaration** node () declares a dependent variable (field variable) used by the physics interface. This node does not make the interface add any shape functions for this variable, it just declares that a dependent variable exists. All dependent variables in a physics interface must have a unique identifier (or reference tag) within an interface. In most cases, you only solve for one type of physical quantity per interface, so the physical quantity works fine as a reference tag. If you need two or

more dependent variables for the same physical quantity, it is necessary to append a unique tag to the reference tag.





You must use this reference tag when defining the actual shape functions for the variable, which you do under a feature or a property; see [Dependent Variable Definition](#).

You can also use the same shapes as a dependent variable for constraints, and then you need the reference tag to point to the correct variable. The reason for using an reference tag instead of a variable name is that the dependent variable name can be changed in the Model Builder. In the Physics Builder, you can only declare a default name.

Add these subnodes: [Dependent Variable Declaration](#), [Initial Values](#), [Component Settings](#), [Disable in Solvers](#), and [Hide in GUI](#).

To add a **Dependent Variable Declaration**, first add a **Physics Interface** or **Multiphysics Interface**, then:

- On the **Physics Interface** toolbar, click **Dependent Variable Declaration** (.
- Right-click the **Physics Interface** or **Multiphysics Interface** node and select it from the **Variables** submenu.

To find the definitions of the variable, click the **Find Declarations of this Variable** button () on the **Settings** window, or click the node and press F7, or right-click the node and choose **Search>Find Definitions**.



The **Settings** window has the following sections:

DECLARATION

Select a **Dependent variable reference**: **Use physical quantity** (the default) or **Use physical quantity + tag**.

- Keep the default **Use physical quantity** if you want to use the physical quantity as the reference.
- For **Use physical quantity + tag** you can specify an arbitrary unique tag. Enter the tag in the **Unique tag** field. This tag is added to the end of the name of the **Physical quantity** chosen. For example, if you choose **Area (m²)** from the list and enter house in the **Unique tag** field, the name for the node (in brackets) in the Physics Builder changes to areahouse.

The **Physical quantity** list defines what quantity the dependent variable represents, including the unit. As mentioned previously, the physical quantity is also used to generate the unique reference tag for the dependent variable. In addition to the predefined physical quantities you can use locally defined physical quantities or physical quantities imported from an external resource:

- Select **Locally defined** from the top of the **Physical quantity** list to use one of the locally defined physical quantities, which you choose from the **Link** list. Click the **Go to Source** button () to move to the **Physical Quantity** node for the selected local physical quantity.
- Select **Imported from external resource** from the **Physical quantity** list to use physical quantities from another imported Physics Builder file, which you choose from the **Imported file** list. Click the **Go to Source** button () to move to the **Import** node for the imported Physics Builder file.



If you choose **None** from the list it is recommended to use the option **Use physical quantity + tag** in the **Dependent variable reference** list. As **None** is not a physical quantity, enter an explicit unit in the **SI unit** field. It is possible, in some contexts, to use arguments and values of user inputs to define the SI unit; this way you can enable dynamic units from arguments or other user inputs. There is also an operator, `evalUnit`, that you can use to parse units of known variables, typically dependent variables (example, `evalUnit(dep.u)`).

The **Default variable name** field declares the default name for the dependent variable, and the **Description** field has the descriptive text for the variable shown in analysis and variable listings.

Enter a LaTeX-encoded string in the **Symbol (LaTeX encoded)** field to define a symbol (`\mu`, for example, to display the Greek letter μ).

Select a **Dimension: Scalar, Vector (3x1), Matrix (3x3)**, and **Custom**. For **Custom**, you can specify a nonstandard dimension (for example, $3 \times 3 \times 3$ if you need a tensor of rank 3 with indices of dimension 3).



Do not use tensors of rank higher than 4 due to the rapid increase in memory usage.

PREFERENCES

Select or clear the check boxes **Show in plot menu** and **Announce variable to feature inputs**. See [Preferences](#) described for [Variable Declaration](#).

DISCRETIZATION

This sections contains settings for defining the discretization levels that control the shape-function order used in the physics interface and the **Discretization** section of the physics interface instance. By default the parameter for the shape order is set automatically and includes five levels for order 1–5. You can also specify a default level (set to 2 by default). Use the **Parameter** list to specify if the discretization parameter name and description should be defined automatically (the default) or manually. This is the list in the physics interface instance that, in its automatic configuration, has the description **Element order** and has valid values **Linear**, **Quadratic**, and so on. Below the **Parameter** setting is a table with the following columns: **Level**, **Level description**, **Shape order**, and **Lower level**. This table controls the values that can be selected in the discretization parameter. Each row in this table represents a discretization level, which corresponds to a shape order for the dependent variable and an allowed value for the discretization parameter. Enter the value in the **Level** column and its description in the **Level description** column. Each level has a shape order, which you define in the **Shape order** column. The **Lower level** column's value should point to a discretization level that has a shape order that is smaller than the current one, so it needs to be a value that is present in the **Level** column. The **Lower level** setting is used by the multigrid preconditioner.

In the **Default level** list, select the default level for the discretization parameter (default value: 2, for quadratic order of the shape functions).

From the **Geometry shape order rule** list, choose a rule for determining the geometry shape order for the dependent variable. The following options are available

- **Prefer maximum order (0)**: Only let this variable choose the geometry shape order if no other variable requires a maximum order.
- **Require maximum order if first variable (1)** (the default): Let this variable require a maximum geometry shape order if it is the first dependent variable for a physics interface.

- **Require maximum order for all variables (2):** Let this variable require a maximum geometry shape order no matter where it is in the list of dependent variables.
- **From expression:** Let an expression, which you enter in the **Rule index** field that appears, evaluate to an integer representing one of the above rules. The integer for a rule appears within parentheses.

Select the **Enable accurate boundary flux option** check box to make the **Compute boundary fluxes** check box visible in the **Discretization** section of the physics interface instance. The **On by default** check box (selected by default) controls the default value of that parameter. Note that to make the computation of accurate boundary fluxes work as well as possible, it is also necessary to add **Flux Definition** nodes that define the flux for the dependent variable anywhere it is defined.


ADVANCED

This section contains advanced options that you do not have to change in most cases. In the **Base vector system** list, you can override the base vector system specified by the parent (for example, a feature or property) by choosing something other than the option **Same as parent**.



- [Using Coordinate Systems](#)
- [Transformation Between Coordinate Systems](#)

Discretization


You can add a **Discretization** node () to define a *discretization field*, which is a physics field that does not declare any model-scoped dependent variable. The setting basically is the **Discretization** section from the Dependent variable declaration node (see [Discretization](#) above), but also with a name and a parameter description declaration.

A **Variable Definition** node that defines a shape function (shape variables) can refer to a discretization field and let that field determine the shape order, complex value type, and boundary flux settings for that shape. A variable definition can also refer to a dependent variable declaration node, and thereby use that variable's shape order, complex value type, and boundary flux setting.

You can also refer to this node in expressions using the `order` prefix, which will return the currently selected order for the discretization field. There is also an `intOrder` prefix to get the integration order for a dependent variable or a discretization field.

Right-click the **Discretization** node to add a [Hide in GUI](#) subnode if required. You can, for example, add a condition for hiding the settings in the physics interface or to disable declaration of all shape variables that refer to this Discretization node.

Initial Values

A dependent variable needs an initial value for nonlinear and transient simulations. In the **Initial Values** node () you define the initial value for the dependent variable declared in the parent node. This node represents a special type of feature in the Model Builder that you get by default when you add a physics interface.

By default an **Initial Values** node is added to the [Dependent Variable Declaration](#).

INITIAL VALUE SETTINGS

In the **Settings** window of the variable you specify the initial value in the **Default initial expression** field. The expression you enter here follow the rules outlined in [Entering Names and Expressions](#), but with an exception. If you use a variable name here, you must specify the scope it uses. As an example, assume that you want to use the interface variable `init` as initial condition, then enter `phys.init` in the expression field.



Otherwise, the initial value expression becomes `init`, and at solution time this gets a component scope. Select the **Set initial value on time derivatives** check box to activate the initial values for the first time derivative. In the **Geometric entity level** list, choose the entity level that the initial value is placed on. This must match the entity level you put the dependent variable definition on.

PREFERENCES

See the [Preferences](#) section for [Variable Definition](#).

Hide in GUI

For each dependent variable added by a physics interface a few GUI components are added automatically. These GUI components let the user control properties of the dependent variable such as the name and element order. In some situations you might not want to display all these GUI components.

To restrict the visibility, right-click a **Dependent Variable Declaration** () node and choose **Hide In GUI** (). Only one such subnode per [Dependent Variable Declaration](#) node is allowed. You can also right-click an **ODE States Collection** node to add a **Hide in GUI** subnode.

The **Settings** window has the following sections:

SETTINGS

By default, the check boxes in this section, except the first one, are selected:

- Select the **Skip declarations and definitions** check box to skip the declaration of the dependent variable, which then will not be defined, and all its properties will be hidden. By default, the dependent variable is declared and defined.
- When the **Hide in initial values feature** check box is selected, the text field for setting the initial value of the dependent variable is hidden from the automatically generated initial values feature.
- The **Hide in discretization** section check box controls if the settings for choosing element order and value type of the dependent variable are hidden from the discretization property of the physics interface.
- The **Hide in dependent variables section** check box controls if the settings for setting the name of the dependent variable are hidden from the Dependent variables property of the physics interface.
- The **Hide in Model Wizard** check box controls if the settings for specifying the name of the dependent variable are hidden from the Model Wizard.

For a **Hide in GUI** subnode under an **ODE States Collection** node, only the three last check boxes above are available.

CONDITION

Add a condition on when the GUI components are hidden as specified in the **Settings** section. This condition only has effect if you select the **Condition for hiding** check box. If it is left in its default state of being not selected the GUI components as specified in the **Settings** section are always hidden. The settings for the condition work in the same way as the settings under the **User input** check box in the [Usage Condition](#) node for components.

Disable in Solvers

In some cases a physics interface may declare degrees of freedom that should only be solved for in certain situations. Use the **Disable in Solvers** (🔍) subnode to provide a condition for when a degree of freedom should not be solved for.

To add this subnode, right-click a **Dependent Variable Declaration** (📄) or **Variable Declaration** (📄) node and choose **Disable in Solvers** (🔍).

Only one subnode per [Dependent Variable Declaration](#) or [Variable Declaration](#) node is allowed.



When **Disable in Solvers** is a subnode to a **Variable Declaration** node, it is only relevant when that variable is a degree of freedom; that is, when it is defined as a shape function.

SETTINGS

The **Settings** section has two groups. The first one is enabled by the **Disable for study types** check box, and in it you can specify the study types for which the variable should not be solved. The second group is enabled by the **Additional condition for disabling** check box and makes it possible to provide an extra condition on a user input that also needs to be fulfilled in order for the variable not to be solved for. The settings for the condition work in the same way as the settings under the **User input** check box in the [Usage Condition](#) node for components.

Degree of Freedom Initialization

A variable that has a shape function definition needs an initial value for nonlinear and transient simulations. In the **Degree of Freedom Initialization** node ([u,v,w](#)) you define the initial value for such variable definitions. You can also use this to set the initial condition for dependent variables, but it is recommended that you use the [Initial Values](#) node instead. In situations where you do not want the Initial value feature in the Model Builder, you can customize the initial value definition using this node.

To add a **Degree of Freedom Initialization**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component 1** toolbar, click **Degree of Freedom Initialization** ([u,v,w](#)).
- Right-click the **Component** node and select it from the **Variables** submenu.

The **Settings** window has the following sections:

DEFINITION

You type the name of the variable you add the initial value for in the **Variable name** field. The variable name follows the rules described in [Entering Names and Expressions](#), and must match the name of a variable declaration somewhere in the same physics interface. The name should also be the name of a variable that has a shape definition,


but no errors result if this is not the case. Select the **Set initial value on time derivatives** check box to add the initial values for the first time derivative.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

The rest of the settings are described in the [Preferences](#) and [Protection](#) section for [Variable Definition](#).

Component Settings

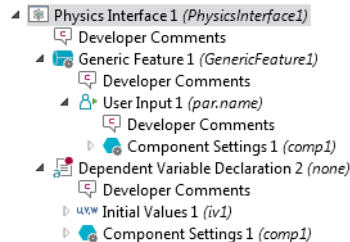
You can override the default names and descriptions for tensor elements with the **Component Settings** subnode (). This node is a valid subnode of user inputs and variable declarations and simply changes the default behavior for the parent node.



It only makes sense to use this node for non-scalar tensors, like vectors or matrices.

To add a **Component Settings**:

- 1 Add a **Physics Interface** or **Multiphysics Interface**.
- 2 Add a [User Input](#) to a feature node (for example a [Generic Feature](#)) or a variable declaration (for example, a [Dependent Variable Declaration](#)).
- 3 Right-click the node (in this example, **Generic Feature** or **Dependent Variable Declaration**) and select it from the context menu.



The **Settings** window has the following sections:

COMPONENT SETTINGS

The **Create component names by** list contains several options for creating names and description according to different rules. The first option, **Appending coordinates to the name**, is the default behavior for spatial tensors that concatenate the tensor name with the coordinate name for each tensor component:

Axy

The option **Appending indices to the name**, concatenate the tensor name with the tensor index:

A12

This is the default for non-spatial tensors. Use the option **Specifying a template** if you have a certain naming convention for the i :th component. For example, assume that you want to have the following component names:

x_vel, y_vel, z_vel

Then you enter the following template:

```
str.append(coord.i,_vel)
```

To get the descriptions

x-velocity, y-velocity, z-velocity

type

```
#coord.i#-velocity
```

The last option for vector-valued variables is **Specifying each component separately**. To get the same names and descriptions as the previous example, fill in the following to the table under the **Create component names by** list:

COMPONENT NAMES	COMPONENT DESCRIPTIONS
<code>str.append(coord.1,_vel)</code>	<code>#coord.1#-velocity</code>
<code>str.append(coord.2,_vel)</code>	<code>#coord.2#-velocity</code>
<code>str.append(coord.3,_vel)</code>	<code>#coord.3#-velocity</code>

Select the **Custom symbol** check box to control the LaTeX symbols next to an array input where each component has its own symbol. Enter the symbol using LaTeX syntax in the **Component symbol** field. Symbols are only used by user inputs, so this setting is only applicable when you have a user input of rank 1 (array) and you place it in a layout where each component has an individual text field (not table).

SCOPE SETTINGS

Select the scope (namespace) to use for the modified names in the **Scope** list. The options are **Same as variable** (the default), **Root scope**, **Component scope**, **Physics scope**, **Feature scope**, and **Device scope**. The selected scope is used for the created variable in the Model Builder.



Entering Names and Using Customized Names and Descriptions

Frame Shape



This node has been deprecated and only appears when opening Physics Builder files created in versions earlier than 5.3. This node is no longer needed with the updated moving frame functionality added in version 5.3.

Add a **Frame Shape** node (u,v,w) to define the motion of the moving frame, either as an expression or as degrees of freedom.

To add a **Frame Shape**:

- Add a **Component** node under **Building Blocks>Components** (see [Components](#)). Or add feature nodes such as a **Generic Feature** or **Domain Condition** (see [Features](#)).
- Right-click the node and select it from the **Variables** submenu.

SETTINGS

Select a **Frame motion** from the list: **Free** (the default) or **Given by expression**. Choose **Free** to define degrees of freedom for the motion. If you choose **Given by expression**, enter an **Expression** to prescribe the motion.

ODE States Selection

The **ODE States Collection** node (u,v,w) groups the ODE states that match a certain regular expression into a single ODE entity.

The ODE entity displays as a **Field** node under the **Dependent Variables** branch in the **Solver Sequence** branch when generating the solver sequence. This enables default

solver settings to be made on each collection that affects all ODE states in the collection at once.

The **Settings** window has the following sections:

DECLARATION

Enter the default name in the **Name** field. This is the name that the collection gets in the first interface added to a model. Subsequently added interfaces automatically get the first unique name. Use the `dep.` prefix and the default name to refer the collection when specifying solver defaults. The text entered in the **Description** field appear in the **Dependent Variable** section of the physics interface. In this section you can change the name of the collection.



If the user changes the name of a collection in the **Dependent Variable** section of the physics interface, it only affects the identifier of the **Field** node in the solver sequence. The names of the ODE states are not affected.

Enter a regular expression in the **Include states pattern (regular expression)** field. The collection contains all ODE states that match this pattern.

ADVANCED

Choose **Real** or **Complex** (the default) from the **Default value type** list. This becomes the default choice when solving with splitting of complex degrees of freedoms (DOFs) into real-valued and complex-valued DOFs.



[Compile Equations](#) in the *COMSOL Multiphysics Reference Manual*

Equations

You need a weak form equation (a differential equation using a weak formulation) for all features that contribute to the governing equations. The equation contributes to the weak form of representing the system of PDEs you want to solve for. You can enter the equations in these forms:

- Directly as a weak form equation, which is the most general technique to specify an equation.
- In the general form similar to the General Form PDE interface in the Model Builder.
- In the coefficient form similar to the Coefficient PDE interface in the Model Builder.
- In a boundary element method (BEM) equation form.

In this section:

- [Weak Form Equation](#)
- [General Form Equation](#)
- [Coefficient Form Equation](#)
- [Boundary Element Equation](#)
- [Shared Quantity Definition](#)

Weak Form Equation

As described in [Equations](#), you need a **Weak Form Equation** ($\int_{\Omega} \dots$) (a differential equation using a weak formulation) for all features that contribute to the governing equations.

To add a **Weak Form Equation**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component 1** toolbar, click **Weak Form Equation** (\int_{Ω}).
- Right-click the **Component** node and select it from the **Equations** submenu.

The **Settings** window has the following sections:

INTEGRAND

The content of the **Expression** field adds up as a weak form contribution to the system of equations that you later solve for in the Model Builder.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES

Clear the **Use volume factor in axial symmetry and for non-orthonormal systems** check box if you want to skip any volume factors in the weak form equation. In axial symmetry this means that you do not get the factor $2\pi r$. For more information on non-orthonormal systems.

ADVANCED

If you want to use a different base vector system than the parent feature or property, you can choose a different system in the **Base vector system** list.



- [Specifying Selections](#)
- [Entering Names and Expressions](#)
- [Using Coordinate Systems](#)
- [The Weak Form PDE in the *COMSOL Multiphysics Reference Manual*](#)

General Form Equation

As described in [Equations](#), as an alternative to the weak form you can enter the equation using the **General Form Equation** node ($\nabla \cdot \Gamma$).

To add a **General Form Equation**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component 1** toolbar, click **General Form Equation** ($\nabla \cdot \Gamma$).
- Right-click the **Component** node and select it from the **Equations** submenu.

The **Settings** window has the following sections:

EQUATION

This section displays the equation that you define by defining the coefficients in the next section.

COEFFICIENTS

There are four coefficient in a general form equation. You define each coefficient by defining a tensor expression that requires a certain dimension. The dimensions are defined by the dimension of the dependent variable, N , and the number of spatial dimension (always 3). The table below lists the dimensions for all four coefficients.

COEFFICIENT	DESCRIPTION	DIMENSION
Γ	Conservative flux	N-by-3
f	Source term	N-by-1
d_a	Damping or mass coefficient	N-by-N
e_a	Mass coefficient	N-by-N

If the dependent variable is a scalar, you can simplify the tensor dimensions by skipping all singleton dimensions. This simplifies the γ coefficient to be a spatial vector (length 3). The dependent variable can only be a tensor up to rank 1 (a vector), so use the weak form equation for dependent variables of higher ranks.

You also specify these coefficients for boundary conditions, which differs slightly from the General from PDE interface in the Model Builder. It is straightforward to convert a Flux/Source boundary condition from this interface to an equivalent representation using only the f coefficient

$$f = g - qu$$

where g and q are the coefficients in the Flux/Source boundary condition, and u is the dependent variable name.

DEPENDENT VARIABLE

In the **Variable name** text field, write the name of the dependent variable you enter the general form equation for. The variable name can also be a variable that you have defined as a shape function.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES

Clear the **Use volume factor in axial symmetry and for non-orthonormal systems** check box if you want to skip any volume factors in the weak form integration. In axial

symmetry this means that you do not get the factor $2\pi r$. For more information on non-orthonormal systems.

ADVANCED

If you want to use a different base vector system than the parent feature or property, you can choose a different system in the **Base vector system** list.



- [Variables](#)
- [Specifying Selections](#)
- [Entering Names and Expressions](#)
- [Using Coordinate Systems](#)
- [The General Form PDE in the *COMSOL Multiphysics Reference Manual*](#)

Coefficient Form Equation

As described in [Equations](#), as an alternative to the weak form you can enter the equation using the **Coefficient Form Equation** node (**-C∇u**).

To add a **Coefficient Form Equation**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component 1** toolbar, click **Coefficient Form Equation** (**-C∇u**).
- Right-click the **Component** node and select it from the **Equations** submenu.

The **Settings** window has the following sections:

EQUATION

This section displays the equation that you define by defining the coefficients in the next section.

COEFFICIENTS

There are eight coefficients in a coefficient form equation. You define each coefficient by defining a tensor expression that requires a certain dimension. The dimensions are

defined by the dimension of the dependent variable, N , and the number of spatial dimension (always 3). The table below lists the dimensions for all eight coefficients.

COEFFICIENT	DESCRIPTION	DIMENSION
γ	Conservative flux	N - by - 3
a	Absorption coefficient	N - by - N
α	Conservative flux convection coefficient	N - by - N - by - 3
β	Convection coefficient	N - by - N - by - 3
c	Diffusion coefficient	N - by - N - by - 3 - by - 3
f	Source term	N - by - 1
d_a	Damping or mass coefficient	N - by - N
e_a	Mass coefficient	N - by - N

If the dependent variable is a scalar, you can simplify the tensor dimensions by skipping all singleton dimensions. This simplifies the γ , α , and β coefficients to be spatial vectors (length 3). The dependent variable can only be a tensor up to rank 1 (a vector), so use the weak form equation for dependent variables of higher ranks.

You also specify these coefficients for boundary conditions, which differs slightly from the General from PDE interface in the Model Builder. It is straightforward to convert a Flux/Source boundary condition from this interface to an equivalent representation using only the f coefficient instead of the g coefficient.

DEPENDENT VARIABLE

Under this section you choose the dependent variable to use for the coefficient form equation. Similar to the dependent variable definition, you specify a reference to a dependent variable. In the **Dependent variable reference** list, you choose if you can use the physical quantity as the reference (choose **Use physical quantity**, which is the default), or if you have to append an unique tag (choose **Use physical quantity + tag** and enter a tag in the **Unique tag** field).

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES

Clear the **Use volume factor in axial symmetry and for non-orthonormal systems** check box if you want to skip any volume factors in the weak form integration. In axial

symmetry this means that you do not get the factor $2\pi r$. For more information on non-orthonormal systems.

ADVANCED

If you want to use a different base vector system than the parent feature or property, you can choose a different system in the **Base vector system** list.



- [Variables](#)
- [Specifying Selections](#)
- [Entering Names and Expressions](#)
- [Using Coordinate Systems](#)
- [Dependent Variable Definition](#)
- [The Coefficient Form PDE in the COMSOL Multiphysics Reference Manual](#)

Boundary Element Equation

As described in [Equations](#), as an alternative to the various equation forms based on the finite element method, you can also add a **Boundary Element Equation** node (\int_{du}) for creating an equation using the boundary element method.

To add a **Boundary Element Equation**, first add a **Component** node under **Building Blocks>Components**, then right-click the **Component** node and select it from the **Equations** submenu.

The **Settings** window has the following sections:

EQUATION

This section displays the equation that you define by defining the coefficients in the next section.

DECLARATION

In the **Operator name** field, enter an input that can be used to identically create a BEM entity. If more than one BEM Element Equation node have the same operator name, the boundary element selections will be merged in the group of each BEM selection types. In that case, all sharing coefficients and definitions between the nodes must be the same.

From the **Boundary type** list, select from the following types of boundary elements:

- **Single-sided BEM boundary**
- **Double-sided BEM boundaries with continuous field.** Use this option for boundaries, acting as BEM sources, that are adjacent to the same BEM domain on both sides and where the field is known to be continuous across the boundary.
- **Double-sided BEM boundaries with discontinuous field.** Use this option for boundaries, acting as BEM sources, that are adjacent to the same BEM domain on both sides and where a discontinuity across the boundary is allowed.
- **Edge flux.**
- **Edge gradient** (for an isolated edge)

COEFFICIENTS

There are four coefficient in a boundary element equation. You define each coefficient by defining a tensor expression that requires a certain dimension. The dimensions are defined by the dimension of the dependent variable, N , and the number of spatial dimension (always 3). The table below lists the dimensions for all coefficients.

COEFFICIENT	DESCRIPTION	DIMENSION
a	Absorption coefficient	N - by - N
α	Conservative flux convection coefficient	N - by - N - by - 3
β	Convection coefficient	N - by - N - by - 3
c	Diffusion coefficient	N - by - N - by - 3 - by - 3

If the dependent variable is a scalar, you can simplify the tensor dimensions by skipping all singleton dimensions. This simplifies the α and β coefficients to be spatial vectors (length 3).

You also specify the outward normal vector \mathbf{n} in the **Outward normal** field. The default value is the vector {root.n.1, root.n.2, root.n.3}.

VARIABLE DEFINITION

Use this section to define the dependent variable of this BEM equation. It also used to define the boundary flux and some other requirement variables for a BEM entity. Similar to the dependent variable definition, you specify a reference to a dependent variable. In the **Dependent variable reference** list, you choose if you can use the physical quantity as the reference (choose **Use physical quantity**, which is the default), or if you have to append an unique tag (choose **Use physical quantity + tag** and enter a tag in the **Unique tag** field).

You can also define a **Boundary flux variable** and a **Background variable**.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

CONDITION AT INFINITY

For the condition at infinity in an unbounded void region for a BEM model, choose **None**, **Laplace equation** (for Laplace-type equations), or **Helmholtz equation** (for wave equations).

QUADRATURE SETTINGS

Here you specify the orders for the integration (quadrature) of the various elements and contributions. This section evaluates the input parameters and pass them to the BEM entity to define the integration order of boundary elements.

FAR-FIELD APPROXIMATION

You can choose an approximation type for the far-field approximation: **None**, **ACA+**, or **ACA**. ACA and ACA+ are variants of adaptive-cross-approximation compression algorithms.

SYMMETRY


In this section you can specify symmetry in the yz , xz , and xy planes. Select **Off** for no symmetry, **Symmetry**, or **Antisymmetry**.

If you select the **Create synchronize selection inputs** check box, three new selections will be defined in the Model Builder (will be hidden in the user interface). The selection's name is the name that is defined in the **Selection input** field. The selection's entities are the entities of a geometry that lies “exactly” on the symmetry planes.


ADVANCED

If you want to use a different base vector system than the parent feature or property, you can choose a different system in the **Base vector system** list.

Shared Quantity Definition

Use a **Shared Quantity Definition** node () to define quantities that are shared among several features in the physics interface — for example, to control mesh or solver settings.

To add a **Shared Quantity Definition**, first add a **Component**, **Feature**, or **Property** node then:

- From the contextual toolbar for **Component**, **Feature**, or **Property**, click the **Shared Quantity Definition** () button.
- Right-click the **Component**, **Feature**, or **Property** node and select it from the **Variables** submenu.

The **Settings** window has the following sections:

DEFINITION

From the **Shared quantity** list, select a quantity to share (for example, **Minimum wavelength**, which can be a quantity useful for determining a suitable mesh size).

Enter a suitable name for the shared quantity variable in the **Variable name** field. Also, enter an expression for the variable in the **Expression** field.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid.



Specifying Selections

Constraints

A feature that puts some sort of constraint on a dependent variable (a Dirichlet boundary condition, for example) needs a **Constraint** node ($R=0$). A constraint forces an expression to be zero, using the expression together with a constraint-force expression. In most cases, the constraint force can be generated from the constraint expression, but the constraint node also provides the possibility to customize the constraint force. You can add an **Excluding Selection** subnode to exclude selections from the selections that the parent **Constraint** node already has.

Constraints can also be formulated as a Weak Form Equation using an extra degree of freedom. This technique is called weak constraints, and you can add a **Weak Constraint** node ($R=0$) to add such a weak constraint.

Constraint

As described in [Constraints](#), a feature that puts some sort of constraint on a dependent variable (a Dirichlet boundary condition, for example) needs a **Constraint** node ($R=0$).

To add a **Constraint**, first add a **Component** node under **Building Blocks>Components**, then:

- On the **Component I** toolbar, click **Constraint** ($R=0$).
- Right-click the **Component** node and select it from the **Equations** submenu.

DECLARATION

In the **Expression** field, you enter the constraint expression that the solver forces to zero when solving (for example, the expression $T - 293.15[\text{K}]$ makes T equal to 293.15 K). The options in the **Constraint force** list let you change how the constraint affects the governing equations. The **Automatic** option means that the user can choose between **Bidirectional**, **symmetric** and **Unidirectional** constraints in the **Constraint type** list of an instance of this feature in the Model Builder. If you choose **Customized**, a **Force expression** field becomes visible. Here you can write an expression for controlling the constraint force.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

SHAPE DECLARATION

Under the **Shape Declaration** section you choose what shape function to use for the constraint. Similar to the [Dependent Variable Definition](#), you can choose to specify a reference to a [Dependent Variable Declaration](#), and the constraint then uses the same shape function as that dependent variable. To use this option, choose **Field quantity** from the **Constraint shape function** list. With the option **Use specific shape** in this list, you can specify an arbitrary shape function for the constraint.

ADVANCED

The **Allow weak constraints** check box lets you deactivate support for weak constraints, which is selected by default. When selected, the constraint can potentially generate a weak form equation. Therefore, it is necessary to support a **Base vector system** list selection similar to the [Weak Form Equation](#). You can also select **Customized** from the **Weak expression** list instead of **Automatic**, which is the default, and then enter a custom weak-form expression for the constraint in the **Weak expression** field. In addition, if you have selected **Use specific shape** from the **Constraint shape function** list in the **Shape Declaration** section above, then you can select **Customized** from the **Lagrange multiplier name** list instead of **Automatic**, which is the default, and then enter a Lagrange multiplier base name in the **Base name** field (default: name_1m). Otherwise, the Lagrange multiplier's name is determined automatically. Under **When selected, the Lagrange multiplier may differ from the reaction forces**, you can select the **Use volume factor in axial symmetry for non-orthonormal systems** check box, which adds a $2\pi r$ volume factor in axisymmetric models.

Clearing the **Allow weak constraints** check box means that it is not possible to activate weak constraints for an instance of the feature owning this constraint in the Model Builder. Then all the settings above are disabled.

Select the **Exclude from constraint grouping** check box to make sure that the constraint is not affected by constraint grouping. This setting is only relevant when constraint grouping is activated for the physics feature. To activate constraint grouping add an **Auxiliary Settings** node to the parent feature and select the setting **Enable constraint groups**.



- [Entering Names and Expressions](#)
 - [Specifying Selections](#)
 - [Using Coordinate Systems](#)
 - [Constraint](#) in the *COMSOL Multiphysics Reference Manual*
-

Weak Constraint

[Constraints](#) can also be formulated as a [Weak Form Equation](#) using an extra degree of freedom.

To add the **Weak Constraint** node (**R=0**), first add a **Component**, **Feature**, **Property**, or usage condition then:

- On the contextual toolbar, click **Weak Constraint** (**R=0**).
- Right-click the component, feature, property, or usage condition node and select it from the **Equations** submenu.

The **Settings** window has the following sections:

INTEGRAND

The content of the **Expression** field adds up as a weak-form contribution to the system of equations that you later solve for in the Model Builder.

SELECTION

The options in the **Selection** list and **Output entities** list define the selection where this contribution definition is valid. See [Specifying Selections](#) for more information.

PREFERENCES

Clear the **Use volume factor in axial symmetry and for non-orthonormal systems** check box if you want to skip any volume factors in the weak contribution. In axial symmetry this means that you do not get the factor $2\pi r$.

ADVANCED

If you want to use a different base vector system than the parent feature or property, you can choose a different system in the **Base vector system** list. You can also specify the integration order from the Integration order list: **Automatic** (the default, which picks an integration order that is consistent with the shape function order) or

Customized; if you choose **Customized**, enter an integration order (a positive integer; default value: 2) in the **Order** field.



- [Entering Names and Expressions](#)
 - [Specifying Selections](#)
 - [Using Coordinate Systems](#)
 - [Group Members Define a Section](#)
 - [Weak Constraint](#) in the *COMSOL Multiphysics Reference Manual*
-

Excluding Selection

Add an **Excluding Selection** subnode (**R=0**) to a **Constraint** node to exclude selections already added to that node.

The options in the **Selection** list and **Output entities** list define the selection that should be excluded. See [Specifying Selections](#) for more information. The excluding selection's options are similar to the selection's options, except there is no options for a global selection, and the output entities should have a dimension that is lower than the dimension of the selection of the parent **Constraint** node.

Device Systems

In this section:

- [Creating Device Systems](#)
- [Device Model](#)
- [Port Model](#)
- [Device Constants](#)
- [Device Inputs](#)
- [Device](#)
- [Input Modifier](#)
- [Device Variables](#)
- [Device Equations](#)
- [Port](#)
- [Port Connections](#)
- [Device Feature](#)


Creating Device Systems

For certain global interfaces that do not have any distributed dependent variables (no spatial dependence), the complexity of the system often lies in the large number of dependent variables it needs. It is usually rather complicated to define a set of dependent variables to solve for, so there is a special frame work for handling such systems.

You define a hierarchy of device models that defines a set of variables and equations. A [Device Model](#) specifies a type of a device. The device model also needs ports that define how devices communicate with other devices.

A port also has a type, called a [Port Model](#). A port model defines a set of variables that take part in the communication between devices. A typical example of this rather abstract description is an electrical circuit. All resistors in a circuit belongs to the same device model, the resistor model. The resistor model defines two ports representing the two pins that a resistor has. Each resistor is a device where the two ports to communicate with other resistors and possibly other types of devices like capacitors and inductors. The port model for the port needs a voltage and a current. The voltage represents the node potential of the connection, and is the same for all ports that are connected. The current that flows out of a port must flow into the other ports, so the sum of all currents must be zero. The port model declares this behavior.


Device Model


As described in [Creating Device Systems](#), the **Device Model** () defines a set of variables and equations that corresponds to a certain device characteristic. Typical examples from the field of electrical circuits are capacitors, resistors, voltages sources, and bipolar transistors.

To a device model you can add modifiable inputs, variables, equations, other devices, and ports as child nodes. Right-click the **Device Model** node to add [Device Constants](#), [Device Inputs](#), [Device](#), [Device Variables](#), [Device Equations](#), [Port](#), [Port Connections](#), [Port Model](#), and [Usage Condition](#) subnodes. You can also add other **Device Model** subnodes.

To add a **Device Model** to **Features** ():

- 1 Under the **Building Blocks** branch click **Features**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click to add any of the available features (for example, a [Device Model Feature](#) or [Domain Condition](#)).
- 3 Right-click the node (in this example the **Device Model Feature** or **Domain Condition** node) and choose **Device Model** from the **Devices** submenu.

To add a **Device Model** to **Components** ():

- 1 Under the **Building Blocks** branch click **Components**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click **Component** ()
- 3 Right-click the **Component** node and choose **Device Model** from the **Devices** submenu.

The **Settings** window has the following sections:


DEVICE MODEL

In the **Type** field you define a unique string that identifies the device model. When you create a [Device](#), you must specify a device model for that device. The device then instantiate the device model. A device model type should be unique, but if you declare several device models with the same type, the last one is used. In the **Inherited type** field, you can enter a parent type that this device model inherits all inputs, ports, variables, and equations from. Leave this field empty to skip inheritance. A device model can also be abstract, meaning that you can only specify this type as an inherited type of another device model. A device cannot instantiate an abstract device model. Select the **Abstract model type** check box to make a device model abstract.

DEVICE INPUTS


Fill the table in this section with the device inputs that the device model supports. Each device input also needs a default value. You can also define device inputs with the device input child node. Use the [Input Modifier](#) of a device to override the default value of a device input. For valid entries in the **Dimension** column, see [Variable Declaration](#).


Port Model

As described in [Creating Device Systems](#), a **Port Model** () declares types of ports that a devices use to communicate with each other. To a port model you can only add device variables and device equations, where a variable can be declared as a flow variable. When you make a connection between two or more devices, the connection sets all non-flow variables equal. For the flow variables on the other hand, the connection sets the sum of all flow variables equal to zero. You can also right-click the **Port Model** node to add [Device Variables](#) and [Usage Condition](#) subnodes.

To add a **Port Model** to **Features** ():

- 1 Under the **Building Blocks** branch click **Features**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click to add any of the available features (for example, a [Device Model Feature](#) or [Domain Condition](#)).
- 3 Right-click the node (in this example the **Device Model Feature** or **Domain Condition** node) and choose **Port Model** from the **Devices** submenu.

To add a **Port Model** to **Components** ():

- 1 Under the **Building Blocks** branch click **Components**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click **Component** ().
- 3 Right-click the **Component** node and choose **Port Model** from the **Devices** submenu.


The **Settings** window has the following section:

PORT MODEL

In the **Type** text field enter the port type that a port instance refers to. It is also possible to let a port model inherit from another port model. To do so, enter that port model in the **Inherited type** text field.

Select the **Abstract model type** check box to make the make the port model abstract (it can then not be instantiated by a port, for example).


Device Constants

You can use these **Device Constants** () in the equations of the same device model or any device model that inherits from it. Right-click the **Device Model** node to add the **Device Constants** subnode.

DEVICE CONSTANTS

Fill the table with the constants you need for a device model. Enter the **Name** of the constant, the constant **Expression**, and the **Dimension**. The expression must be a constant with respect to device variables, but they can depend on ordinary variables declared outside the device context. For valid entries in the **Dimension** column, see [Variable Declaration](#).


Device Inputs


Use the **Device Inputs** () to declare the inputs that the device model supports. Each device input also needs a default value. These inputs can be used in the equations of the device model. Right-click the **Device Model** node to add the **Device Inputs** subnode.


DEVICE INPUTS

Fill the table with the inputs you need for a device model. A device instantiating a device model can modify the default values through the [Input Modifier](#). As an alternative to specify the inputs in this node, you can add them directly in the **Device Inputs** section of a **Device Model**. For valid entries in the **Dimension** column, see [Variable Declaration](#).



Device

A **Device** () is a device model instance that you can add under a feature or under a device model. If you add it under a feature, the program creates one device per feature instance.

To add a **Device** to **Features** ():

- 1 Under the **Building Blocks** branch click **Features**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click to add any of the available features (for example, a [Device Model Feature](#) or [Domain Condition](#)).
- 3 Right-click the node (in this example the **Device Model Feature** or **Domain Condition** node) and choose **Device** () from the **Devices** submenu.

To add a **Device** to **Components** ():


- 1 Under the **Building Blocks** branch click **Components**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click **Component** ().
- 3 Right-click the **Component** node and choose **Device** () from the **Devices** submenu.
Or add a **Device Model** and right-click to choose **Device** as a subnode.

DEVICE

To make this work properly, ensure that the device gets a unique **Name**, typically by specifying the name to `entity.tag`. The device then gets the same name as the parent feature.

In the **Type** text field enter the device model that the device is an instance of. If the Device is under a **Device Model** you can access the variables defined by the device's device model by prepending the name of the device to the name of the variable. Assume that the device model declares a variable named `R`, and the device has the name `R1`. To use this variable in the device model that the device belongs to you then type `R1.R`.


Input Modifier

A **Device Model** defines a set of device inputs that you define a default value for. To change this default value for a certain device, add an **Input Modifier** () subnode to a **Device**.

DEVICE INPUTS

Fill the table with the **Name** of the input you wish to modify, and an **Expression** for the new value.

Device Variables

You can use these **Device Variables** () in the equations of the same device model or any device model that inherits from it. Right-click a **Device Model** or **Device Model Feature** node and choose **Device Variables** from the **Devices** submenu.

DEVICE VARIABLES


Fill in the table columns and rows with the variables needed. Enter a **Name** and specify an **Expression** for a variable. This directly defines an equation for the variable. This is

equivalent to defining an equation with the right-hand side set to the variable name and the expression as the left-hand side.

In the **Elimination weight** column, specify an integer weight for the variable. The weight is used during the variable elimination process, which removes redundant state variables from the equation system. Variables with negative weights are never eliminated, while variables with higher weights are eliminated before the ones with lower weights. Typically, variables being reinitialized in events should have negative weights. In most cases, however, leaving the default value of zero for all variables (meaning that no variables are preferred) is an adequate choice.

For valid entries in the **Dimension** column, see [Variable Declaration](#).

Device Equations

Right-click a [Device Model](#) or [Device Model Feature](#) node and choose **Device Equations** () from the **Devices** submenu.


DEVICE EQUATIONS

Fill in the table with **Left-hand side** and **Right-hand side** of an arbitrary number of **Device Equations**. You can use any device variable, device input, or device constant in these equations. If you wish to use non-device variables, add a scope specifier to the variable.

Assume for example that you want to use the variable *A* declared by a variable declaration in a device equation. In an ordinary expression you just type *A*, but in a device equation, the parser assumes that this is a device variable named *A*. Instead you type `phys.A` to specify that this is a variable outside the device context.

For valid entries in the **Dimension** column, see [Variable Declaration](#).


Port

A **Port** () is a port model instance that you add to define the connections that devices use to communicate with each other. To add this subnode, right-click a [Device Model](#) or [Device Model Feature](#) node and choose **Port** from the **Devices** submenu.

PORT

In the **Name** text field you enter the name of the port. This name defines the variable names that you use to access the variables declared in the port model that you refer to in the **Type** text field. If a port model declares a variable named *v*, and you typed *a* in the **Name** text field, you access the variable *v* by typing `a.v`.

Port Connections

The **Port Connections** () defines how devices connect to each other. There are two situations when you can use port connections.

The first is when you add the port connections to a **Device Model** or **Device Model Feature**. You use this to define connections between devices that belongs to the same device model. To add this subnode, right-click a [Device Model](#) or [Device Model Feature](#) node and choose **Port Connections** from the **Devices** submenu.

The second situation is when you add port connections to a **Device**. Then you typically want to define how the ports of that device connect to another device that belongs to another feature. Right-click a [Device](#) node and choose **Port Connections** from the context menu.

PORT CONNECTION


Port Connection Added to a Device Model or Device Model Feature

Fill in the table with the names of the ports that you wish to connect, where the names in the **Connection, 1** column connects to the names in the **Connection, 2** column. This is typically ports defined by the device models that a device refers to, so the name of the port is appended to the device name. For example, you type `R.p` to access port `p` of the device named `R`.

Port Connection Added to a Device

Fill the table with the port name in the **Local connection** column, and a virtual connection in the **External connection** column. The virtual connections defines a temporary name space that identifies the port names in the right column in any port connections that connects. So lets assume that you have a parameter for a feature called `p1us` where you want the user to enter a node that port `p` connects to. Then you enter `p` in the right column, and `par.p1us` in the right. If two features exist where the user entered the same value for the parameter `plus`, the port `p` of those device gets connected.

Device Feature

The **Device Feature** () node is a special device that can only exist under a [Device Model Feature](#) node. The device is of the type that the device model feature defines, and it is placed at the same level as the device model feature in the device system hierarchy. This makes it different compared to a normal **Device** node placed under a

Device Model Feature node. Such a device is part of the device model defined by the device model feature as a subdevice to it.


To add a **Device Feature** subnode, right-click the [Device Model Feature](#) node and choose **Device Feature** from the **Devices** submenu. The **Settings** window is the same as for a [Device](#).

Operators and Functions

Operators

You can add an operator to a feature, property, or component.

To add an **Average**, **Integration**, **General Extrusion**, **Maximum**, or **Minimum** operator:

- 1 Under the **Building Blocks** branch click **Components**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click **Component** ().
- 3 Right-click the **Component** node and choose **Average**, **Integration**, **General Extrusion**, **Maximum**, or **Minimum** from the **Operators** submenu.

Defining an operation is similar to the way it works in the Model Builder but with a few differences.

- The **Operator name** follows the same rules as variables; see [Entering Names](#). You usually have to use feature scope for any operation defined by a feature; otherwise, you get a duplicate definition if you have more than one feature of that type.
- The **Show in plot menu** check box is selected by default. Clear that check box to remove the operator from the plot menus in results and probe features.
- For the **Selection** section, in the Physics Builder you cannot specify absolute selections as you can in the Model Builder. See [Specifying Selections](#) for more information.

The [Average](#), [Integration](#), [General Extrusion](#), [Maximum](#), and [Minimum](#) component couplings are supported as operations in the Physics Builder. The [Integration Over Extra Dimension](#) operator is also available.



- [Entering Names of Operators and Functions](#)
 - For more information about the other settings, see [Component Couplings and Coupling Operators](#) in the *COMSOL Multiphysics Reference Manual*
-


Functions

You can add a **Function** to a feature, property, or component. The function names follow the same rules as variables; see [Entering Names](#). You usually have to use a

feature scope for any function defined by a feature; otherwise, you get a duplicate definition if you have more than one feature of that type. Apart from the name syntax, defining a function works in exactly the same way as in the Model Builder.

The following functions are available: Analytic, Gaussian Pulse, Image, Interpolation, Piecewise, Ramp, Random, Rectangle, Step, Triangle, Waveform, and [Tensor-Valued Function](#).

To add one of the available functions:

- 1 Under the **Building Blocks** branch click **Components**.
- 2 On the **Home** toolbar, from the **Building Blocks** group, click **Component** ()
- 3 Right-click the **Component** node and choose a function from the **Functions** submenu.

In the function node's Settings windows (except for Tensor-Valued Function), the **Show in plot menu** check box is selected by default. Clear that check box to remove the operator from the plot menus in results and probe features.


SPECIAL SETTINGS FOR INTERPOLATION

In the settings for the **Interpolation** function, you can choose **User input** from the **Data source** list (in addition to **File** and **Local table**). By using the **User input** definition, the defined interpolation will read the data from the user inputs for both function and argument data. Enter an expression for the function, such as `par.time`, in the **Function expression** field and expressions for one or more arguments, such as `par.T`, in the **Arguments expression** list.




- [Entering Names of Operators and Functions](#)
 - [Functions](#) in the *COMSOL Multiphysics Reference Manual*
-

Average


The **Average** () operator is almost identical to the average operator you can add in the Model Builder. The main difference is how you specify the operator name and selections, which is described in the [Operators](#) section.

Integration

The **Integration** () operator is almost identical to the integration component coupling you can add in the Model Builder. The main difference is how you specify the

operator name and selections, which is described in the [Operators](#) section. There is also a special option in the **Frame** list: **Same as parent**. You can use it to dynamically refer the integration operator frame to the frame of the feature.

General Extrusion

The **General Extrusion** () operator is almost identical to a general extrusion coupling you can add in the Model Builder. The main difference is how you specify the operator name and selections, which is described in the [Operators](#) section.

Maximum

The **Maximum** (**MAX**) operator is almost identical to the Maximum component coupling you can add in the Model Builder. The main difference is how you specify the operator name and selections, which is described in the [Operators](#) section.

Minimum

The **Minimum** (**MIN**) operator is almost identical to the Minimum component coupling you can add in the Model Builder. The main difference is how you specify the operator name and selections, which is described in the [Operators](#) section.

Integration Over Extra Dimension

Use an **Integration Over Extra Dimension** to add an integration over an extra dimension in the physics interface. It is almost identical to the Integration Over Extra Dimension you can add in the Model Builder. The main difference is how you specify the operator name and selections, which is described in the [Operators](#) section.

To add an **Integration Over Extra Dimension**, first add a [Component](#), then right-click **Component** and add it from the context menu.




[Using Extra Dimensions](#) in the *COMSOL Multiphysics Reference Manual*



Physics Areas

In the Model Builder you create a model with the Model Wizard. After choosing geometry, you select physics interfaces from a tree view. This tree view contains categories representing physics areas at the top level, and possible subcategories in each [Physics Area](#) or [Predefined Multiphysics](#). When you create your own physics interface you can always put it in an existing category; see [Physics Interface](#) for more information about how you do this.

Physics Area


Use a **Physics Area** () node to categorize the physics in different physics areas. It collects the physics interfaces into more intuitive and quick access groups.



To add a **Physics Area**:

- On the **Home** toolbar click the **Physics Area** button () , or
- Under [Definitions Library](#) right-click **Physics Areas** () and choose it from the context menu.

The **Settings** window has the following sections:

PHYSICS AREA

This section contains a tree view of all physics areas and subcategories available from built-in resources, other areas defined in the current builder file, and areas defined in any imported file under the **External Resources** branch. To put your new physics area under a specific category in the tree, you first select that item. Then you click the **Set as parent** button () below the tree. As an alternative, you can also right-click the selected item, and choose **Set as parent** from the submenu. You find the currently selected category under the **Parent area** divider. If you do not specify a parent area, the area has **Root** as parent area.

Also control the position of the physics area in the list of areas it currently belongs to. Click the **Move up** () or **Move down** () buttons to move the area in the list.


PHYSICS AREA SETTINGS

In the **Name** field you define a unique string that identifies the physics area, which must be unique among all areas present in the Model Wizard. The text you write in the **Description** field is the text COMSOL Multiphysics displays for the physics area in the

Model Wizard. You can also specify an icon image in the **Icon** field, either by typing the name or by choosing an icon from the file system. The **Weight** field controls the position of your physics area in the list it currently belongs to. The higher the weight (a larger number), the further down the position in the list. The **Move up** and **Move down** buttons in the **Physics Area** section provide another way to change the weight.




Predefined Multiphysics

Use a **Predefined Multiphysics** () node to categorize the multiphysics interfaces in different physics areas. It collects the multiphysics interfaces into more intuitive and quick access groups. Right-click **Predefined Multiphysics** to add [Contained Multiphysics Coupling](#) and [Contained Interface \(Predefined Multiphysics\)](#) subnodes.

To add a **Predefined Multiphysics** node, under [Definitions Library](#) right-click **Physics Areas** and choose it from the context menu.

See [Physics Area](#) for the settings.

Contained Multiphysics Coupling

Use a **Contained Multiphysics Coupling** () node to create a multiphysics coupling that is contained in multiphysics interface and is added directly when a user adds that multiphysics interface in the Model Wizard, for example.

To add a **Contained Multiphysics Coupling**:

- 1 Under [Definitions Library](#) right-click **Physics Areas** and choose [Predefined Multiphysics](#) from the context menu.
- 2 Right-click **Predefined Multiphysics** and choose **Contained Multiphysics Coupling** from the context menu.


COUPLING FEATURE LINK

See [Multiphysics Coupling](#) for the settings.

COUPLINGS

See [Generic Multiphysics Coupling](#) for the settings.

Contained Interface (Predefined Multiphysics)

A **Contained Interface** () node is a link node, similar to the [Feature Link](#) and [Property Link](#) nodes.

To add a **Contained Interface**:

- 1 Under [Definitions Library](#) right-click **Physics Areas** and choose [Predefined Multiphysics](#) from the context menu.
- 2 Right-click **Predefined Multiphysics** and choose **Contained Interface** from the context menu.

SOURCE INTERFACE

The settings are the same as for [Source Interface](#).


PROPERTY DEFAULTS

The settings are the same as for [Property Defaults](#).



[Contained Interface](#)

Selections


A **Selections** () branch definition can either be an explicit selection or a selection being the result of a Boolean operation. Right-click the [Definitions Library](#) node to add this from the context menu.

Right-click to add [Selection](#), [Selection Filter Sequence](#), and [Extra Dimension Selection](#) subnodes. There are also other features available as subnodes to the **Selection Filter Sequence**: [Override Rule Filter](#), [Selection Component Filter](#), and [Multiphysics Coupling Selection Filter](#).




- [Specifying Selections](#)
 - [Named Selections](#) in the *COMSOL Multiphysics Reference Manual*
-

Selection

Add the **Selection** node () to create a selection that you can use to define other selections.

You can either link to a selection component directly or as part of an operation between selections such as a union, intersection, or difference. Selection components can be referenced from other selection components, or from any item that has a selection section.


To add a **Selection**:

- On the **Home** toolbar click the **Selection** () button, or
- Under [Definitions Library](#), right-click the [Selections](#) node and choose **Selection** from the menu.



- [Specifying Selections](#)
 - [Named Selections](#) in the *COMSOL Multiphysics Reference Manual*
-

Selection Filter Sequence

The **Selection Filter Sequence** node () defines the entities that a physics feature is applicable on. A feature can reference such a definition by choosing **From sequence** in

the **Applicable entities** list in the **Settings** section of a feature’s window (see [Generic Feature](#)).

To add a **Selection Filter Sequence**, under **Definitions Library** right-click the [Selections](#) node and choose **Selection Filter Sequence** from the menu.

The function of a selection filter sequence is defined by right-clicking **Selection Filter Sequence** and to add [Override Rule Filter](#), [Selection Component Filter](#), or [Multiphysics Coupling Selection Filter](#) subnodes.

The **Settings** window has the following section:


SETTINGS

The table displays a list of all filter subnodes under this **Selection Filter Sequence** node. For each row select **Take complement** and choose an **Operation with next** to use with the next filter: **Intersection** (the default) or **Union**. The choices in this table produce a complete filter sequence without precedence between operations. For example, if the table has the following three rows:



FILTERS	TAKE COMPLEMENT	OPERATION WITH NEXT
Filter 1	X	Intersection
Filter 2	√	Union
Filter 3	X	Intersection

The contents of this table represent the following logical expression: Filter 1 *and not* (Filter 2) *or* Filter 3.

Override Rule Filter

An **Override Rule Filter** node () defines a single condition of override rules (specified through their names) and a list of applicable entities; see [Override Rule](#). The condition represents a geometry selection given by the selection of the features that belong to the given override rules. The specified domain types define what kind of selection it should be — for example, exterior boundaries to the feature’s selection.

To add an **Override Rule Filter**:


- 1 Under [Definitions Library](#) right-click the [Selections](#) node () and choose **Selection Filter Sequence** from the context menu.
- 2 Right-click **Selection Filter Sequence** () and choose **Override Rule Filter** from the context menu.

The **Settings** window has the following section:



SETTINGS

Fill the **Override rule names** table with the ones to use for this filter. Then add the **Allowed entity types** — **Active**, **Inactive**, **Exterior**, **Symmetry axis**, **Interior**, or **Geometry** — to the list. For more information on entity types, see [Selection Terminology](#).

Selection Component Filter

A **Selection Component Filter** node () filters the selection by using selection components. It is basically a reference to a [Selection](#). This filter is versatile and can be used to, for example, find the selections that are adjacent to features of certain types. The output selection of this filter can be of an entity level that is higher than that of the physics feature that should use the selection. In this case the entity level of the selection is lowered by taking the adjacent entities of the selection.

To add a **Selection Component Filter**:

- 1 Under [Definitions Library](#) right-click the [Selections](#) node () and choose **Selection Filter Sequence** from the context menu.
- 2 Right-click **Selection Filter Sequence** () and choose **Selection Component Filter** from the context menu.

The **Settings** window has the following section:

SELECTION


Choose a **Selection**: **Operation on sibling-feature selections** (the default) or **From definitions library**.

For **Operation on sibling-feature selections** also choose an **Operation type**: **Union**, **Intersection**, **Difference**, or **Complement** and enter information in the table.





- [Specifying Selections](#)
 - [Named Selections](#) in the *COMSOL Multiphysics Reference Manual*
-

Multiphysics Coupling Selection Filter

A selection filter sequence that uses **Multiphysics Coupling Selection Filter** nodes () can only be used by a multiphysics coupling feature. Otherwise, this filter works almost

exactly like the **Selection Component Filter** but it also has a setting to specify the input selection to the filter.

To add a **Multiphysics Coupling Selection Filter**:

- 1 Under **Definitions Library**, right-click the **Selections** node () and choose **Selection Filter Sequence** from the context menu.
- 2 Right-click **Selection Filter Sequence** () and choose **Multiphysics Coupling Selection Filter** from the context menu.

The **Settings** window has the following section:

SETTINGS


Choose an **Input selection**: **Intersection of coupled physics selections** (the default) or **Union of coupled physics selections**:


- **Intersection of coupled physics selections** restricts the selection to the entities that are selected by all physics that are coupled to the coupling feature.
- **Union of coupled physics selections** restricts the selection to entities that are selected by any of the physics that are coupled to the coupling feature.



- [Specifying Selections](#)
 - [Named Selections](#) in the *COMSOL Multiphysics Reference Manual*
-

Extra Dimension Selection

Use the **Extra Dimension Selection** node () to add selections to an extra dimension for the connection to an ordinary space-dimension part of the model.

To add a **Extra Dimension Selection**, under **Definitions Library** right-click the **Selections** node () and choose it from the context menu.

SELECTION

See [Specifying Selections](#) for information.

SOURCE EXTRA DIMENSION

Choose an option from the **Specify extra dimension list**: **By reference** (the default) or **By tag suffix**. For **By reference**, select an option from the **Links from** and **Link** lists. For **By tag suffix**, enter a **Tag suffix**.

SELECTION ON EXTRA DIMENSIONAL GEOMETRY

Select a **Type**: **Predefined** or **User defined**.


For **Predefined**, select an existing predefined selection from the **Selection** list.

For **User defined**, choose an **Entity dimension**: **Domain** or **Boundary**. From the **Entities** list, choose **Specified** or whether **All entities** are used. For **Specified** enter the **Entity indices** in the text field.



[Using Extra Dimensions](#) in the *COMSOL Multiphysics Reference Manual*

Extra Dimensions


The **Extra Dimensions** branch () under the [Definitions Library](#) is used to create extra dimensions for use in a physics interface.

Right-click **Extra Dimensions** to add [1D Interval](#), [Multiple 1D Intervals](#), [2D Rectangle](#), [2D Circle](#), and [3D Sphere](#) nodes.



- [Extra Dimension Link](#), [Integration Over Extra Dimension](#), [Extra Dimension Selection](#)
- [Using Extra Dimensions in the COMSOL Multiphysics Reference Manual](#)

1D Interval

Use the **1D Interval** node () to add a 1D extra dimension as a 1D interval.

To add a **1D Interval**, under [Definitions Library](#) right-click the [Extra Dimensions](#) node and choose **1D Interval** from the context menu.

PARAMETERS

Specify parameters by filling in the columns **Name**, **Description**, and **Default expression**.

EXTRA DIMENSION SPECIFICATION


Enter a **Tag suffix**.

In the table you can edit the **First**, **Second**, and **Third Default coordinate names**, which are **xd1**, **xd2**, and **xd3**, respectively.

INTERVAL

Enter a value for the **Left endpoint**, **Right endpoint**, and the **Number of mesh points**.

Multiple 1D Intervals

Use the **Multiple 1D Intervals** node () to add a 1D extra dimension as multiple 1D intervals.


To add a **Multiple 1D Intervals** under [Definitions Library](#) right-click the [Extra Dimensions](#) node and choose **Multiple 1D Intervals** from the context menu.

See [1D Interval](#) for the **Parameters** and **Extra Dimension Specification** settings.

MULTIPLE INTERVALS

Enter a value for the **Points** and **Mesh points**.

2D Rectangle

Use the **2D Rectangle** node () to add 2D extra dimension as a rectangle.

To add a **2D Rectangle** under [Definitions Library](#) right-click the [Extra Dimensions](#) node and choose **2D Rectangle** from the context menu.

See [1D Interval](#) for the **Parameters** and **Extra Dimension Specification** settings.


RECTANGLE

Under **Size**, enter values for the **Width** and **Height**.

Under **Position**, choose the **Base** position: **Center** (the default) or **Corner**. Enter values for **x** and **y**.

Under **Mapped mesh**, enter values for the **Mesh points in x direction** and **Mesh points in y direction**.

2D Circle

Use the **2D Circle** node () to add 2D extra dimension as a circle.

To add a **2D Circle** under [Definitions Library](#) right-click the [Extra Dimensions](#) node and choose **2D Circle** from the context menu.

See [1D Interval](#) for the **Parameters** and **Extra Dimension Specification** settings.

CIRCLE

Enter a radius for the circle in the **Radius** field.

Under **Position**, enter values for the circle's center coordinates in the **x** and **y** fields.

Under **Object type**, choose **Solid** or **Curve** from the **Type** list for a filled circle or a circle as a curve only.


Under **Element size parameters**, enter values for the following element-size parameters:

- The **Maximum element size**
- The **Minimum element size**

- The **Maximum element growth rate**
- The **Curvature factor**
- The **Resolution of narrow regions**

See the documentation in the *COMSOL Multiphysics Reference Manual* for the **Size** node in a mesh sequence for more information about these parameters.

3D Sphere

Use the **3D Sphere** node () to add 3D extra dimension as a sphere.

To add a **3D Sphere** under **Definitions Library** right-click the **Extra Dimensions** node and choose **3D Sphere** from the context menu.

See **1D Interval** for the **Parameters** and **Extra Dimension Specification** settings.

S P H E R E

Enter a radius for the circle in the **Radius** field.

Under **Position**, enter values for the sphere's center coordinates in the **x**, **y**, and **z** fields.

Under **Object type**, choose **Solid** or **Surface** from the **Type** list for a filled sphere or a sphere as a surface only.

Under **Element size parameters**, enter values for the following element-size parameters:

- The **Maximum element size**
- The **Minimum element size**
- The **Maximum element growth rate**
- The **Curvature factor**
- The **Resolution of narrow regions**

See the documentation in the *COMSOL Multiphysics Reference Manual* for the **Size** node in a mesh sequence for more information about these parameters.


Auxiliary Definitions

In this section:



- [Material Property Group](#)
- [Material Property \(Auxiliary Definitions\)](#)
- [Physical Quantity](#)
- [Override Rule](#)
- [Plot Menu Definition](#)
- [Equation Display \(Auxiliary Definitions\)](#)

Material Property Group

When adding a [Material Property](#) node as an input under a feature or property, you can choose a basic property type or, among other options, a property type defined locally in the current Physics Builder file. The latter choice refers to a **Material Property Group** node.


A **Material Property Group** () contains a set of material properties, which can be selected from a list of basic material properties or be customized. When defining a material property group, physics interfaces can use its properties through a material property, and you can also add values to a material library database.


To add an **Material Property Group** node:

- On the **Home** toolbar, click **Material Property Group** () , or
- Right-click the **Auxiliary Definitions** node () and select it from the context menu.

The **Settings** window has the following sections:

PARENT CATEGORIES FOR GROUPS

Each material property group can be categorized under one or several group categories. This simplifies access in the **Material** node in the Model Builder. To create a new parent for the current property group, click the **Create New Parent** button () under the **Root** tree.

Click the **Set As Parent** button () to choose an existing category as parent. For created categories, you can edit the name in the field under **Category**, enabled when the **Create New Parent** button is clicked.

The tree also contains built-in property groups (●) and categories (🏢), for example the **Solid Mechanics** property group. Available groups are based on your license.



You can use a built-in category as a parent, but the name cannot be changed.

MATERIAL PROPERTY GROUP

In the **Name** field define a string that identifies the property group. It must be unique among all property groups available to materials. COMSOL Multiphysics displays the text entered in the **Description** field for the property group in the **Material properties** section of the **Material** node in the Model Builder.



[Materials](#) in the *COMSOL Multiphysics Reference Manual*

Material Property (Auxiliary Definitions)

The **Material Property** subnode (**P_i**) declares a new property for a node. It is similar to the [Material Property](#) node used in features or properties but with fewer options.

To add an **Material Property** node right-click **Material Property Group** and select it from the context menu.

The **Settings** window has the following section:



MATERIAL PROPERTY

Select a **Property type**: **Custom material property** (the default) or **Basic material property**. For **Basic material property**, select a material **Physical quantity**. For **Custom material property** see the [Variable Declaration](#) node for the settings.

Physical Quantity

Add a **Physical Quantity** node (🏢) to a Physics Builder file to define variable declarations and definitions of material properties. These are in addition to the predefined physical quantities available.

To add an **Physical Quantity** node:

- On the **Home** toolbar, click **Physical Quantity** () , or
- Right-click the **Auxiliary Definitions** node () and select it from the context menu.



To use this customized physical quantity when defining a [Variable Declaration](#), [Dependent Variable Definition](#), [Dependent Variable Declaration](#), [Material Property \(Auxiliary Definitions\)](#), or [Material Property](#) node (or any node with the Physical quantity list), select **Locally defined** from the **Physical quantity** list (the top item in the list), and then select the added physical quantity from the **Link** list.

The **Settings** window has the following sections:


PHYSICAL QUANTITY

- Enter a **Name** for the physical quantity (`seebeck_coefficient`, for example).
- Enter a **Description** for the physical quantity (`Seebeck coefficient`, for example).
- Enter a symbol for the property in the **Symbol (LaTeX encoded)** field (`S`, for example). You can use LaTeX syntax for Greek letters, subscripts, superscripts, and mathematical symbols if desired. See [Mathematical Symbols and Special Characters](#) in the *COMSOL Multiphysics Reference Manual*.
- Define the dimension of the physical quantity by typing the corresponding **SI unit** (`V/K`, for example).
- The **Dimension** list determines the dimension of the physical quantity: Use the default value, **Scalar**, for a scalar quantity or select **Vector (3x1)** for a vector-valued quantity, **Matrix (3x3)** for a tensor quantity, or **Custom** to type a dimension in the *NXM* format.

PREFERENCES

To specify that the physical quantity also is a material property, select the **Is material property** check box.



Override Rule

Add an **Override Rule** node () , to define new override rules in addition to the exclusive and contributing types.

When creating a model in the Model Builder, you add several feature instances as children to a physics interface. If these features support selections, they obey certain rules about how a feature of one type overrides or gets overridden by another feature. This is not to be confused with features being applicable to a certain entity, for example some features only apply on interior boundaries.


The overriding of selections is based on grouping features into override types, and the rules apply between override types. The four standard built-in override types are **Exclusive**, **Contributing**, **Override features of same type**, and **Never overridden**. In addition to the built-in override types you can create custom override types. The names and behavior of override types are defined in an **Override Rule** node.

To add an **Override Rule** node:

- On the **Home** toolbar, click **Override Rule** (), or
- Right-click the **Auxiliary Definitions** node () and select it from the context menu.

You can access all added rules from any feature in the same file, or in any other file that has the former file added as an **Import** node under the **External Resources** branch.

OVERRIDE RULE

The table that displays all override types as a matrix with the types as rows, and what types they override as columns. A selected cell means that the row type overrides the column type. The predefined rules are **exclusive**, which overrides other exclusive or contributing nodes, and **contributing**, which does not override other exclusive or contributing nodes. You can add and define new override types by clicking the **Add** () button. The override rule table defines the override behavior of the included override types. There are two things to keep in mind when editing these tables:

- Firstly, the defined override rules need to obey the following rule: If type A overrides type B and type B overrides type C, then type A must also override type C.
- Secondly, an override table does only define the override behavior of the types included in the table. A physics interface can use override types defined in different tables for its features. In this situation the override behavior between some features might not be well defined. To remedy this, you can provide the missing information in an override table that is available in the **Physics Interface** node's **Override Rule** section. This table contains all the override types used by the features under the physics interface.


The **Settings** window also allows for setting a name for the override rule. This is done by changing the **Identifier** setting from **Default** to **Customized** and then typing a name


in the **Name** field that is made visible. The reason for setting the name is that the override rule may then be used in an [Override Rule Filter](#) node.



Physics Exclusive and Contributing Node Types in the *COMSOL Multiphysics Reference Manual*

Plot Menu Definition

A **Plot Menu Definition** node () defines the plot menus that you can group related variables into (see [Variable Declaration](#)).



A plot menu shows up as a submenu in the **Replace Expression** () menu of the result nodes in the Model Builder. Specify another menu name as parent if you need several levels of submenus; otherwise leave the parent field blank.



A blank parent field means that the menu is a submenu to the menu of the physics interface.

The plot menu that a variable belongs to is specified under the variable declaration node below the **Show in plot menu** option. Using plot menu definitions you can create a structure where related variables are grouped into separate menus.

To add a **Plot Menu Definition** node:


- On the **Home** toolbar, click **Plot Menu Definition** () , or
- Right-click the **Auxiliary Definitions** node () and select it from the context menu.


The **Settings** window has the following section:

MENUS

In the table under each column, enter the **Name**, **Description**, and **Parent** to define a new menu.

Equation Display (Auxiliary Definitions)

Add an **Equation Display** node () to define a display of an equation that you can type using LaTeX syntax.

To add an **Equation Display** node, on the **Home** toolbar, click **Equation Display** () . Or right-click the **Auxiliary Definitions** node and select it from the context menu.

DECLARATION

Enter a **Name**.


EQUATION

Add the LaTeX syntax to the **Enter equation in LaTeX syntax** field.




You can also add a different kind of [Equation Display](#) node (Δu) to many other features, for example, components, physics interfaces, multiphysics interfaces, features, or properties. This is different from this node and it is selected from the context menu for these specific features.

Mesh Defaults

Use a **Mesh Defaults** node  to customize the mesh generation if your physics controls the mesh. One **Mesh Defaults** node is available for each physics or multiphysics interface.

To add a **Mesh Defaults** node, first add a [Physics Interface](#) or [Multiphysics Interface](#) then,


- On the **Physics Interface** toolbar, click the **Mesh Defaults** button ,
- Right-click the **Physics Interface** or **Multiphysics Interface** node and select it from the context menu.

Right-click the **Mesh Defaults** node to add a [Usage Condition](#) subnode if you need to vary the mesh generation depending on space dimension or user input value. Also right-click to add a [Mesh Size](#) subnode.



Meshing in the *COMSOL Multiphysics Reference Manual*.

Mesh Size

In the **Mesh Size** node () specify how the predefined mesh sizes generate meshes. If the physics interface is the controlling interface for the mesh, these settings are used when automatically generating the mesh each time it is solved. Right-click the [Mesh Defaults](#) node to add a **Mesh Size** node.


MESH SIZE SUGGESTION

Fill in the table with suitable values for the **Size** settings:

- **Maximum element size**
- **Minimum element size**
- **Curvature factor**
- **Maximum element growth rate**
- **Resolution of narrow regions**

Also define these values for each of the predefined mesh sizes: **Extremely fine**, **Extra fine**, **Finer**, **Fine**, **Normal**, **Coarse**, **Coarser**, **Extra coarse**, and **Extremely coarse**.

Study and Solver Defaults


To define **Study/Solver Defaults** () for a physics or multiphysics interface, create a solver sequence, or part of a solver sequence. The sequence is then used as a solver suggestion when solving a model that includes the interface.



There are many options available when creating a solver default, but the most important rule is to try to specify as little as possible.

One **Study/Solver Defaults** node is available for each physics or multiphysics interface.

To add a **Study/Solver Defaults** node, first add a [Physics Interface](#) or [Multiphysics Interface](#), and then use one of these options:

- On the **Physics Interface** toolbar, click the **Study/Solver Defaults** button () , or
- Right-click the **Physics Interface** or **Multiphysics Interface** node and select it from the context menu.

Right-click the **Study/Solver Defaults** node to add a variety of subnodes to further define it. Add a [Usage Condition](#) node, for example, to specify one default for stationary problems and another for time-dependent problems.

For most of the nodes in the actual solver sequences this works in exactly the same way as in the Model Builder. This section only describes the nodes that differ significantly or do not have a counterpart in the Model Builder.


In this section:

- [Field](#)
- [Absolute Tolerance](#)
- [Segregated Step](#)
- [Outer Job Parameters](#)
- [Eigenvalue Transform](#)
- [Study Sequence](#)
- [Stationary](#)
- [Time Dependent](#)



[Introduction to Solvers and Studies](#) in the *COMSOL Multiphysics Reference Manual*.

Field

Use the **Field** node () to set the default scaling for a dependent variable. Right-click the **Study/Solver Defaults** node to add this node.

The **Settings** window has the following sections:

GENERAL

In the **Dependent variable reference** and **Physical quantity** lists, you specify what dependent variable to change the scaling for. See [Dependent Variable Declaration](#) for the settings information.


SCALING

Set the scaling method in the **Method** list.





See [Dependent Variables](#) in the *COMSOL Multiphysics Reference Manual* for more about scaling.

Absolute Tolerance

Use the **Absolute Tolerance** node () if you want to set the default absolute tolerance for a dependent variable.

To add an **Absolute Tolerance** node:

- 1 Right-click **Study/Solver Defaults** () and select **Time-Dependent Solver** () from the **Solvers** menu, then
- 2 Right-click **Time-Dependent Solver** to add **Absolute Tolerance** as a subnode.

The **Settings** window has the following sections:

GENERAL

In the **Dependent variable reference** and **Physical quantity** lists, you specify what dependent variable to change the absolute tolerance for. See [Dependent Variable Declaration](#) for more information about the dependent variable reference. Use the option **Use field name** in the **Dependent variable reference** list to specify a variable defined as a degree of freedom.


ABSOLUTE TOLERANCE

Set the absolute tolerance method in the **Method** list.



See [Time-Dependent Solver](#) in the *COMSOL Multiphysics Reference Manual* for more information about setting absolute tolerances.

Segregated Step

The **Segregated Step** node () specifies the corresponding node in the solver sequence.

To add a **Segregated Step** node:

- 1 Right-click **Study/Solver Defaults** () and select **Segregated** (), then
- 2 Right-click **Segregated** to add **Segregated Step** as a subnode.



GENERAL

Specify the variables to be included in the segregated step by adding them to the **Variables** table. For a dependent variable add it to the table by typing the name of the physical quantity into the **Dependent variable reference** column or by typing the default name of the dependent variable into the **Shape variable name** column. Note that the name of the physical quantity needs to be spelled out in lower case letters, for example, `electricpotential`. To include an ordinary variable that is defined as a shape function simply type its variable name in the **Shape variable name** column.



For information about the **Method and Termination** settings, see [Segregated Step](#) in the *COMSOL Multiphysics Reference Manual*



Outer Job Parameters

If the interface uses predefined parameters that must be in an outer sweep, add these to the **Outer Job Parameters** node (). To add an **Outer Job Parameters** node, right-click **Study/Solver Defaults** () and select it from the context menu.

OUTER JOB PARAMETERS

Enter the parameters in the **Parameter names** column.

Eigenvalue Transform

Use the **Eigenvalue Transform** node () to define a transform between the internal eigenvalue computed by the eigenvalue solver and a more user-friendly quantity. To add an **Eigenvalue Transform** node, right-click **Study/Solver Defaults** () and select it from the context menu.

The **Settings** window has the following sections:



DECLARATION

In the **Study types** list, choose among study types that use the eigenvalue solver. Enter a unique name and description for the transform in the **Name** and **Description** text fields.

RELATION BETWEEN EIGENVALUES

Enter the name of the variable name in the **Eigenvalue name** text field. This name is used in the transform definitions. In the **Transform to lambda** text field, enter the expression that computes the internal eigenvalue, λ , when the transform's eigenvalue is known. Enter the inverse transform in the **Transform from lambda** text field.

Study Sequence

A study sequence is a sequence of study steps that a user can choose in the last step of the Model Wizard. Use the **Study Sequence** node () to define such predefined study sequences. To add a **Study Sequence** node, right-click **Study/Solver Defaults** () and select it from the context menu.


Right-click **Study Sequence** to add the study steps that are part of the sequence. For example, [Stationary](#) or [Time Dependent](#). For other study steps, see the *COMSOL Multiphysics Reference Manual*.



It is only possible to add the study steps that the physics interface supports. If you add a sequence under a physics interface component, all available study steps can be added to the sequence, but it is then not allowed to link to this component from an interface that does not support the study steps in the list.

The available [Identifiers](#) and [Restrictions](#) sections are described for the [Physics Interface](#) node.

Stationary

The **Stationary** () study is used when field variables do not change over time, such as in stationary problems.


To add a **Stationary** node:

- 1 Right-click **Study/Solver Defaults** () and select **Study Sequence** (), then
- 2 Right-click **Study Sequence** to add **Stationary** as a subnode.



For settings details, see [Stationary](#) in the *COMSOL Multiphysics Reference Manual*.

Time Dependent

The **Time Dependent** () study is used when field variables change over time.


To add a **Time Dependent** node:

- 1 Right-click **Study/Solver Defaults** () and select **Study Sequence** (), then
- 2 Right-click **Study Sequence** to add **Time Dependent** as a subnode.




For settings details, see [Time Dependent](#) in the *COMSOL Multiphysics Reference Manual*.

Result Defaults

Add a **Result Defaults** node () to a physics interface or multiphysics interface to create customized plot groups that are added to a model you solve for it in the Model Builder. One **Result Defaults** node is available for each physics or multiphysics interface.

To add a **Result Defaults** node, first add a [Physics Interface](#) or [Multiphysics Interface](#) then,

- On the **Physics Interface** toolbar, click the **Result Defaults** button , or
- Right-click the **Physics Interface** or **Multiphysics Interface** node and select it from the context menu.

Right-click the **Result Defaults** node add plot groups, plots, derived values, and data sets. Add the [Usage Condition](#) node to vary the result defaults. Also right-click to add the [Plot Defaults](#) subnode and further define default expression of variables to use for newly created plots of certain types.



It is common to specify one default for 3D and another for 2D, because you usually use different plot groups and different plot types.



Most of the nodes you can add to **Result Defaults** work in the same way as in the Model Builder. This section only describes the nodes that differ significantly or do not have a counterpart in the Model Builder.



In the **Settings** window form most of the nodes you can add to **Result Defaults**, there is **Settings** section where you can define a description for the node. By default, it uses the default name of the node. Select the **Description** check box to enter and use another node description.

The **Settings** window has one section:

SETTINGS


Select the **Plot in spatial frame** check box if you want the result plots to be plotted in the spatial frame instead of the default material frame. Select it if plotting for the parent physics interface should be done in the spatial frame by default.

Most of the nodes in a result defaults setting work in exactly the same way as in the Model Builder. This chapter only describes the nodes that differ significantly or do not have a counterpart in the Model Builder. This section includes all the information about the [Plot Defaults](#).



Results Analysis and Plots in the *COMSOL Multiphysics Reference Manual*

Plot Defaults


All child nodes to the **Plot Defaults** branch () define the default expression of variables to use for newly created plots of certain types.

To add a **Plot Defaults** subnode, first add a [Result Defaults](#) node then right-click the **Plot Defaults** node to select the following supported types of plots from the context menu: **Default Scalar Plot**, **Default Vector Plot**, **Default Deformation Plot**, **Default Multi Scalar Plot**, or **Default Plot Parameters**.




Entering Names and Expressions


DEFAULT SCALAR PLOT

For the **Default Scalar Plot** () , enter the expression in the **Expression** field. The expression is used in all new plots created by the user that requests a scalar value — for example, a surface plot. Enter the **Description** for the expression.


DEFAULT VECTOR PLOT

For the **Default Vector Plot** () , enter a valid vector variable name in the **Variable name** field. The components of the vector are used in all new plots created by the user that requires a vector quantity — for example, an arrow plot.


DEFAULT DEFORMATION PLOT

For the **Default Deformation Plot** () , enter a valid vector variable name in the **Variable name** field. The components of the vector are used in all new deformation plots created by the user. Deformation plots show a deformed shape, typically using a displacement vector as the vector variable.

DEFAULT MULTI SCALAR PLOT

For the **Default Multi Scalar Plot** () , fill the table with expressions and descriptions in the **Expression** column and **Description** column. It is used in all new plots created by the user that requests several scalar values — for example, a global plot.

DEFAULT PLOT PARAMETERS

For the **Default Plot Parameters** () , fill the columns **Name**, **Value**, and **Description** in table for the parameters that you need in your new plots. Any result node in the Model Builder that has the **Parameters** table is filled with the values entered here.



[Results Analysis and Plots](#) in the *COMSOL Multiphysics Reference Manual*

Migration

In this section:

- [About Backward Compatibility](#)
- [Version](#)
- [Physics Interface \(Migration\)](#)
- [Feature \(Migration\)](#)
- [Property \(Migration\)](#)
- [Change Type](#)
- [Rename Inputs](#)
- [Migration Links](#)

About Backward Compatibility

Migration or backward compatibility has to be considered in situations when you make changes to your physics interface design but still want users of this interface to use COMSOL Multiphysics model files created in the old version of the interface. If the migration is done properly, the old model file is corrected when opened, and the user can continue working with it without any problems. Otherwise, the user can get a series of error messages, complaining about invalid names and values, for example.

Another situation occurs if users have saved their models as model files for Java. The change you made in the interface can then break that model file for Java, so the user cannot execute it. It is possible to define migration for this as well, so generated files can execute although they contain Java code in an old syntax. This procedure is referred to as compatibility for the Model Object API. Generally, API migration is more complex to handle, and there are situations when you cannot avoid breaking old model files for Java.

There are settings that you can change without any need for migration. There are also settings that you cannot handle with migration at all — for example, if you change the



list of supported space dimensions. The table below summarizes some common changes and whether you should consider migration for the change.

CHANGE OPERATION	OPEN MODEL MIGRATION	API MIGRATION
Change type	Yes	Yes
Rename input	Yes	Yes
Remove feature	No	Not possible
Remove input	No	Not possible
Remove input group	No	No
Change description	No	No
Change expression	No	No
Change icon	No	No
Change symbol	No	No
Remove supported space dimension	Not possible	Not possible
Add supported space dimension	No	No
Remove supported study types	Not possible	Not possible
Add supported study types	No	No


For the most common operations that do require migration, you automatically get the proper migration operation included under the last version. This only works if there is a version when you did the change. There can be occasions when you do not want to register a migration operation for every change you do. To turn off the automatic migration, click the **Migration** node and then clear the **Add migration operations automatically** check box in the **Migration settings** section.

The nodes for the migration operations appear in a **Version** under **Building Blocks** and under the following container nodes: **Components**, **Properties**, and **Features**.


Version

Right-click the **Migration** node () to add a **Version** branch (), which contains migration operations from an older version to a newer version. The old version is the current version at the time when the version node was created. It then handles all migration operations to the current version until you create a new version node. This means that the last node in the list of versions performs the migration to the current version. All other nodes perform the migration from an older version to the next version node.


Physics Interface (Migration)

A **Physics Interface** node () contains all migration operations for the interface's settings, and for the features that you use in the interface. If an interface uses feature links to a feature under the **Building Blocks** branch, there must be a feature link node under the physics interface. You handle the migration of the linked feature in a feature node under the **Version>Building Blocks** branch. You handle property links in a similar manner.


Feature (Migration)

A **Feature** node contains all migration operations for the feature's settings and for the subfeatures that you use for the feature. If a feature uses feature links to a feature under the **Building Blocks** branch () , there must be a feature link node under the feature. You handle the migration of the linked feature in a feature node under the **Version>Building Blocks** branch.

Property (Migration)

A **Property** node () contains all migration operations for the property's settings.

Change Type

Use the **Change Type** node () to change the type of a physics interface or a feature. Changing the type makes all files saved in an old version unusable unless you handle the migration properly. The **Settings** window has the following sections:

CHANGE TYPE

In the **Old type** label you see the old type, and you can adjust the new type in the **New type** field. The automatic logging of changes should prepare new **Change Type** nodes with the a correct new type.

COMPATIBILITY

This section contains two check boxes. One for activating migration or backward compatibility when opening COMSOL Multiphysics files, and the other for activating compatibility for the model object API. The default is to use both types of compatibility, but you can clear any of the check boxes to deactivate the particular compatibility.

Rename Inputs

The following nodes are used to rename the associated node:

- **User Input**
- **Material Parameter**
- **Feature Input**
- **Material List**

You can rename a user input, which makes a file saved in an old version unaware of that the old user input value is the value of the new input. Nothing actually breaks, but the input gets its default value, so you should handle migration. The situation can be worse for API migration because a model file for Java cannot execute if you try to access the old input.

The **Settings** window has the following sections:

CHANGE NAME

In the **Old name** label you see the old name, and you can adjust the new name in the **New name** field. The automatic logging of changes should prepare new **Rename User Input** nodes with the a correct new type.

COMPATIBILITY

Identical to the [Compatibility](#) section of the [Change Type](#) node.

Migration Links

The following nodes are described:

- Feature link
- Property link
- Contained interface
- Contained feature

The migration link nodes contain a link to the actual node under the **Building Blocks** branch that handles the actual migration. You specify the link in the **Link** list.

Documentation

In this section:

- [Introduction to Comments and Documentation](#)
- [Physics Interface Documentation](#)
- [User Documentation](#)
- [Comments](#)
- [The Documentation Node](#)
- [Documentation Text Components](#)
- [The Preview Window](#)

Introduction to Comments and Documentation



Depending on the use of the created physics interfaces, the need for internal documentation (comments about implementation and for simplifying extending and maintaining the implementation) and external documentation (user documentation and context help) varies. The Physics Builder includes tools for creating documentation for both internal and external documentation. There are two main types of nodes that you can add to the physics interface and its features:

- **Comment** nodes, where you can add comments about each feature for internal use. Those comments can provide information about the implementation, its benefits and limitations, any remaining issues, or ideas for future extensions. You can add a **Comment** node to each individual node in a physics interface, including the components that you use to create a physics interface feature (such as **User Input** and **Variable Declaration** nodes).
- **Documentation** nodes, for creating end-user documentation, including context help for the physics interface and its features. For in-depth documentation, you can use various sections that add plain text, equations, images, lists, tables, references, and other documentation items to build a full documentation of the functionality, use, and theory behind the physics interface and its features.


The following sections provide details about how to add documentation to physics interfaces in the Physics Builder.

Physics Interface Documentation

By default, the preferences are set up so that **Developer Comment** and **User Documentation** nodes appear under the nodes in the interface that needs comments and documentation. You can change this in the **Preferences** dialog box under **Comments and documentation** on the **Physics Builder** page. For the user documentation, a default **User Documentation** text node is added automatically under the main **User Documentation** node, but you can add any other documentation components such as equations, images, notes, and tables. The main **User Documentation** node adds a heading and the topic description used for the context help when clicking the node in the COMSOL Desktop.

When you have created the documentation contents, right-click the main Physics Interface node and select **Compile Documentation** () to compile the documentation from the various **User Documentation** nodes into a complete document that appears under the **Documentation** node () at the bottom of the tree in the **Physics Builder** window. Under **Documentation** you can add additional documentation components to describe parts of the physics that are not directly connected to the nodes for the interface such as an introduction or a theory section.


User Documentation

The **Documentation** node () contains the end-user documentation for a physics interface node, feature node, or other builder component that needs documentation. When added by default, it uses the label **User Documentation**. By default, a **User documentation** Text node appears under the main **User Documentation** node for **Feature** nodes' documentation, for example. Right-click the main **User Documentation** node to add other documentation components as needed.

In the **Section Heading** section, you add a **Heading**, which is typically the name of the node that the documentation is for. The default headings, `<ref entity="doc.physics">` for a physics interface and `<ref entity="doc.feature">` for a feature, are references to the contents of the **Description** fields in those nodes' **Settings** windows.

Clear the **Use topic resource if available** check box if you do not have a topic resource file, which would be the case for most development of physics interfaces. Then enter a description of the node under **Topic description**.




By default, no **Documentation** nodes are added or displayed. From the **Show** () menu in the **Physics Builder**, you can choose one of the following options from the **Documentation** menu:

- **Show All Documentation** to show all **Documentation** nodes.
- **Show No Documentation** (the default) to show no **Documentation** nodes.

This setting is also available on the **Physics Builder** page in the **Preferences** dialog box.


Comments

The **Comments** node () contains developer comments about the implemented feature or other builder component. When added by default as a subnode to a node that you add in the **Physics Builder**, it's called Developer Comments. These comments can include known capabilities and limitations, ideas for further development, and implementation detail that can be useful for maintaining and extending the functionality.

In the **Text** section you add this information. You can use the character formatting tools above and below the text field to format parts of the text. The reference that is

included by default, `<ref entity="doc.entity">`, is a general reference to the entity that the comment is about.







By default, **Comments** nodes are added as subnodes to physics builder nodes that you add, and the **Comments** node are displayed. From the **Show** () menu in the **Physics Builder**, you can choose on of the following options from the **Comments** menu:

- **Show All Comments** (the default) to show all **Comments** nodes.
- **Show Changed Comments** to show only **Comments** nodes where the contents has changed.
- **Show No Comments** to not show any **Comments** nodes.

This setting is also available on the **Physics Builder** page in the **Preferences** dialog box.

The Documentation Node

The main **Documentation** node () contains information about the formatting and defaults for a documentation. Click the **Preview Selected** () or **Preview All** () button to show a preview of the document in the **Preview** window. Click the **Write** button () in the toolbar for the **Documentation Settings** window to create a document. The **Write** option is also available by right-clicking any node in the documentation. Selecting **Write** from any documentation node's context menu generates the entire document.

FORMAT

You can select to create a document in one of the following formats, which you choose from the **Output format** list:

- **HTML** (the default format), for creating the document as an HTML file for display in a web browser.
- **Microsoft Word**, for creating the document as a Microsoft® Word .docx file.
- **Integrated help**, for creating a help project folder that can become an integral part of the COMSOL documentation and help system.

Settings for Documentation in HTML Format

When generating documentation, you need to specify its name and title where to store the file. Enter the output directory for the HTML files in the **HTML output directory**

field, or click **Browse** to open the **Specify HTML Output Directory** dialog box and browse to the desired location. Also specify the document's name a title in the **Document name** and **Document title** fields, respectively.

Settings for Documentation in Microsoft® Word Format

These settings are the same as for the HTML format, except you specify the output directory in the **Word output directory** field or using the **Specify Word Output Directory** dialog box.

Settings for Documentation as Integrated Help

When creating an integrated help project, you need to specify the following details and properties:

- Enter the output directory for the help project in the **Help output directory** field, or click **Browse** to open the **Specify Help Output Directory** dialog box and browse to the desired location.
- Enter a **Namespace** to use a common namespace for your help projects.
- Enter project name and title in the **Name** and **Title** fields. The complete project name is formed from the namespace and the name. The title is the text that displays in the **Contents** tree on the left side of the standalone **Help** window or on the **Contents** page of the **Help** window for context-sensitive help integrated in the COMSOL Desktop.
- From the **Add to list**, select **None** (the default) to not link the project to the contents of any COMSOL product, or select COMSOL Multiphysics or any of its modules to link the documentation to that of one of the products. As an advanced option, you can select **Custom** to link to a custom help project that you specify in the **Link to help project folder** field and the **Link to anchor ID** field. The custom help project must contain the specified anchor for your help project to link to.

Documentation Text Components

Right-click **Section** nodes to select and add these documentation nodes: **Bibliography**, **Code**, **Equation**, **Heading**, **Image**, **List**, **Note**, **Bibliography**, **Table**, and **Text**. In addition there are subnodes for creating a **Reference**, **List Item**, **Table Heading Row**, or **Table Row**.



Reports and Custom Report Components in the *COMSOL Multiphysics Reference Manual*

The following table lists all documentation components:

TABLE 3-2: DOCUMENTATION COMPONENTS
















DOCUMENTATION COMPONENT	ICON	DESCRIPTION
Bibliography		Adds a reference or bibliography to the report or document. Right-click to add Reference nodes for each reference.
Code		Adds a text block for code using a code (monospace) font. You can also make part of the text using an italic or bold variant of the code font.
Equation		Adds an equation to the report or document. You can use LaTeX markup directly or import the equation as an image. Under Equation preview you can see the equation that the LaTeX commands that you enter create.
Heading		Adds a heading to the report or document with a text from the Text field and a layout for the level (Level 1–Level 6) from the Level list. The default is to use the level where the Heading node appears.
Image		Adds an image to the report or document. Select the image source from the Source list: Plot group to select the plot from available plots in the Plot group list or External to use any external image file in PNG, Windows Bitmap (BMP), or JPEG format. Add a Caption if desired.
List		Adds a list. By default, the Numbered check box is selected, giving a numbered list; clear the check box for an unordered (bullet) list. Right-click the List node to add List Item nodes.
List Item		Right-click the List node to add this node with a Text area for the list item's contents. Right-click to add Code , Equation , Image , Table , Text , or other List nodes for inserted texts, equations, images, or tables in the list or for creating nested lists.
Note		Adds a Note node for adding one of the following note types, which you select from the Type list: Note (the default), Caution , Important , Model , See also , or Tip . From the Show list, select Icon (the default) to display the icon only, Description (the type), or Icon and description . Then add the text for the note.


TABLE 3-2: DOCUMENTATION COMPONENTS

DOCUMENTATION COMPONENT	ICON	DESCRIPTION
Reference		Adds a reference to a bibliography or reference sections. Select one of the following reference types from the Type list: Journal article (the default), Book , Conference paper , Thesis , or Web . You can always add Authors and Title ; the other parts of the reference depends on the type. For Web , you add the URL (web address) to the web page.
Table		Adds a table with a Title and a Number of columns (default: 3 columns). Right-click to add a Table Heading Row and Table Rows .
Table Heading Row		Right-click the Table node to add this node and then define headings for each column.
Table Row		Right-click the Table node to add this node and then add the contents for each column in a row of a table.
Text		Provides a Text area where text can be added (including HTML tags for formatting and links).




For all **Text**, **List Item**, and **Note** nodes' settings, a set of tools above and beyond the text field provides a quick way to add formatting to the text:

- The formatting tools above the text provide character formats for user-interface labels, emphasis, code (standard, bold, and italic), equation components (bold, variables, and constants), subscript, and superscript. To convert a part of the text to any of these character formats, highlight the text that you want to format and then click  , for example, to mark the text as a user-interface label (a sans-serif boldface font) using the HTML tags `<1>` and `</1>` before and after the text.
- From the character tools below the text, click the character that you want to insert, for example, click  to insert an uppercase omega as `\Omega` in the text. The character tools include lowercase and uppercase Greek letters and the en-dash (–) and em-dash (—) punctuation symbols.

For creating equations and text that includes mathematical symbols as part of the documentation, COMSOL supports a subset of the LaTeX language. Commands include Greek and other characters, mathematical symbols and operators, arrows, text and font formats, and environments for text and mathematical typesetting. See [Mathematical Symbols and Special Characters](#) in the *COMSOL Multiphysics Reference Manual* for all available LaTeX commands.

Click **Preview Selected** () to display a preview of the text, including formatting, in the **Preview** window.

The Preview Window

The **Preview** window  opens when you click the **Preview Selected** () or **Preview All** () button. It shows the current documentation in HTML format so that you can test the documentation for a physics interface with the links to move up and down in the document. You can use the arrow buttons at the top of the window to move back, forward, or up to the top of the document.

Elements

You can create low-level *elements* that are not supported by any other standard node. A variable definition is actually an element, but it is much easier to use the [Variable Definition](#) node than creating the low-level element from scratch. Using these elements requires in-depth knowledge about the element syntax and is considered advanced usage. There is also limited documentation on the low-level element syntax.

In this section:

- [Element](#)
- [GeomDim](#)
- [Src](#)
- [Array](#)
- [Record](#)
- [String](#)
- [Elinv](#)
- [Elpric](#)
- [Event](#)
- [DG Wave Element, General Form](#)
- [Degree of Freedom Re-Initialization](#)
- [Shape Interpolation Element](#)


Element

An **Element** node creates a new element of the type entered in the **Element type** field. This node is a special type of [Record](#) node that represents the top level of an element.

In addition to the type entered in the **Element type** field, selecting the **Auxiliary element** check box enables the use of a special family of elements called *auxiliary elements*. These are used to define solver events, which the **Event** node also creates.


To add this node, go to **Building Blocks**. Right-click **Components** to add a **Component** node. Then select **Event** from the **Elements** menu.

GeomDim

The **GeomDim** node () is a selection specification of the `geomdim` type. See [Specifying Selections](#) for more information.


To add this node, go to **Building Blocks>**. Right-click **Components** to add both a **Component** and **Element** node. Then select **GeomDim**.

Src

An **SRC** node () is a selection specification of the src type. See [Specifying Selections](#) for more information.


To add this node, go to **Building Blocks>**. Right-click **Components** to add both a **Component** and **Element** node. Then select **SRC**.

Array

The **Array** node () is a container for other nodes of the types [Array](#), [String](#), and [Record](#). If this node is a child to a [Record](#) node, you specify the name of this record in the **Name** field.


To add this node, go to **Building Blocks>**. Right-click **Components** to add both a **Component** and **Element** node. Then select **Array**. Or add it as a child node to an **Array**, **Record** or **String** node.

Record

The **Record** node () is a container for other named nodes of the types [Array](#), [String](#), and [Record](#). All children to this node have to specify a unique record name to identify the record. If this node is a child to a [Record](#) node, you specify the name of this record in the **Name** field.

To add this node, go to **Building Blocks** then right-click **Components** to add both a **Component** and **Element** node. Then select **Record**. Or add it as a child node to an **Array**, **Record** or **String** node.

String

A **String** node () is used for string data for the element. The **Settings** window contains one or two sections, depending on if it is a child to a record node or not.

To add this node, go to **Building Blocks>**. Right-click **Components** to add both a **Component** and **Element** node. Then select **String**. Or add it as a child node to an **Array**, **Record** or **String** node.

STRING VALUE


In the **Value type** list, you choose the type of string data that you enter in the **Value** field. The option **Custom string** means that the data can be an arbitrary string. Use

Variable name to interpret the value as a variable name according to the rules outlined in [Entering Names](#). Choose **Expression** to interpret the value as an expression.

RECORD NAME


If this node is a child to a [Record](#) node, you specify the name of this record in the **Name** field.

Elinv

The **Elinv** node () is a special element for inverting square matrices using numerical algorithms, which is more efficient than an analytical inversion for large matrices (size > 3). The declaration is similar to the [Variable Declaration](#) node, and this node also generates a new matrix variable for the inverse. You enter the expression to be inverted in the **Expression** field under the **Input Matrix Definition** section.

To add this node, go to **Building Blocks>**. Right-click **Components** to add a **Component** node. Then select **Elinv** from the **Elements** menu.

Elpric

The **Elpric** node () is a special element for computing the eigenvectors and eigenvalues of square 3-by-3 symmetric matrices. The declaration is similar to the [Variable Declaration](#) node, and this node also generates three new vector variables plus three scalar eigenvalues. You enter the expression of the input matrix in the **Expression** field under the **Input Matrix Definition** section. The Elpric element also works with unsymmetric matrices, in which case it takes the upper-triangular part of the matrix and mirrors it to the lower-triangular part to get a symmetric matrix.


The **Elpric** node declares six variables: three scalar eigenvalues and three eigenvectors. The eigenvalues get the names using the template `<name><i>[_<suffix>]` where *i* is the eigenvalue number and can be 1, 2, or 3. Similarly, the eigenvector has the same name but with a `vec.` prefix in front of the name. As an example, let the name be `eig` and the suffix be empty. This gives the following variables: `eig1`, `eig2`, `eig3`, `vec.eig1`, `vec.eig2`, and `vec.eig3`. The scalar components of the first eigenvector become: `eig2x`, `eig2y`, and `eig2z`.



The `vec.` prefix is not used for the components. It is only used to access the entire vector.

To add this node, go to **Building Blocks**>. Right-click **Components** to add a **Component** node. Then select **Elpric** from the **Elements** menu.

Event


The **Event** node () defines a solver event that can trigger the solver to stop, reinitialize some dependent variables, and then continue. For more information about solver events see [The Events Interface](#) in the *COMSOL Multiphysics Reference Manual*. The reinitialization step can either be defined by the adding a **Degree of Freedom Initialization** node to the **Event** node, or adding a **Degree of Freedom Re-Initialization** node somewhere else.

To add this node, go to **Building Blocks**. Right-click **Components** to add a **Component** node. Then select **Event** from the **Elements** menu.

DECLARATION

Enter a unique tag in the **Event tag** field to identify the event to **Degree of Freedom Re-Initialization** nodes. In the **Event type** list, choose the **Explicit** or **Implicit** event type. For explicit events, specify a **Start time** and optionally a **Period** for cyclic events. The implicit event requires a **Condition** that triggers the event when the value of the condition goes from 0 to 1. The condition must contain a degree of freedom with the **Solver field type** set to **Quadrature** in the **Advanced** section of either a **Variable Definition** node or a **Dependent Variable Definition** node. Note that this option is currently only supported for global states. Set the **Solver field type** to **Discrete** for global states that are discontinuous in time. Such states are typically used as logical help variables in if-statements to turn on and off equations.

DG Wave Element, General Form

Using this element to create an element optimized for solving wave equation using the *discontinuous Galerkin method*. The section settings for the **DG Wave Element, General Form** node () are similar to the ones for the [General Form Equation](#) node, with the following exceptions:

COEFFICIENTS

Use the **Type** list to select the type of the equation: **Normal** (the default) and **Interior boundary source**. The latter option can only be used in boundary conditions. Also note that boundary conditions do not support the Γ and d_a coefficients. Those settings are

disabled for the **Interior boundary source** type; otherwise they are simply ignored when the equation is used on a boundary.

For this element, the conservative flux Γ , the source term f , and the damping or mass coefficient d_a are applicable.

DEPENDENT VARIABLE

The **Variable names** field supports a comma-separated list of dependent variable names. It is important that you enter multiple dependent variables in the same order for all element nodes. Multiple dependent variables are necessary to allow cross-coupling in the d_a and Γ coefficients. All dependent variables must use the nodal discontinuous Lagrange shape function. For multiple dependent variables, the value of N in the **Coefficients** section is the sum of all lengths of the dependent variables.

ADVANCED

Always enter numerical values in the **Lax-Friedrichs flux parameter** field with the length N , the sum of all lengths of the dependent variables.

To include filter coefficients, select the **Activate filter** check box. The filter coefficients — α , η , and s — can be NaN (not-a-number, which is the default value) to disable that filter coefficient. Each filter coefficient expects a scalar value.



Axial symmetry is not automatically handled by the equation form. The coefficients have to be reformulated to include gradients and volume factors for axial symmetry.

Degree of Freedom Re-Initialization

The **Degree of Freedom Re-Initialization** node ([u,v,w](#)) is identical to the **Degree of Freedom Initialization** node that initializes degrees of freedom (dependent variables, shape variables, and global states). There is only one extra section for needed for reinitialization after triggered solver events:

REFERENCE

Enter the unique tag in the **Event tag** field that points to the **Event** node that triggers this reinitialization step.



See [Degree of Freedom Initialization](#) for the rest of the settings.

To add this node, go to **Building Blocks**. Right-click **Components** to add a **Component** node. Then select **Degree of Freedom Re-Initialization** from the **Elements** menu.

Shape Interpolation Element

The **Shape Interpolation Element** node (**a=**) (which you add from a feature or **Component** node's **Elements** submenu) adds a special contribution to declared variables that adds an element. The element behaves as a shape function but instead of solving for the shape's degrees of freedoms, an expression defines their values directly. The variable the shape defines uses the shape function interpolation to compute the variable's values everywhere. The **Settings** window contains the sections:

DEFINITION

Enter the name of the variable you add the definition for in the **Variable name** field. The variable name follows the rules described in [Entering Names and Expressions](#) and must match the name of a variable declaration somewhere in the same physics interface. The **Shape function** list contains the shape functions that this definition supports. Enter the degree of freedom expression in the **Expression** field. This is the expression the shape function use to find the values of the variable it defines.

SELECTION

The options in the **Selection** list and **Output entities** list defines the selection where this variable definition is valid. See [Specifying Selections](#) for more information.

Examples of Custom Physics

The examples in this chapter show how to create custom physics interfaces for different applications:

- Joule heating is a well known fundamental multiphysics phenomenon, but it is not the only type of electro-thermal interaction. In addition, there is the thermoelectric effect, which historically is known under three different names: the Seebeck, Peltier, and Thomson effects. The first example in this chapter shows how to use the Physics Builder to create a custom physics interface for solving generic combined Joule heating and thermoelectric effects. See [The Thermoelectric Effect](#) and the following sections.
- The Schrödinger equation describes the behavior of a quantum state in quantum mechanics. The second example in this chapter shows how to use the Physics Builder to create a custom physics interface for solving a version of the Schrödinger equation. See [The Schrödinger Equation](#) and the following sections.

Both the Physics Builder files (MPHPHB-files) and model examples (MPH-files) for these two custom physics are included in the COMSOL installation. You find the files in the `demo/builder` directory in your COMSOL installation directory.

The Thermoelectric Effect

In this section:

- [Introduction to the Thermoelectric Effect](#)
- [Equations in the Physics Builder](#)



A predefined Thermoelectric Effect multiphysics interface is available in the Heat Transfer Module.

Introduction to the Thermoelectric Effect

The *thermoelectric effect* is the direct conversion of temperature differences to electric voltage or the other way around. It is the mechanism behind devices such as thermoelectric coolers for electronic cooling or portable refrigerators. While *Joule heating* (resistive heating) is an irreversible phenomena, the thermoelectric effect is in principle reversible. Historically, the thermoelectric effect is known under three different names, reflecting its discovery in experiments by Seebeck, Peltier, and Thomson. The *Seebeck effect* is the conversion of temperature differences into electricity, the *Peltier effect* is the conversion of electricity to temperature differences, while the *Thomson effect* is heat produced by the product of current density and temperature gradients. These three effects are thermodynamically related by the Thomson relations:

$$P = ST$$
$$\mu = T \frac{dS}{dT}$$

where P is the Peltier coefficient (SI unit: V), S is the Seebeck coefficient (SI unit: V/K), T is the temperature (SI unit: K), and μ is the Thomson coefficient (SI unit: V/K). These relations show that all three effects can be considered as one and the same effect. This example primarily uses the Seebeck coefficient and also, merely as an intermediate variable, the Peltier coefficient. The Thomson coefficient is not used.

The flux quantities of interest when simulating the thermoelectric effect are the heat flux \mathbf{q} and the flux of electric current \mathbf{J} :

$$\begin{aligned}\mathbf{q} &= -k\nabla T + P\mathbf{J} \\ \mathbf{J} &= -\sigma\nabla V - \sigma S\nabla T\end{aligned}$$

Some other quantities of relevance are:

$$\begin{aligned}\mathbf{E} &= -\nabla V \\ Q &= \mathbf{J} \cdot \mathbf{E}\end{aligned}$$

where \mathbf{E} is the electric field and Q is the Joule heating.

Conservation of heat energy and current gives:

$$\begin{aligned}\rho C \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{q} &= Q \\ \nabla \cdot \mathbf{J} &= -\frac{\partial \rho_c}{\partial t}\end{aligned}$$

where ρ is the density, C is the heat capacity, and ρ_c is the space charge density. In this example, consider the stationary case only:

$$\begin{aligned}\nabla \cdot \mathbf{q} &= Q \\ \nabla \cdot \mathbf{J} &= 0\end{aligned}$$

More explicitly, the thermoelectric equations become:

$$\begin{aligned}\nabla \cdot (-k\nabla T + P(-\sigma\nabla V - \sigma S\nabla T)) &= (-\sigma\nabla V - \sigma S\nabla T) \cdot (-\nabla V) \\ \nabla \cdot (-\sigma\nabla V - \sigma S\nabla T) &= 0\end{aligned}$$

It is pretty clear that the explicit form of the equations are cumbersome to work with, and this example makes use of a series of intermediate variables to simplify entering them in the Physics Builder.

Equations in the Physics Builder

The Physics Builder requires partial differential equations to be entered in the weak form. To transfer to weak form, multiply each of the two equations with the test functions corresponding to the unknowns T and V (here called v_T and v_V , respectively) and integrate over the whole computational domain D :

$$\int_D (\nabla \cdot \mathbf{q}) v_T = \int_D Q v_T$$

$$\int_D (\nabla \cdot \mathbf{J}) v_V = 0$$

Partial integration gives:

$$-\int_D \mathbf{q} \cdot \nabla v_T + \int_B \mathbf{n} \cdot \mathbf{q} v_T = \int_D Q v_T$$

$$-\int_D \mathbf{J} \cdot \nabla v_V + \int_B \mathbf{n} \cdot \mathbf{J} v_V = 0$$

where B is the boundary of D , and \mathbf{n} is the unit normal of D .

Assuming that there are known values for the heat and current flux in the direction of the boundary normal as q_0 and J_0 , respectively, the equations become:

$$-\int_D \mathbf{q} \cdot \nabla v_T + \int_B q_0 v_T = \int_D Q v_T$$

$$-\int_D \mathbf{J} \cdot \nabla v_V + \int_B J_0 v_V = 0$$

The Physics Builder requires you to enter the domain parts of the weak form equations while the boundary parts are more or less automatically available.

The integrands of the domain parts are:

$$0 = \mathbf{q} \cdot \nabla v_T + Q v_T$$

$$0 = \mathbf{J} \cdot \nabla v_V$$



The COMSOL convention collects all terms on the right-hand side.

Using Physics Builder syntax, the right-side expressions become:

$$\mathbf{q} \cdot \text{test}(\nabla T) + Q \text{test}(T)$$

$$\mathbf{J} \cdot \text{test}(\nabla V)$$

and this is what you enter into the weak form text fields for the integrands' expressions when creating the Thermoelectric Effect physics interface.

This example also uses the fact that you get the heat equation as a subset of the thermoelectric equation system. To handle cases where one or more domains are electrically insulating but thermally conductive, first create a heat transfer equation interface and then define the full thermoelectric equations as a second step. The weak form integrand for “pure” heat transfer is:

$$\mathbf{q} \cdot \text{test}(\nabla T)$$

In this way you only need to solve for one degree of freedom, T , in the electrically insulating domains.

In addition to the domain weak form equations, a number of different boundary conditions are implemented:

$$T = T_0$$

$$V = V_0$$

$$q_0 = 0$$

$$J_0 = 0$$

$$q_0 = q_{\text{in}}$$

- The first condition sets a temperature T_0 on a boundary.
- The second condition sets a voltage V_0 on a boundary.
- The two conditions that set the heat flux and current density on the boundary to zero are so-called *natural boundary conditions*. They are called natural because they arrive “naturally” as part of the weak form partial integration. The natural boundary conditions represent thermal and electrical insulation. The implementation in this example bundles these two conditions into one single insulation boundary condition. It is not even necessary to define this bundled boundary condition because that would anyway have been available: any boundaries not explicitly set to a certain condition automatically obey the natural conditions. However, for better usability of the thermoelectric physics interface, the natural boundary condition is available as one of the choices. This way you can clearly see which boundaries are insulated.
- The last boundary condition sets the value of the heat flux in the direction of the boundary normal. The heat flux boundary condition is implemented with a weak equation:

$$\int_B q_0 v_T$$

which is one of the terms in the complete weak form equation for the heat transfer part of the thermoelectric equations, as seen earlier.

The following table summarizes all quantities relevant for the thermoelectric physics interface:

NAME	DESCRIPTION	SI UNIT	SIZE	GEOMETRY LEVEL	USER INPUT
k	Thermal conductivity	W/(m·K)	Scalar	Domain	Yes
sigma	Electric conductivity	S/m	Scalar	Domain	Yes
S	Seebeck coefficient	V/K	Scalar	Domain	Yes
T0	Temperature	K	Scalar	Boundary	Yes
V0	Electric potential	V	Scalar	Boundary	Yes
qin	Heat flux	W/m ²	Scalar	Boundary	Yes
T	Temperature	K	Scalar	Domain	No
V	Electric potential	V	Scalar	Domain	No
q	Heat flux	W/m ²	3-by-1 vector	Domain	No
J	Current density	A/m ²	3-by-1 vector	Domain	No
P	Peltier coefficient	V	Scalar	Domain	No
E	Electric field	V/m	3-by-1 vector	Domain	No
Q	Joule heating	W/m ³	Scalar	Domain	No

Thermoelectric Effect Implementation

In this section:

- [Overview](#)
- [Thermoelectric Effect Interface — Creating It Step by Step](#)

Overview

To implement a physics interface for the thermoelectric effect, you need to specify the following items:

- The name and description for the physics interface.
- The supported space dimension for the physics interface.
- The study types (Stationary, Time Dependent, Eigenvalue, and so on) that the physics interface supports.
- The equations, written using a weak formulation, to solve, and the input variables that they need.
- The boundary conditions that the physics interface needs, including the default boundary condition, and the inputs that they need.
- Any additional variables that are relevant to define for use in, for example, results analysis and visualization.
- A suitable default plot to be displayed when the solver has finished and possibly custom quantities as default plot expressions for new plots.

Optionally, you can also add customized default settings for the mesh generation and the solvers.

NAME AND DESCRIPTION

The name of this interface is *Thermoelectric Effect*. The short name is `tee`. There is also a type, `ThermoelectricEffect`, which is primarily used by the Java® and LiveLink™ for MATLAB® interfaces.

SUPPORTED SPACE DIMENSIONS

The Thermoelectric Effect interface is available in all space dimensions.

THE STUDY TYPES

The Thermoelectric Effect can be made available as a Stationary and Time Dependent study types (and perhaps even other study types for more unusual applications). In this example, stationary is the only study type.

THE EQUATIONS

Heat Transfer Model

The first equation is called a Heat Transfer Model and is represented by the following weak form equation:

$$\mathbf{q} \cdot \text{test}(\nabla T)$$

The weak formulation using the COMSOL tensor syntax becomes

$$\mathbf{q} \cdot \text{test}(\nabla T)$$

In this expression, ∇ is the *del* vector differential operator, and \cdot (dot) represents the *dot product* (*scalar product*).

Thermoelectric Model

The second equation is called the Thermoelectric Model and is represented by the following weak form equation:

$$\begin{aligned} \mathbf{q} \cdot \text{test}(\nabla T) + Q \text{test}(T) \\ \mathbf{J} \cdot \text{test}(\nabla V) \end{aligned}$$

The weak formulation using the COMSOL tensor syntax becomes

$$\begin{aligned} \mathbf{q} \cdot \text{test}(\nabla T) + Q * \text{test}(T) \\ \mathbf{J} \cdot \text{test}(\nabla V) \end{aligned}$$

where $*$ is ordinary multiplication between scalars.

A number of parameters are defined in order to efficiently use the COMSOL tensor syntax for defining the weak form equations and also variables available for results and visualization:

- The thermal conductivity k is a user input to both the Heat Transfer Model and the Thermoelectric Model. The default value is set equal to $1.6 [W / (m \cdot K)]$, which corresponds to the thermoelectric material Bismuth telluride.
- The electric conductivity σ is a second user input for the Thermoelectric Model. The default value is set equal to $1.1e5 [S/m]$, which also corresponds to the thermoelectric material Bismuth telluride.
- The Seebeck coefficient S is a third and final user input for the Thermoelectric Model. The default value is set equal to $200e-6 [V/K]$, which once again corresponds to the thermoelectric material Bismuth telluride.
- To make the definition of the weak equation easier you define a variable for the heat flux q as a 3x1 vector with the following expression:

$$P \cdot J - k \cdot \nabla T$$

There is a dot product between the thermal conductivity and the temperature gradient. This makes it easy to generalize the physics interface to an anisotropic thermal conductivity at a later time, if needed.

- A variable for the current density J is defined as a 3x1 vector with the following expression:

$$-\sigma \cdot (\nabla V + S \cdot \nabla T)$$

- A variable for the Peltier coefficient P is defined as a scalar with the following expression:

$$S \cdot T$$

- A variable for the electric field E is defined as a 3x1 vector with the following expression:

$$-\nabla V$$

- A variable for the Joule heating Q is defined as a scalar with the following expression:

$$J \cdot E$$

THE BOUNDARY CONDITIONS

The Thermoelectric Effect interface includes the following boundary conditions:

- A constraint for the temperature:

$T_0 - T$

where T_0 is a user input. The expression given for a constraint is understood to be set to zero, so the above constraint equation expression means: $T = T_0$.

- A constraint for the voltage:

$V_0 - V$

where V_0 is a user input.

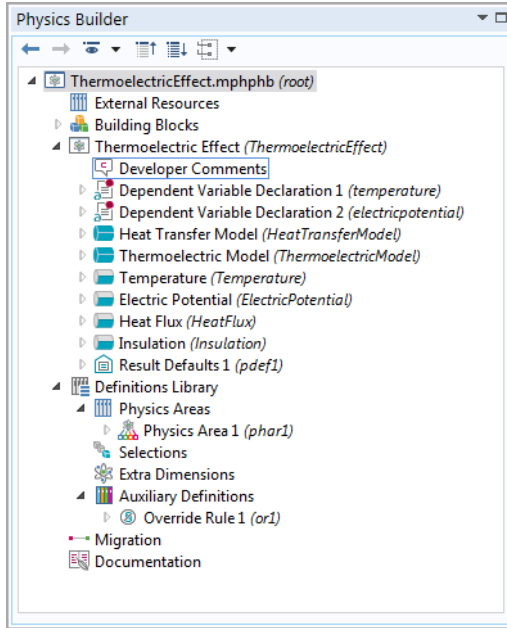
- A heat flux

q_{in}

where q_{in} is a user input.

- An insulation boundary condition with no user inputs. See the above theory section.




The constraint boundary conditions and the flux condition are *contributing*, while the insulation boundary condition is *exclusive*. A contributing boundary condition allows for more than one instance of the same boundary condition on a given boundary, where the model includes the combined effect of these boundary conditions. An exclusive boundary condition overrides any other previously defined boundary conditions on the given boundary. Ideally the constraint conditions should also be exclusive; however, this prevents you from having a boundary with simultaneous temperature and voltage constraints. If you accidentally set several contributing constraint boundary conditions on the same boundary, then the last boundary condition overrides all previously defined. The final Physics Builder tree displays as below:






Thermoelectric Effect Interface — Creating It Step by Step

The following steps show how to define the Thermoelectric Effect physics interface using the implementation defined in the previous section.

CREATING THE BASICS


- 1 Open COMSOL Multiphysics.
- 2 From the **File** menu (Windows) or the **Options** menu (the cross-platform version), choose **Preferences**. In the **Preferences** dialog box, select **Physics Builder** in the list and then select the **Enable Physics Builder** check box if not selected already.
- 3 From the **File** menu, choose **New** (). On the **New** page, click the **Physics Builder** button (). The **Physics Builder** window replaces the **Model Builder** window on the COMSOL Desktop.
- 4 On the **Home** toolbar, click **Add Physics Interface** () (or right-click the root node (**Untitled.mphph**) and select **Physics Interface**). This adds a **Physics Interface** node.


- 5 Go to the **Settings** window for **Physics Interface**. In the **Identifiers** section:
 - In the **Description** field enter Thermoelectric Effect. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - The **Type** field defaults to ThermoelectricEffect.
 - In the **Default name and tag** field enter tee.
 - If you have a custom **Icon** for the interface, click the **Browse** button to locate the icon file. The default is to use the physics.png icon ()
- 6 The Thermoelectric Effect interface should support all space dimensions except 0D, so under **Restrictions**, leave the **Allowed space dimensions** list with the default contents, which includes all space dimensions except 0D.
- 7 Under **Restrictions** in the **Allowed study types** list, select the default **Time Dependent** study type and click the **Delete** button () underneath the list. For this example only the **Stationary** study type is used.
- 8 In the **Settings** section, verify that **Domain** is selected in the **Top geometric entity level** list. This means that the equations in the physics interface apply to the domains in the geometry, which is the case for most physics interfaces. Leave the setting in the **Default frame** list at the default value (**Material**).
- 9 It is good practice to save the physics interface after completing some steps. From the **File** menu, choose **Save** and create a physics builder file, ThermoelectricEffect.mphpb in the default location. Click **Save**.

This concludes the initial steps that set up the fundamentals for the physics interface. The next steps adds equations, boundary conditions, and variables.

ADDING THE DEPENDENT VARIABLES



First declare the dependent variables T and V :

- 1 On the **Physics Interface** toolbar click **Dependent Variable Declaration** () . Or right-click **Thermoelectric Effect** (Physics Interface 1) node and from the **Variables** menu select **Dependent Variable Declaration**.




- 2 In the **Settings** window for **Dependent Variable Declaration** locate the **Declaration** section:
 - Keep the default setting in the **Dependent variable reference** list as **Use physical quantity**.
 - From the **Physical quantity** list select **Temperature (K)**, which makes suitable default values appear for the variable name and description.
 - In the **Symbol (LaTeX encoded)** field enter T.
 - Keep the default **Dimension** as **Scalar** because T is a scalar field.
- 3 Keep the default **Preferences** settings that make the dependent variable available for plotting (**Show in plot menu**) and for use as an input in other physics interfaces (**Announce variable to feature inputs**).
- 4 Keep the default **Discretization** settings (for default second-order Lagrange elements).
- 5 Add another **Dependent Variable Declaration** () node.
- 6 In the **Settings** window locate the **Declaration** section.
 - Keep the default setting in the **Dependent variable reference** list as **Use physical quantity**.
 - From the **Physical quantity** list select **Electric potential (V)**, which makes suitable default values appear for the variable name and description.
 - In the **Symbol (LaTeX encoded)** field enter V.
 - Keep the default **Size** setting as **Scalar** because V is a scalar field.
- 7 Keep the default **Discretization** and **Preferences** sections settings.


ADDING THE HEAT TRANSFER MODEL

Next add the Heat Transfer Model as a domain feature:

- 1 Right-click the **Thermoelectric Effect** node and from the **Features** menu select **Domain Feature** ()
- 2 In the **Settings** window locate the **Identifiers** section.
 - In the **Description** field enter Heat transfer model. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - The **Type** field updates automatically to HeatTransferModel.
 - In the **Default name and tag** field enter htm.
- 3 Under **Restrictions** keep the default setting **Same as parent** for both the **Allowed space dimensions** and **Allow study types** lists. If you select **Customized** you can restrict the

feature to a subset of the allowed space dimensions or study types for the physics interface.

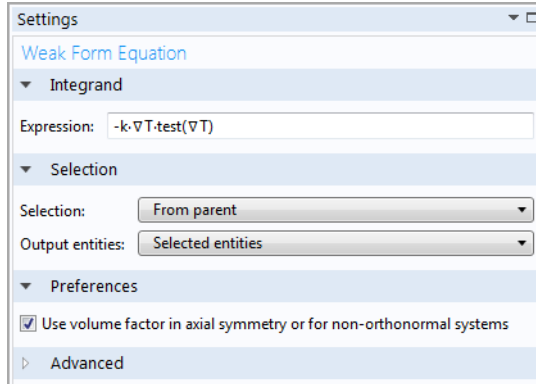
- 4 Keep the default values for the rest of the sections for the **Heat Transfer Model** node.
- 5 Right-click the **Heat Transfer Model** node and from the **Variables** menu select **Dependent Variable Definition** ().
- 6 In the **Settings** window locate the **Definition** section. From the **Physical quantity** list select **Temperature (K)**.
- 7 Keep the default settings for the rest of the **Dependent Variable Definition** node.
- 8 Right-click the **Heat Transfer Model** node and from the **Inputs** menu select **User Input** ().
- 9 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter `k`.
 - In the **Description** field enter Thermal conductivity.
 - In the **Symbol (LaTeX encoded)** field enter `k`.
 - From the **Physical quantity** list select **Thermal conductivity (W/(m*K))**
 - Keep the default settings for the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also keep the default **Allowed values** to **Any** for an entry of a general scalar number.
 - In the **Default value** field enter `1.6[W/(m*K)]` (typical value for bismuth telluride).
- 10 Keep all other default values for the **User Input** node.
- 11 Right-click the **User Input** node and select **Variable Definition** (). Or click the same button on the toolbar.

This user input then declares a variable with the name `k` and the expression `par.k`, which evaluates to the value entered for the user input. You can also refer to `k` directly in the weak form equation. A user of this physics interface refers to this variable as `tee.k` in the predefined expressions for results evaluation, for example.
- 12 Keep all other default values for the **Variable Definition** subnode.
- 13 Right-click the **Heat Transfer Model** node and from the **Equations** menu select **Weak Form Equation** (). Or click the same button on the toolbar.

This adds a **Weak Form Equation** node where you specify an equation for the domains in the physics interface.

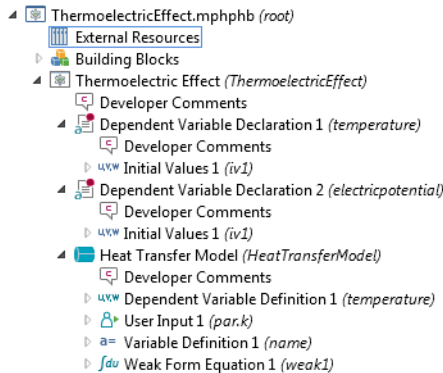
14 In the **Settings** window, locate the **Integrand** section. In the **Expression** field enter $-k \cdot \nabla T \cdot \text{test}(\nabla T)$. This expression implements the heat equation formulation for this interface.

Press Ctrl+Space to get lists of the supported operations, including any special characters. See [Tensor Parser](#) for the keyboard entries to create the del operator (∇) and the dot product (\cdot).




15 Keep all other default values for the **Weak Form Equation** node.




So far your Physics Builder tree should match the figure:




ADDING THE THERMOELECTRIC MODEL





Next add the ThermoElectric Model as a domain feature:

- I Right-click the **ThermoElectric Effect** node and from the **Features** menu select **Domain Feature** ().

- 2 In the **Settings** window for **Domain Feature** locate the **Identifiers** section.
 - In the **Description** field enter `Thermoelectric model`. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - The **Type** field updates automatically to `ThermoelectricModel`.
 - In the **Default name and tag** field enter `tem`.
- 3 In the **Preferences** section select the **Add as default feature** check box. Keep the setting in the **Default entity types** list to make this a default physics model in all domains.
- 4 Keep the rest of the settings for the **Thermoelectric Model**.
- 5 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Dependent Variable Definition** ()
- 6 In the **Settings** window locate the **Definition** section. From the **Physical quantity** list select **Temperature (K)**. Keep the other default settings.
- 7 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Dependent Variable Definition** ()
- 8 In the **Settings** window locate the **Definition** section. From the **Physical quantity** list select **Electric potential (V)**. Keep the other default settings.

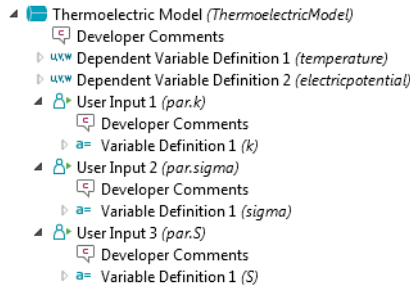
ADDING THREE USER INPUTS AND VARIABLE DEFINITIONS

- 1 Right-click the **Thermoelectric Model** node and from the **Inputs** menu select **User Input** ()
- 2 In the **Settings** window, locate the **Declaration** section.
 - In the **Input name** field enter `k`
 - In the **Description** field enter `Thermal conductivity`.
 - In the **Symbol (LaTeX encoded)** field enter `k`.
 - From the **Physical quantity** list select **Thermal conductivity (W/(m*K))**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - In the **Default value** field enter `1.6[W/(m*K)]` (typical value for bismuth telluride).
- 3 Keep all other default settings for the **User Input 1** node.

- 4 Right-click the **User Input 1** node and select **Variable Definition** ().
You can then refer to `k` directly in the weak form equation. A user of this interface refers to this variable as `tee.k` in the predefined expressions for results evaluation, for example.
- 5 Keep all other default settings for the **Variable Definition** subnode.
- 6 Right-click the **Thermoelectric Model** node and from the **Inputs** menu select **User Input** ().
- 7 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter `sigma`.
 - In the **Description** field enter `Electric conductivity`.
 - In the **Symbol (LaTeX encoded)** field enter `\sigma` to create a Greek σ symbol.
 - From the **Physical quantity** list select **Electrical conductivity (S/m)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - In the **Default value** field enter `1.1e5[S/m]` (typical value for bismuth telluride).
- 8 Keep all other default settings for the **User Input 2** node.
- 9 Right click **User Input 2** and select **Variable Definition** () from the context menu.
You can then refer to `sigma` directly in the weak form equation. A user of this interface refers to this variable as `tee.sigma` in the predefined expressions for results evaluation, for example.
- 10 Keep all other default settings for the **Variable Definition** subnode.
- 11 Add a **User Input** () node to the **Thermoelectric Model** node.
- 12 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter `S`.
 - In the **Description** field enter `Seebeck coefficient`.
 - In the **Symbol (LaTeX encoded)** field enter `S`.
 - From the **Physical quantity** list select **Seebeck coefficient (V/K)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - In the **Default value** field enter `200e-6[V/K]` (typical value for p-type bismuth telluride).







- 13 Keep all other default settings for the **User Input 3** node.
- 14 Right click **User Input 3** and select **Variable Definition** ($a=$) from the context menu.
You can then refer to S directly in the weak form equation. A user of this interface refers to this variable as $tee.S$ in the predefined expressions for results evaluation, for example.
- 15 Keep all other default settings for the **Variable Definition** subnode.

The Physics Builder tree for the Thermoelectric Model User Inputs and Variable Definitions should match the figure so far:




ADDING FIVE VARIABLE DECLARATIONS AND DEFINITIONS

- 1 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Variable Declaration** (\equiv).
- 2 In the **Settings** window, locate the **Declaration** section.
 - In the **Variable name** field enter q .
 - In the **Description** field enter Heat flux.
 - In the **Symbol (LaTeX encoded)** field enter q .
 - As **Dimension**, select **Vector (3x1)**.
 - From the **Physical quantity** list select **Inward heat flux (W/m²)**.
- 3 Right-click the **Variable Declaration 1** node and select **Variable Definition** ($a=$).
- 4 In the **Settings** window locate the **Definition** section. In the **Expression** field, enter $P * J - k \cdot \nabla T$.
Press Ctrl+Space to get lists of the supported operations, including any special characters. See [Tensor Parser](#) for the keyboard entries to create the del operator (∇) and the dot product (\cdot).
- 5 Keep all other default settings for the **Variable Definition 1** subnode.


- 6 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Variable Declaration** ().
- 7 In the **Settings** window, locate the **Declaration** section.
 - In the **Variable name** field enter J.
 - In the **Description** field enter Current density.
 - In the **Symbol (LaTeX encoded)** field enter J.
 - As **Dimension**, select **Vector (3x1)**.
 - From the **Physical quantity** list select **Current density (A/m²)**.
- 8 Right-click the **Variable Declaration 2** node and select **Variable Definition** ().
- 9 In the **Settings** window locate the **Definition** section. In the **Expression** field, enter $-\sigma \cdot (\nabla V + S \cdot \nabla T)$. Keep all other default settings for the **Variable Definition 1** subnode.
- 10 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Variable Declaration** ().
- 11 In the **Settings** window locate the **Declaration** section.
 - In the **Variable name** field enter P.
 - In the **Description** field enter Peltier coefficient.
 - In the **Symbol (LaTeX encoded)** field enter P.
 - Keep the default **Dimension** as **Scalar**.
 - From the **Physical quantity** list select **Electric potential (V)**.
- 12 Right-click the **Variable Declaration 3** node and select **Variable Definition** ().
- 13 In the **Settings** window locate the **Definition** section. In the **Expression** field enter $S \cdot T$. Keep all other default settings for the **Variable Definition 1** subnode.
- 14 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Variable Declaration** ().
- 15 In the **Settings** window locate the **Declaration** section.
 - In the **Variable name** field enter E.
 - In the **Description** field enter Electric field.
 - In the **Symbol (LaTeX encoded)** field enter E.
 - As **Dimension**, select **Vector (3x1)**.
 - From the **Physical quantity** list select **Electric field (V/m)**.
- 16 Right-click the **Variable Declaration 4** node and select **Variable Definition** ().

17 In the **Settings** window locate the **Definition** section. In the **Expression** field enter $-\nabla v$. Keep all other default settings for the **Variable Definition 1** subnode.

18 Right-click the **Thermoelectric Model** node and from the **Variables** menu select **Variable Declaration** ().

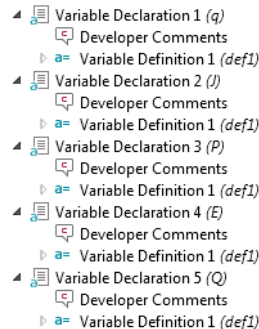
19 In the **Settings** window, locate the **Declaration** section.

- In the **Variable name** field enter Q .
- In the **Description** field enter Joule heating.
- In the **Symbol (LaTeX encoded)** field enter Q .
- Keep the default **Dimension** as **Scalar**.
- From the **Physical quantity** list select **Heat source (W/m³)**.


20 Right-click the **Variable Declaration 5** node and select **Variable Definition** ().

21 In the **Settings** window locate the **Definition** section. In the **Expression** field, enter $J \cdot E$. Keep all other default settings for the **Variable Definition 1** subnode.


The Variable Declaration and Variable Definitions subnodes should match the figure so far:



ADDING TWO WEAK FORM EQUATIONS

1 Right-click the **Thermoelectric Model** node and from the **Equations** menu select **Weak Form Equation** (). Or click the **Weak Form Equation** button on the toolbar.

2 In the **Settings** window locate the **Integrand** section. In the **Expression** field, enter $q \cdot \text{test}(\nabla T) + Q * \text{test}(T)$. Press Ctrl+Space to enter the del operator (∇) and the dot product (\cdot).





3 Right-click the **Thermoelectric Model** node and from the **Equations** menu select **Weak Form Equation** ().

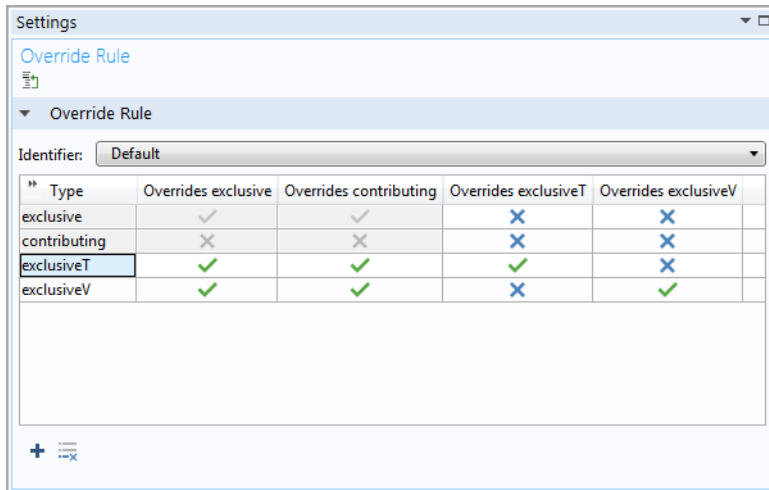
- In the **Settings** window, locate the **Integrand** section. In the **Expression** field, enter $-J \cdot \text{test}(E)$. Note that $-E$ is used instead of ∇V . Either syntax would work.

ADDING BOUNDARY CONDITIONS






Boundary conditions are defined in a way that is similar to domain features. Boundary conditions can have their own user inputs and equations. This physics interface requires a special rule for overriding the temperature boundary condition and the electric potential boundary condition.

Override Rule for Boundary Conditions

- Under **Definitions Library** () right-click **Auxiliary Definitions** () and select **Override Rule** ().
- Click the **Add** button () twice to add two rows and two columns.
- In the third row of the **Type** column, enter `exclusiveT` (replace `type_id_0`).
- In the fourth row of the **Type** column, enter `exclusiveV` (replace `type_id_1`).
- Click to set the **Overrides exclusive** and **Overrides contributing** columns for `exclusiveT` and `exclusiveV` to a green check mark.
- Click in column **Overrides exclusiveT** across from `exclusiveT` to set to a green check mark.
- Click in column **Overrides exclusiveV** across from `exclusiveV` to set to a green check mark. The **Settings** window should match the figure.




The Temperature Boundary Condition

- 1 Right-click the **Thermoelectric Effect** node and from the **Features** menu select **Boundary Condition** ().
 - 2 In the **Settings** window for **Boundary Condition** locate the **Identifiers** section.
 - In the **Description** field enter Temperature. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - The **Type** field defaults to Temperature.
 - In the **Default name and tag** field enter tp.
 - 3 In the **Restrictions** section, keep the default setting (**Same as parent**) for the **Allowed space dimensions** and **Allow study types** lists.
 - 4 In the **Selection Settings** section keep **Exterior** and **Interior**.
 - Click the **Add** button  underneath the list. Select **Pair** and click **OK**. It is added to the list under **Applicable entities**. This makes the boundary condition available for all those boundary types.
 - From the **Override rule** list choose **Locally defined**. Keep the default **Override Rule 1** in the **Link** list.
 - From the **Override type** list choose **exclusiveT**.
- 1 Right-click the **Temperature** node and from the **Inputs** menu select **User Input** ().
 - 2 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter T0.
 - In the **Description** field enter Temperature.
 - In the **Symbol (LaTeX encoded)** field enter T.
 - From the **Physical quantity** list select **Temperature (K)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - In the **Default value** field enter 293.15[K] corresponding to a room temperature of 20 degrees Celsius.
 - 3 Keep all other defaults for the **User Input 1** node.
 - 4 Right-click the **User Input 1** node and select **Variable Definition** ().


You can then refer to T0 directly in the constraint equation. A user of this interface refers to this variable as $t_{ee.T0}$ in the predefined expressions for results evaluation, for example.

5 Keep all other defaults for the **Variable Definition I** subnode.


Now define a section in the **Settings** window for the boundary condition:

6 Right-click the **Temperature** node and from the **Inputs** menu select **Section** ().

7 In the **Settings** window, locate the **Declaration** section.

- In the **Group name** field enter TemperatureSection.
- In the **Description** field enter Temperature.
- Click the **Add** () button and choose **User Input I (par.T0)**. Click **OK** and it is added the **Group members** list.

Now define the constraint equation that constrains the temperature to the specified value on the boundary:


8 Right-click the **Temperature** node and from the **Equations** menu select **Constraint** (). Or click the same button on the toolbar.

9 In the **Settings** window locate the **Declaration** section. In the **Expression** field enter $T_0 - T$ to make the temperature T equal to T_0 on the boundary (the constraint makes the expression equal to zero).


10 Locate the **Shape Declaration** section and from the **Physical quantity** list select **Temperature (K)**.

11 Keep all other default values for the **Constraint I** node.

The Electric Potential Boundary Condition


1 Right-click the **Thermoelectric Effect** node and from the **Features** menu select **Boundary Condition** ().

2 In the **Settings** window locate the **Identifiers** section.

- In the **Description** field enter ElectricPotential. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
- The **Type** field defaults to ElectricPotential.
- In the **Default name and tag** field enter ep.

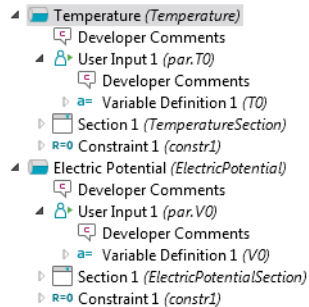
3 In the **Selection Settings** section:

- From the **Override rule** list choose **Locally defined**. Keep the default **Override Rule I** in the **Link** list.
- From the **Override type** list choose **exclusiveV**.




4 Right-click the **Electric Potential** node and from the **Inputs** menu select **User Input** ().


- 5 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter V_0 .
 - In the **Description** field enter Electric potential.
 - In the **Symbol (LaTeX encoded)** field enter V .
 - From the **Physical quantity** list select **Electric potential (V)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - Keep the **Default value** at 0.
- 6 Keep all other defaults for the **User Input I** node.
- 7 Right-click the **User Input I** node and select **Variable Definition** ($a=$).
 You can then refer to V_0 directly in the constraint equation. A user of this interface refers to this variable as tee.V0 in the predefined expressions for results evaluation, for example.
- 8 Keep all other defaults for the **Variable Definition I** subnode.
 Now define a section in the **Settings** window for the boundary condition:
- 9 Right-click the **Electric Potential** node and from the **Inputs** menu select **Section** (\square).
- 10 In the **Settings** window locate the **Declaration** section.
 - In the **Group name** field enter `ElectricPotentialSection`.
 - In the **Description** field enter Electric potential.
 - Click the **Add** ($+$) button and choose **User Input I (par.V0)**. Click **OK** and it is added the **Group members** list.
 Now define the constraint equation for constraining the potential at the boundary to a specified potential.
- 11 Right-click the **Electric Potential** node and from the **Equations** menu select **Constraint** ($R=0$).
- 12 In the **Settings** window locate the **Declaration** section. In the **Expression** field, enter $V_0 - V$, which constrains the value of the potential V to V_0 .
- 13 Locate the **Shape Declaration** section and from the **Physical quantity** list select **Electric potential (V)**.

The Temperature and Electric Potential boundary conditions in the Physics Builder tree should match the figure:



The Heat Flux Boundary Condition

- 1 Right-click the **Thermoelectric Effect** node and from the **Features** menu select **Boundary Condition** ().
- 2 In the **Settings** window locate the **Identifiers** section.
 - In the **Description** field enter Heat flux. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - The **Type** field updates automatically to HeatFlux.
 - In the **Default name and tag** field enter hf.
- 3 In the **Selection Settings** section from the **Override type** list choose **Contributing** (that is, more than one heat flux can contribute to the total heat flux across a boundary).
- 4 Right-click the **Heat Flux** node and from the **Inputs** menu select **User Input** ().
- 5 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter q_{in} .
 - In the **Description** field enter Normal heat flux.
 - In the **Symbol (LaTeX encoded)** field enter q_{in} (for displaying q_{in}).
 - From the **Physical quantity** list select **Inward heat flux (W/m²)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - Keep the **Default value** at 0.

- 6 Right-click the **User Input I** node and select **Variable Definition** ().


You can then refer to q_{in} directly in the weak form equation. A user of this interface refers to this variable as `tee.qin` in the predefined expressions for results evaluation, for example.

- 7 Leave all other default settings for the **Variable Definition I** subnode.


Now define a section in the **Settings** window for the boundary condition:

- 8 Right-click the **Heat Flux** node and from the **Inputs** menu select **Section** ().

- 9 In the **Settings** window locate the **Declaration** section.


- In the **Group name** field enter `HeatFluxSection`.
- In the **Description** field enter `Heat flux`.
- Click the **Add** () button and choose **User Input I (par.qin)**. Click **OK** and it is added the **Group members** list.

Now define the weak form equation for the heat flux:


- 10 Right-click the **Heat Flux** node and from the **Equations** menu select **Weak Form Equation** ().

- 11 In the **Settings** window locate the **Integrand** section. In the **Expression** field, enter `qin*test(T)`. For a theoretical explanation of this expression, see the earlier theory section.

The Insulation Condition

- 1 Right-click the **Thermoelectric Effect** node and from the **Features** menu select **Boundary Condition** ().

- 2 In the **Settings** window locate the **Identifiers** section.

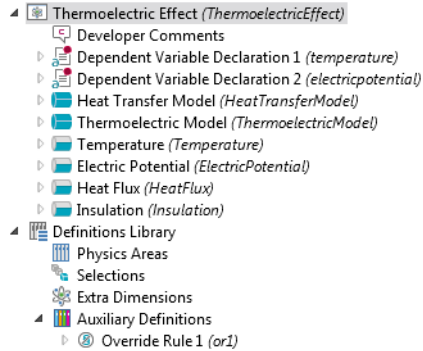
- In the **Description** field enter `Insulation`. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
- The **Type** field updates automatically to `Insulation`.
- In the **Default name and tag** field enter `in`.

- 3 In the **Selection Settings** section keep the default **Override type (Exclusive)**.

- 4 In the **Preferences** section, select the **Add as default feature** check box. Keep the default setting in the **Default entity types** list to make this a default boundary condition on **Exterior** boundaries only.












- 5 Keep all other default settings. If you do not add an explicit boundary condition, it is the same as adding a homogeneous Neumann condition (that is, thermal insulation in this case).

The Physics Builder tree should match this figure so far:










DEFINING DEFAULT PLOTS AND DEFAULT PLOT QUANTITIES

Define a default 3D plot group for plotting the temperature and make the temperature the default scalar quantity for user-defined plots:

- 1 Right-click **Thermoelectric Effect** and select **Result Defaults** ().
- 2 Right-click **Result Defaults** () and select **3D Plot Group** ().
- 3 Right-click the **3D Plot Group** () node and select **Surface** ().
- 4 In the **Settings** window for **Surface** () under the **Expression** section:
 - In the **Expression** field enter T.
 - In the **Unit** field enter K.
 - Click the **Description** check box and enter Temperature in the text field.
- 5 Keep all other default settings.
- 6 Right-click the **Surface** () node and select **Rename** (or click **Surface** and press F2). In the **Rename Surface** dialog box, in the **New label** field enter Temperature. Click **OK**.
- 7 Right-click **Result Defaults I** () and select **Plot Defaults** ().
- 8 Right-click the **Plot Defaults I** () node and select **Default Scalar Plot** ().
 - In the **Expression** field enter T.
 - In the **Description** field enter Temperature. This makes temperature the default for all scalar plots.

MAKING A THERMOELECTRIC DEVICES PHYSICS AREA


To add the Thermoelectric Effect interface to a new physics area for Thermoelectric Devices under the Heat Transfer branch:

- 1 Under **Definitions Library** right-click **Physics Areas** () and select **Physics Area** ()
- 2 In the **Settings** window for **Physics Area** under **Parent Area**, expand the **Root>Heat Transfer** folder () . Right-click **Heat Transfer** and select **Set as Parent** () to sort this physics area under the **Heat Transfer** branch in the Model Wizard and Add Physics windows.
- 3 In the **Physics Area Settings** section:
 - In the **Name** field enter `ThermoelectricDevices`.
 - In the **Description** field enter `Thermoelectric Devices`.
 - The default **Icon** is `physics.png` () , which is appropriate for this physics. Otherwise, click **Browse** to use another icon.
 - In the **Weight** field enter 10 to make the **Thermoelectric Devices** area appear last in the list under **Heat Transfer** (the higher the weight, the lower position the physics area gets in the tree of physics interfaces).
- 4 Click the **Thermoelectric Effect** node. In the **Settings** window for **Physics Interface**, expand the **Physics Area** section.
- 5 Expand the **Heat Transfer** folder. Right-click **Thermoelectric Devices** () and select **Set as Parent** () . This sorts the physics interface under the selected physics area.
- 6 Save the file that contains the physics interface as `ThermoelectricEffect.mphphb`.








This completes the definition of the **Thermoelectric Effect** interface. The next section tests the physics interface.

Testing the Thermoelectric Effect Interface

At any time during the implementation of a physics interface using the Physics Builder, you can launch an updated preview of the physics interface so that you can add feature nodes and check that the contents and behavior of the associated **Settings** windows and other functionality is as expected.

To do this select the main node for the physics interface implementation (for example, **Thermoelectric Effect** and then click the **Show Preview** button () on the **Settings** window toolbar (or press F8). An instance of the physics interface then displays at the bottom of the Physics Builder tree.

When you are finished, update COMSOL Multiphysics to check that the new physics interface is in the Model Wizard and that the functionality and settings are as expected.

- 1 From the **Windows** menu or the **Home** toolbar, choose **Physics Builder Manager** ().
 - 2 Under **Archive Browser**, right-click the **Development Files** node () and select **Add Builder File**. Browse to locate `ThermoelectricEffect.mphphb`. Click to select it and then click **Open**.
 - 3 From **File** menu select **New** ().
 - 4 Click **Model Wizard** ().
 - 5 Select a space dimension, **3D** () for example.
 - 6 On the **Select Physics** page, under **Heat Transfer>Thermoelectric Devices** click **Thermoelectric Effect (tee)**. Click **Add**.
 - 7 Click **Study** ().
 - 8 On the **Select Study** page, verify that **Stationary** is the only available study type under **Preset Studies**. Select it and click **Done** ().
 - 9 In the **Model Builder**, verify that the default nodes appear as expected and that their **Settings** windows contain the user inputs specified.
 - 10 Continue by building an example model (see [Example Model — Thermoelectric Leg](#)) to verify that the Thermoelectric Effect interface solves the correct equation using the correct boundary conditions and that you can plot the various physics quantities.
- 11 Correct any errors or problems found and save the Physics Builder file again.



The Physics Builder Manager

When you have successfully created a first instance of a physics interface you can consider improvements or additions for future development. Typically you can save a model file and then reload it after updating the physics definitions to see how it behaves after applying some extensions or corrections.

The Thermoelectric Effect interface has some natural extensions:

- Adding additional boundary conditions for current input and convective cooling.
- Adding a Time Dependent study type, which requires user inputs for density and heat capacity.
- Allowing for the thermal and electric conductivities to be anisotropic.

- Making use of Material Groups in order to be able to reuse material properties from one modeling session to another.
- Using Building Blocks to make the physics interface easier to maintain and extend.
- Creating additional variables for the separate heating contributions from Thomson heating and Joule heating. These correspond to the right-hand terms in the first of the thermoelectric equations:

$$\nabla \cdot (-k\nabla T + P(-\sigma\nabla V - \sigma S\nabla T)) = (-\sigma\nabla V - \sigma S\nabla T) \cdot (-\nabla V)$$

In other words:

$$\begin{aligned} \text{Joule heating} &= \sigma\nabla V \cdot \nabla V = \mathbf{J} \cdot \mathbf{E} \\ \text{Thomson heating} &= \sigma S\nabla T \cdot \nabla V = \mathbf{J} \cdot S\nabla T \end{aligned}$$

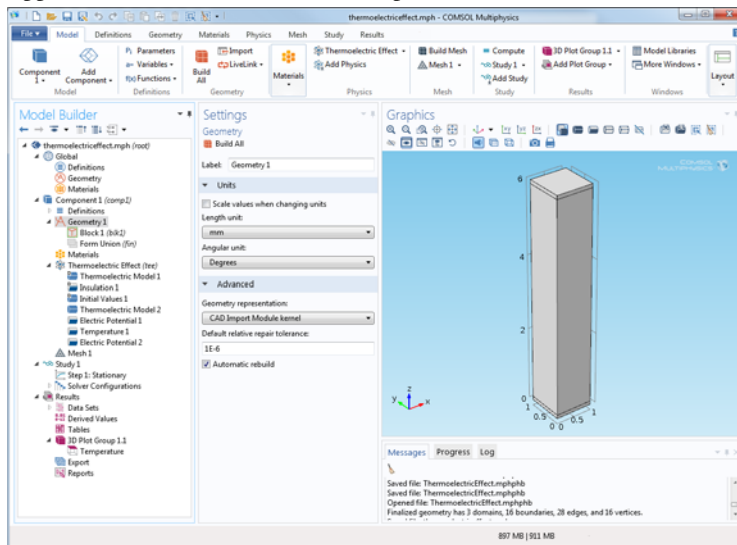
Example Model — Thermoelectric Leg

In this section:

- [Introduction to the Thermoelectric Leg Model](#)
- [Results](#)
- [Reference](#)
- [Modeling Instructions](#)

Introduction to the Thermoelectric Leg Model

A thermoelectric leg is a fundamental component of a thermoelectric cooler (or heater). The component in this example is 1-by-1-by-6 mm, capped by two thin copper electrodes. The thermoelectric part is made of bismuth telluride.



The material properties needed are the thermal conductivity, the electric conductivity, and the Seebeck coefficient of copper and bismuth telluride. The material properties

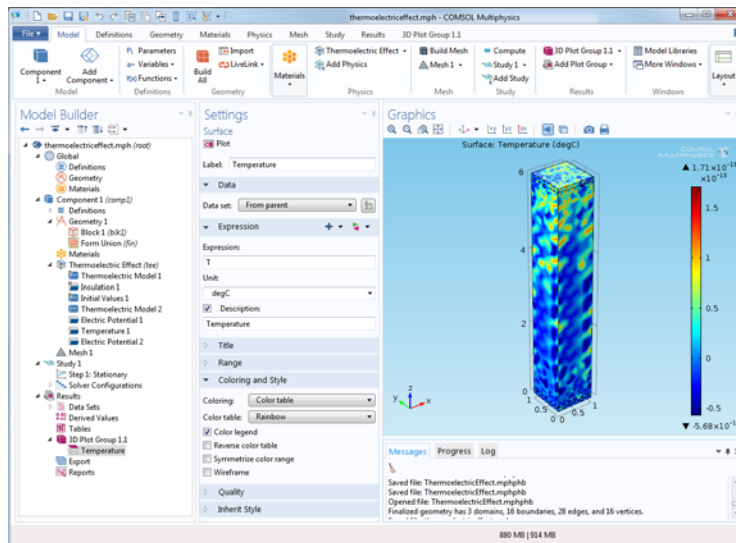
of this model is taken from the paper in [Ref. 1](#):

PROPERTY	SYMBOL AND UNIT	BISMUTH TELLURIDE	COPPER
Thermal conductivity	k (W/(m*K))	1.6	350
Electric conductivity	σ (S/m)	1.1e5	5.9e8
Seebeck coefficient	S (V/K)	p: 200e-6 n: -200e-6	6.5e-6

The bottom electrode surface is held at 0 degrees C and is electrically grounded at 0 V. The top electrode is set to 0.05 V and is thermally insulated. Applying a constraint for only one degree of freedom automatically sets a natural boundary condition for the other.

Results

The results agree with those of [Ref. 1](#) and shows a 61 degree C cooling of the top part of the thermoelectric leg.









Reference

1. M. Jaegle, *Multiphysics Simulation of Thermoelectric Systems — Modeling of Peltier-Cooling and Thermoelectric Generation*, in “Proceedings of the COMSOL Conference 2008,” Hannover. ISBN: 978-0-9766792-8-8.



Modeling Instructions

The following steps show how to build this model using the Thermoelectric Effect interface built in the [Thermoelectric Effect Implementation](#) section.



MODEL WIZARD

- 1 Open COMSOL Multiphysics.
- 2 On the **New** page click **Model Wizard** () then click the **3D** button () on the **Select Space Dimension** page.
- 3 On the **Select Physics** page under **Heat Transfer>Thermoelectric Devices** click **Thermoelectric Effect** () .
- 4 Click **Add** and then the **Study** button () .
- 5 On the **Select Study** page, under **Preset Studies** click **Stationary** () . Click **Done** () .

GEOMETRY MODELING

- 1 Under **Component 1** click **Geometry 1** () .
- 2 In the **Settings** window for **Geometry** select mm from the **Length unit** list.
- 3 Add a **Block** () with **Width** 1 mm, **Depth** 1 mm, and **Height** 6 mm.
- 4 On the **Settings** window expand the **Layers** section. Add two layers to the table: **Layer 1** with **Thickness** 0.1 mm and **Layer 2** with **Thickness** 5.8 mm.







PHYSICS SETTINGS

- 1 Right-click the **Thermoelectric Effect** node () and select **Thermoelectric Model** () . This is in addition to the default node.
- 2 On the **Settings** window for the second **Thermoelectric Model**, add the copper electrodes (domains 1 and 3) to the selection list under **Domain Selection**.

- 3 Under **Thermoelectric Model** replace the defaults with the following:
 - In the **Thermal conductivity** field enter 350.
 - In the **Electric conductivity** field enter $5.9e8$.
 - In the **Seebeck coefficient** field enter $6.5e-6$.




The default **Thermoelectric Model** already has the correct values for bismuth telluride and is assigned to domain 2.

- 4 Right-click **Thermoelectric Effect** () and select **Electric Potential** ().
- 5 On the **Settings** window for **Electric Potential**, select boundary 3 (bottom surface) and keep the default **Electric potential** (0 V).
- 6 Right-click **Thermoelectric Effect** () and select **Temperature** ().
- 7 On the **Settings** window for **Temperature** select boundary 3. In the **Temperature** field replace the default with 273.15 K (0 degrees C).
- 8 Right-click **Thermoelectric Effect** () and select **Electric Potential** ().
- 9 On the **Settings** window for **Electric Potential** select boundary 10 (top surface). Enter an **Electric potential** of 0.05 V.

MESH GENERATION AND COMPUTING THE SOLUTION


A mesh with default parameters is good enough for this very simple model. The default mesh is automatically created if nothing else is specified when solving.

- 1 Right-click the **Study** node and select **Compute** ().

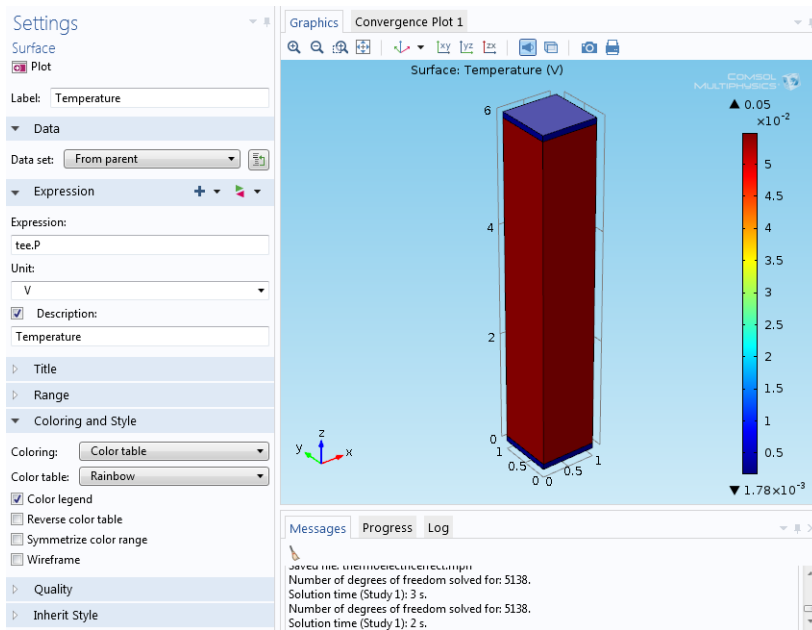
RESULTS AND VISUALIZATION

The default plot under the **3D Plot Group** is for the temperature T. Click the **Temperature** node and under **Expression**, replace the default **Unit** with degC. This is to verify the 61 degree C temperature drop.

To visualize other predefined expressions defined when creating the physics interface:

- 1 Under **Results>3D Plot Group**, click the **Temperature** node.
- 2 In the **Settings** window for **Surface**, click the **Replace Expression** button ().
- 3 Under **Model>Component 1>Thermoelectric Effect** choose a predefined expression, for example **Peltier coefficient** tee.P.

- 4 Click the **3D Plot Group** node and the **Temperature** plot updates in the **Graphics** window.



The Schrödinger Equation

This section is an [Introduction to the Schrödinger Equation](#).

Introduction to the Schrödinger Equation

The *Schrödinger equation* (from the Austrian physicist Erwin Schrödinger) is an equation that describes the behavior of the quantum state of a physical system as it changes in time:

$$E\Psi = \hat{H}\Psi$$
$$i\frac{\hbar}{2\pi}\frac{\partial\Psi}{\partial t} = \hat{H}\Psi$$

where Ψ is the quantum mechanical wave function (the probability amplitude for different configurations of the system) and H is the *Hamiltonian*. \hbar , the Planck constant, over 2π is often called the *reduced Planck constant* (\hbar -bar).

For describing the standing wave solutions of the time-dependent equation, which are the states with definite energy, the equation can be simplified to a stationary Schrödinger equation. The following version of the stationary Schrödinger equation models the atom as a one-particle system:

$$-\nabla \cdot \left(\frac{\hbar^2}{8\mu\pi^2} \nabla\Psi \right) + V\Psi = E\Psi \quad (4-1)$$

The equation parameters are:

- \hbar (approximately $6.626 \cdot 10^{-34}$ Js) is the Planck constant
- μ is the reduced mass
- V is the potential energy
- E is the unknown energy eigenvalue
- Ψ is the quantum mechanical wave function

The physics interface in this example implements this form of the stationary Schrödinger equation.

Schrödinger Equation Implementation

In this section:

- [Overview](#)
- [Schrodinger Equation Interface — Creating It Step by Step](#)

Overview

To implement a physics interface for solving [Equation 4-1](#) above, you need to specify the following items:

- The name and description for the physics interface.
- The supported space dimension for the physics interface.
- The study types (Stationary, Time Dependent, Eigenvalue, and so on) that the physics interface supports.
- The equation, written using a weak formulation, to solve, and the input variables that it needs.
- The boundary conditions that the physics needs, including the default boundary condition, and the inputs that they need.
- Any additional variables that are relevant to define for use in, for example, results analysis and visualization.
- A suitable default plot to be displayed when the solver has finished and possibly custom quantities as default plot expressions for new plots.

NAME AND DESCRIPTION

The name of this physics interface is *Schrodinger Equation* (avoiding using the character “ö” in the interface). The short name is `scheq`. There is also a type, `SchrodingerEq`, which is primarily used by the Java and LiveLink™ for MATLAB® interfaces.

SUPPORTED SPACE DIMENSIONS

The Schrodinger Equation interface is available in all space dimensions.

THE STUDY TYPES

The Schrödinger equation is an eigenvalue equation, so an eigenvalue study is the only applicable study type.

THE EQUATION

With scalar coefficients in the equation, and using C as a replacement for the coefficient $\frac{\hbar^2}{8\mu\pi^2}$, the weak formulation using the COMSOL tensor syntax becomes

$$-C*\nabla\text{psi} \cdot \text{test}(\nabla\text{psi}) - V*\text{psi} \cdot \text{test}(\text{psi}) + \text{lambda}*\text{psi} \cdot \text{test}(\text{psi})$$

In this expression, ∇ is the *nabla* or *del* vector differential operator, and \cdot represents an inner *dot product* (*scalar product*). $*$ represents normal scalar multiplication. The variable lambda represents the eigenvalues (E in Equation 4-1).

The following equation parameters must be defined:

- The reduced Planck constant, which is a predefined physical constant, hbar_const .
- The reduced mass μ , which for a one-particle system like the hydrogen atom can be approximated as

$$\mu = \frac{Mm_e}{M + m_e} \approx m_e \quad (4-2)$$

where M equals the mass of the nucleus and m_e represents the mass of an electron ($9.1094 \cdot 10^{-31}$ kg). The hydrogen nucleus consists of a single proton (more than 1800 times heavier than the electron), so the approximation of μ is valid in this case. The Schrodinger Equation interface therefore includes a user input for the reduced mass μ with a default value equal to the electron mass m_e , which is a predefined physical constant, me_const .

- The potential energy V , which for a one-particle system's potential energy is

$$V = -\frac{e^2}{4\pi\epsilon_0 r} \quad (4-3)$$

where e is the electron charge ($1.602 \cdot 10^{-19}$ C), ϵ_0 represents the permittivity of vacuum ($8.854 \cdot 10^{-12}$ F/m), and r gives the distance from the center of the atom. The Schrodinger Equation interface includes a user input for the potential energy V with a default value of 0. You can easily enter the expression above, where the electron charge e and the permittivity of vacuum ϵ_0 are physical constants (e_const and epsilon0_const , respectively) and r is a distance that you can formulate using the space coordinates in the space dimension of the model.

THE BOUNDARY CONDITIONS

The Schrodinger Equation interface includes the following boundary conditions:

- Typically you assume that the exterior boundary is such that there is zero probability for the particle to be outside the specified domain. Such a zero probability boundary

condition is equivalent to a Dirichlet condition $\Psi = 0$. This is the default boundary condition.

- There is also a Wave Function Value boundary condition $\Psi = \Psi_0$ for the case that you do not want to specify a zero probability. This boundary condition defines one user input for Ψ_0 .
- For axisymmetric models the cylinder axis $r = 0$ is not a boundary in the original problem, but here it becomes one. For these boundaries the artificial Neumann boundary condition $\mathbf{n} \cdot (\nabla\Psi) = 0$ serves as an axial symmetry condition. This is the default boundary condition for axial symmetry boundaries, and COMSOL Multiphysics adds these automatically.

All boundary conditions are exclusive (that is, only one of them can be active for any of the boundaries).




ADDITIONAL VARIABLES




One variable to add is the quantity $|\Psi|^2$, which corresponds to the unnormalized probability density function of the electron's position. By adding it as a variable, you can make it available as a predefined expression in plots and results evaluation.

Schrodinger Equation Interface — Creating It Step by Step

The following steps show how to define the **Schrodinger Equation** interface using the implementation defined in the previous section.

CREATING THE BASICS


- 1 Open COMSOL Multiphysics.
- 2 From the **File** menu (Windows) or the **Options** menu (the cross-platform version), choose **Preferences**. In the **Preferences** dialog box, select **Physics Builder** in the list and then select the **Enable Physics Builder** check box if not selected already.
- 3 From the **File** menu, choose **New** (). On the **New** page, click the **Physics Builder** button (). The **Physics Builder** window replaces the **Model Builder** window on the COMSOL Desktop.
- 4 On the **Home** toolbar, click **Add Physics Interface** () (or right-click the root node (**Untitled.mphphb**) and select **Physics Interface**).

- 5 Go to the **Settings** window for **Physics Interface**. In the **Identifiers** section:
 - In the **Description** field enter Schrodinger Equation. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - Select the **Type** check box and replace the default with SchrodingerEq.
 - In the **Default name and tag** field enter scheq.
 - If you have a custom **Icon** for the interface, click the **Browse** button to locate the icon file. The default is to use the physics.png icon (
- 6 The **Schrodinger Equation** interface should support all space dimensions except 0D. Under **Restrictions** keep the defaults for the **Allowed space dimensions**, which already excludes 0D.
- 7 In the **Allowed study types** list, select the default study types (**Stationary** and **Time dependent**). Click the **Add** button () and select **Eigenvalue** from the **Allowed study types** list. Click **OK**.
- 8 In the **Settings** section, confirm that **Domain** is selected from the **Top geometric entity level** list. This means that the equations in the physics interface apply to the domains in the geometry, which is the case for most physics. Leave the default setting for the **Default frame** list as **Material**.
- 9 It is good practice to save the physics use interface after completing some steps. From the **File** menu, choose **Save** and create a physics interface file, SchrodingerEquation.mphphb in the default location. Click **Save**.

This concludes the initial steps to set up the structure for the physics interface. The next steps add equations, boundary conditions, and variables.

ADDING FEATURES

First declare the dependent variable Ψ :



- 1 Right-click the **Schrodinger Equation** node and from the **Variables** menu select **Dependent Variable Declaration** () . Or on the **Physics Interface** toolbar, click the button with the same name.





- 2 In the **Settings** window locate the **Declaration** section.
 - Keep the default for the **Dependent variable reference** list (**Use physical quantity**).
 - From the **Physical quantity** list select **Dimensionless (1)**.
 - In the **Default variable name** field enter `psi`.
 - In the **Description** field enter `Wave function`.
 - In the **Symbol (LaTeX encoded)** field enter `\psi` (the LaTeX syntax for the Greek letter ψ).
 - Keep the default **Dimension (Scalar)**, because Ψ is a scalar field.
- 3 In the **Preferences** section, both check boxes are selected by default. Keep these settings.



You also need to add a **Dependent Variable** node in the Schrödinger equation domain feature to create the shape function (element type) for the dependent variable in the domain (see Step 16 below).

ADD A SCHRÖDINGER EQUATION MODEL DOMAIN FEATURE

Next add the Schrödinger equation in a domain feature:



- 1 Right-click the **Schrodinger Equation** node and from the **Features** menu select **Domain Feature** ().
- 2 In the **Settings** window locate the **Identifiers** section.
 - In the **Description** field enter `Schrodinger equation model`. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - Select the **Type** check box and replace the default with `SchrodingerEqu`.
 - In the **Default name and tag** field enter `schequ`.
- 3 In the **Restrictions** section, keep the default setting (**Same as parent**) for the **Allowed space dimensions** and **Allow study types** lists. By selecting **Customized** you can restrict the feature to a subset of the allowed space dimensions or study types for the physics.
- 4 In the **Selection Settings** section, confirm that **Active** is displayed in the **Applicable entities** list. Keep the **Override rule** default setting (**Built in**), and the **Override rule** default (**Exclusive**).
- 5 Under **Coordinate Systems**, keep the default settings (**Frame system**) for the **Input base vector system** and **Base vector system** lists. Also keep the default **Frame type** as **Material**.
- 6 Under **Preferences** click to select the **Add as default feature** check box to make this the default feature for all domains.


- 7** Right-click the **Schrodinger Equation Model** node and from the **Inputs** menu select **User Input** ().
- 8** In the **Settings** window locate the **Declaration** section.
- In the **Input name** field enter `mu`.
 - In the **Description** field enter `Reduced mass`.
 - In the **Symbol (LaTeX encoded)** field enter `\mu` (LaTeX syntax for the Greek letter μ).
 - From the **Physical quantity** list select **Mass (kg)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - In the **Default value** field enter `me_const` (the electron mass, which is a built-in physical constant) in the **Default value** field.
- 9** Right-click the **User Input** node and select **Variable Definition** ().
- This user input then becomes available as a variable `schequ.mu` in, for example, the predefined expressions for results evaluation. You can also refer to `mu` directly in the weak equation. Keep all other default settings for this node.
- 10** Right-click the **Schrodinger Equation Model** node and from the **Inputs** menu select **User Input** ().
- 11** In the **Settings** window, locate the **Declaration** section.
- In the **Input name** field enter `V`.
 - In the **Description** field enter `Potential energy`.
 - In the **Symbol (LaTeX encoded)** field enter `V`.
 - From the **Physical quantity** list select **Energy (J)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - Use the default value `0` in the **Default value** field.
- 12** Right-click the **User Input 2** node and select **Variable Definition** ().
- This user input then becomes available as a variable `schequ.V` in, for example, the predefined expressions for results evaluation. You can also refer to `V` directly in the weak equation. Keep all other default settings for this node.

- 13 Right-click the **Schrodinger Equation Model** node and from the **Equations** menu select **Weak Form Equation** (). This is where you specify the equation for the domains in the physics interface.
- 14 In the **Settings** window locate the **Integrand** section. In the **Expression** field, enter $-\hbar^2 \text{const}^2 / (2 * \mu) * \nabla \psi \cdot \text{test}(\nabla \psi) - (V - \text{lambda}) * \psi \cdot \text{test}(\psi)$
This expression implements the Schrödinger equation formulation for this interface. Press Ctrl+Space to get lists of the supported operations, including any special characters. See [Tensor Parser](#) for the keyboard entries to create the del operator (∇) and the dot product (\cdot).
- 15 In the **Selection** section keep the default **Selection** setting (**From parent**), which means it inherits the selection from the top node. Also keep the default **Output entities** setting (**Selected entities**) to use the selected domains as the output. Keep all other defaults.
- 16 Right-click the **Schrodinger Equation Model** node and from the **Variables** menu select **Dependent Variable Definition** ().
- 17 In the **Settings** window, locate the **Definition** section. From the **Physical quantity** list select **Dimensionless (1)**. Lagrange elements are the most widely used and are suitable for this physics interface. The default shape-function order becomes 2. Keep the other defaults.

ADD A DEFAULT ZERO PROBABILITY BOUNDARY CONDITION

Add the default boundary condition:

- 1 Right-click the **Schrodinger Equation** (Physics Interface 1) node and from the **Features** menu select **Boundary Condition** ().
- 2 In the **Settings** window locate the **Identifiers** section.
 - In the **Description** field enter Zero probability. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - Select the **Type** check box and replace the default with ZeroProb.
 - In the **Default name and tag** field enter zpb.
- 3 Keep all the defaults for the **Restrictions**, **Selection Settings**, and **Coordinate Systems** sections.
- 4 In the **Preferences** section, select the **Add as default feature** check box. Keep the **Default entity types** as **Exterior** to make this a default condition on exterior boundaries only.




- 5 Right-click the **Zero Probability** node and from the **Equations** menu select **Constraint** ().
- 6 In the **Settings** window locate the **Declaration** section. Enter $0 - \text{psi}$ in the **Expression** field to make $\Psi = 0$ on the boundary (this expression is set equal to zero by the constraint).
- 7 In the **Shape Declaration** section, from the **Physical quantity** list select **Dimensionless (I)** to declare the correct dimension for the constrained variable Ψ . Keep all other default settings.





Click the **Schrodinger Equation** node. On the **Settings** window under **Default Features** note that the Schrodinger Equation Model and the Zero Probability boundary condition are listed.

ADD A WAVE FUNCTION VALUE BOUNDARY CONDITION

Add the Wave Function Value boundary condition $\Psi = \Psi_0$:



- 1 Right-click the **Schrodinger Equation** node and from the **Features** menu select **Boundary Condition** ().
- 2 In the **Settings** window locate the **Identifiers** section.
 - In the **Description** field enter `Wave function value`. Click the **Rename node using this text** button () to update the node in the **Physics Builder**.
 - Select the **Type** check box and replace the default with `WaveFunc`.
 - In the **Default name and tag** field enter `wvfcn`.
- 3 Keep all the defaults for the **Restrictions**, **Selection Settings**, and **Coordinate Systems** sections.
- 4 Right-click the **Wave Function Value** node and from the **Inputs** menu select **User Input** ().

- 5 In the **Settings** window locate the **Declaration** section.
 - In the **Input name** field enter `psi0`.
 - In the **Description** field enter `Wave function value`.
 - In the **Symbol (LaTeX encoded)** field enter `\psi_0`. This is for the symbol Ψ_0 .
 - From the **Physical quantity** list select **Dimensionless (1)**.
 - Keep the default settings in the **Array type (Single)** and **Dimension (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number.
 - Keep the **Default value** at 0.
- 6 Right-click the **Wave Function Value** node and from the **Equations** menu select **Constraint (R=0)**.
- 7 In the **Settings** window locate the **Declaration** section. Enter `par.psi0-psi` in the **Expression** field to make $\Psi = \Psi_0$ on the boundary. The `par` prefix indicates a local parameter scope and is necessary in order to refer to a user input that is not defined as a variable.
- 8 In the **Shape Declaration** section, from the **Physical quantity** list select **Dimensionless (1)** to declare the correct dimension for the constrained variable Ψ .
This feature uses a user input that should appear in a section in the **Settings** window. The constraint automatically adds an extra section for enabling weak constraints and set the type of constraint, but it is necessary to specify a section for the user input.
- 9 Right-click the **Wave Function Value** node and from the **Inputs** menu select **User Input Group** ().
- 10 In the **Settings** window locate the **Declaration** section. In the **Group name** field enter `WaveFunc_section` and in the **Description** field enter `Wave function`.
- 11 Under the **Group members** list, click the **Add** button () and from the **Group members** list select **User Input 1 (par.psi0)**. Click **OK**.
- 12 In the **GUI Options** section, from the **GUI Layout** list select **Group members define a section**. See [User Input Group GUI Options](#) for more information.

The remaining boundary condition, Axial Symmetry, appears automatically on symmetry boundaries in axisymmetric models.







ADDING AN ADDITIONAL VARIABLE

Add the probability density function $|\Psi|^2$ as a variable that is available in the model and for plotting (when the **Show in plot menu** check box is selected in the **Preferences** section, which is the default setting) or evaluating:

- 1 Right-click the **Schrodinger Equation** node and from the **Variables** menu select **Variable Declaration** ()
- 2 In the **Settings** window locate the **Declaration** section.
 - In the **Variable name** field enter probdens.
 - In the **Description** field enter Probability density function.
 - In the **Symbol (LaTeX encoded)** field enter $\{\mid\psi\mid\}^2$ to display $|\Psi|^2$.
 - For the **Dimension** list, keep the default setting (**Scalar**).
 - From the **Physical quantity** list select **Dimensionless (I)**.
- 3 Keep all the default settings in the **Preferences** section. The **Show in plot menu** check box must be selected for this variable to appear as a predefined expression in plots.
- 4 Right-click the **Variable Declaration I** node and select **Variable Definition** ()
- 5 In the **Settings** window locate the **Definition** section. Enter $\text{abs}(\psi)^2$ in the **Expression** field.






DEFINING DEFAULT PLOTS AND DEFAULT PLOT QUANTITIES

Define a default 2D plot group for plotting the probability density function and make the probability density function the default scalar quantity for user-defined plots:

- 1 Right-click the **Schrodinger Equation** node and select **Result Defaults** ()
- 2 Right-click the **Result Defaults I** node and select **2D Plot Group** ()
- 3 Right-click the **2D Plot Group I** () node and select **Surface** ()
- 4 In the **Settings** window for **Surface** in the **Description** field, select the check box and enter Probability density, and then in the **Expression** field enter probdens.
- 5 Right-click the **Surface I** node and select **Rename** (or press F2). In the **Rename Surface** dialog box, in the **New label** field enter Probability Density. Click **OK**.
- 6 Right-click **Result Defaults I** node and select **Plot Defaults** ()
- 7 Right-click the **Plot Defaults I** node and select **Default Scalar Plot** ()
- 8 In the **Settings** window in the **Expression** field enter probdens. In the **Description** field enter Probability density. This makes the probability density function the default plot for all scalar plots.

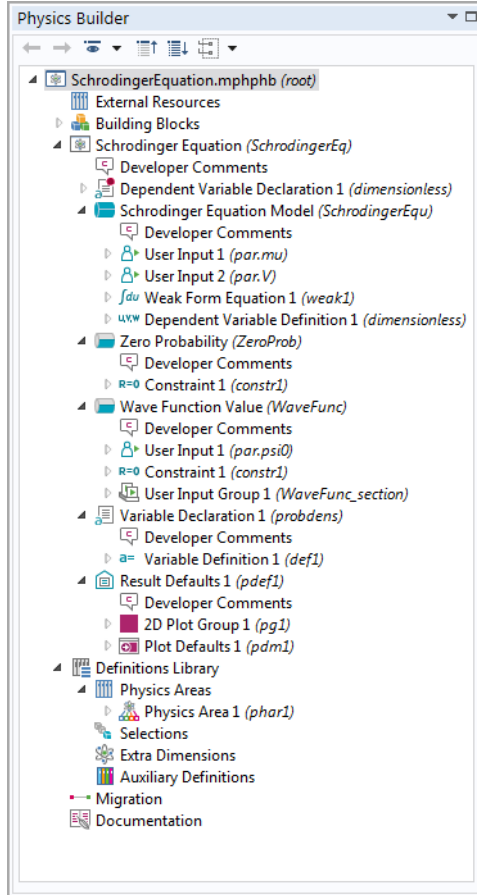
MAKING A QUANTUM MECHANICS PHYSICS AREA

To add the Schrodinger Equation interface to a new physics area for Quantum Mechanics under the Mathematics branch:


- 1 Under **Definitions Library** right-click **Physics Areas** () and select **Physics Area** ()
- 2 In the **Settings** window locate the **Physics Area Settings** section.
 - In the **Name** field enter QuantumMechanics.
 - In the **Description** field enter Quantum Mechanics.
 - The default **Icon** is physics.png () , which is appropriate for this physics. Otherwise, click **Browse** to use another icon.
 - In the **Weight** field enter 10 to make the **Quantum Mechanics** area appear last in the list under **Mathematics** (the higher the weight, the lower position the physics area gets in the tree of physics interfaces).
- 3 Under **Parent Area** click the **Mathematics** node. Click the **Set as Parent** button () . This moves the new physics area under the **Mathematics** node.
- 4 Click the **Schrodinger Equation** node. Locate the **Physics Area** section and click to expand it.
- 5 Expand the **Mathematics** branch and click **Quantum Mechanics**. Click the **Set as Parent** button () . This places the Schrodinger Equation interface under the **Quantum Mechanics** physics area.
- 6 Save the file that contains the physics interface as SchrodingerEquation.mphpb.

This completes the definition of this Schrodinger Equation interface. Save the file that contains the physics interface.







The final node sequence in the Physics Builder tree should match this figure:



Testing the Schrodinger Equation Interface

At any time during the implementation of a physics interface using the Physics Builder, you can launch an updated preview of the physics interface so that you can add feature nodes and check that the contents and behavior of the associated **Settings** windows and other functionality is as expected. To do so, select the main node for the physics interface implementation (for example, **Schrodinger Equation**) and then click the **Show Preview** button () on the **Settings** window toolbar, or press F8. An instance of the physics interface then appears at the bottom of the Physics Builder tree.

When you are finished, update COMSOL Multiphysics to check that the Schrodinger Equation interface appears in the Model Wizard and that the functionality and settings appear as expected.

- 1 From the **Windows** menu or the **Home** toolbar, click **Physics Builder Manager** ().
 - 2 Under **Archive Browser**, right-click the **Development Files** node () and select **Add Builder File**. Browse to locate `SchrodingerEquation.mphphb`, select it, and click **Open**.
 - 3 From **File** menu select **New** ().
 - 4 Click **Model Wizard** ().
 - 5 Select any space dimension.
 - 6 On the **Select Physics** page, under **Mathematics>Quantum Mechanics** select **Schrodinger Equation (scheq)**. Click **Add**.
 - 7 Click **Study** ().
 - 8 On the **Select Study** page, verify that **Eigenvalue** is the only available study type under **Preset Studies**. Select it and click **Done** ().
 - 9 In the Model Builder, verify that the default nodes appear as expected and that their **Settings** windows contain the user inputs that you specified. Also right-click the **Schrodinger Equation** node to add the **Wave Function Value** boundary condition, which is not available by default.
 - 10 Proceed by building an example model (see [Example Model — Hydrogen Atom](#)) to verify that the Schrodinger Equation interface solves the correct equation using the correct boundary conditions and that you can plot the probability density function as a predefined expression.
- 11 Correct any errors or problems that you find and save the physics interface again.

When you have successfully created a first instance of a physics interface you can consider improvements or additions for future development. Typically you can save a model file and then reload it after updating the physics definitions to see how it behaves after applying some extensions or corrections.



Example Model — Hydrogen Atom

In this section:

- [Introduction to the Hydrogen Atom Model](#)
- [Results](#)
- [Modeling Instructions](#)

Introduction to the Hydrogen Atom Model

The quantity $|\Psi|^2$ corresponds to the probability density function of the electron's position in a hydrogen atom. In this example,

$$\mu = \frac{Mm_e}{M + m_e} \approx m_e$$

where M equals the mass of the nucleus and m_e represents the mass of an electron ($9.1094 \cdot 10^{-31}$ kg). The hydrogen nucleus consists of a single proton (more than 1800 times heavier than the electron), so the approximation of μ is valid with a reasonable accuracy. Thus you can treat the problem as a one-particle system.

The system's potential energy is

$$V = -\frac{e^2}{4\pi\epsilon_0 r}$$

where e equals the electron charge ($1.602 \cdot 10^{-19}$ C), ϵ_0 represents the permittivity of vacuum ($8.854 \cdot 10^{-12}$ F/m), and r is the distance from the center of the atom.

In the model, the boundary condition for the perimeter of the computational domain is a zero probability for the electron to be outside the specified domain. This means that the probability of finding the electron inside the domain is 1. It is important to have this approximation in mind when solving for higher-energy eigenvalues because the solution of the physical problem might fall outside the domain, and no eigenvalues are found for the discretized problem. Ideally the domain is infinite, and higher-energy eigenvalues correspond to the electron being further away from the nucleus.

Results

The solution provides a number of the lowest eigenvalues.

Three quantum numbers (n, l, m) characterize the eigenstates of a hydrogen atom:

- n is the principal quantum number.
- l is the angular quantum number.
- m is the magnetic quantum number.

These quantum numbers are not independent but have the following mutual relationships:

$$\begin{aligned}n &= 1, 2, 3, \dots \\0 &\leq l \leq n - 1 \\-l &\leq m \leq l.\end{aligned}$$

An analytical expression exists for the energy eigenvalues in terms of the quantum number n

$$E_n = -\frac{h^2}{8\pi^2\mu a_0^2 n^2}$$

where

$$a_0 = \frac{h^2\epsilon_0}{\pi\mu e^2} \quad (4-4)$$

This expression is called the Bohr radius and has an approximate value of $3 \cdot 10^{-11}$ m.

The first three energy eigenvalues, according to the above expression with $\mu \approx m_e$, are:

- $E_1 \approx -2.180 \cdot 10^{-18}$ J
- $E_2 \approx -5.450 \cdot 10^{-19}$ J
- $E_3 \approx -2.422 \cdot 10^{-19}$ J

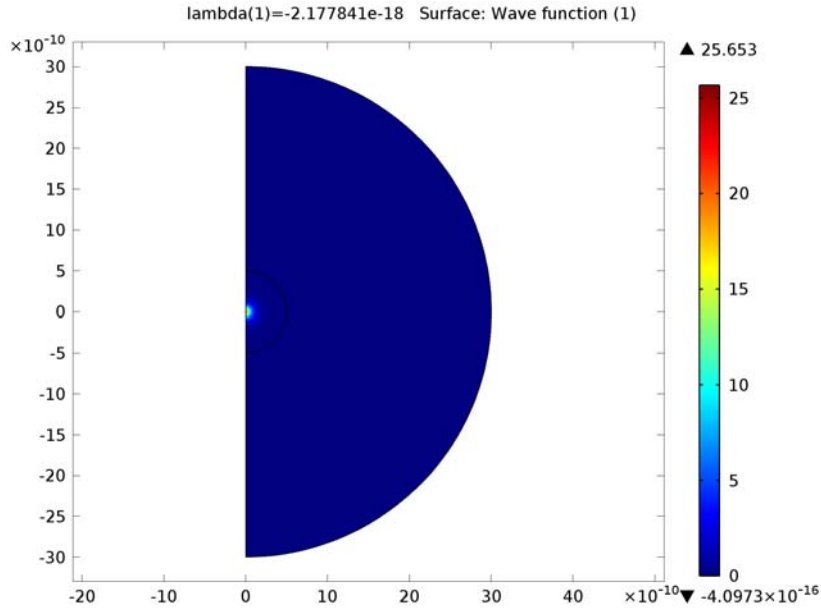


Figure 4-1: The wave function ψ for the first energy eigenvalue.

Comparing these numbers with the computed eigenvalues, you can see a 2-fold degeneracy for $n = 2$ and a 3-fold degeneracy for $n = 3$. This degeneracy corresponds to the following quantum triplets: $(2,0,0)$ and $(2,1,0)$; $(3,0,0)$, $(3,1,0)$, and $(3,2,0)$. The computed values are separated due to the approximate numerical solution.

By refining the mesh and solving again, you can achieve more accurate results. The states with $l = 0$ correspond to spherically symmetric solutions, while states with $l = 1$ or 2 correspond to states with one or two radial node surfaces. The 0 energy level corresponds to the energy of a free electron no longer bounded to the nucleus. Energy levels closer to 0 correspond to excited states.







The wave function by itself has no direct physical interpretation. Another quantity to plot is $|\Psi|^2$, which is proportional to the probability density (unnormalized) function for the electron position after integration about the z -axis. The plot shows the unnormalized probability density function.

To determine the ground state energy, you can use adaptive mesh refinement.




Modeling Instructions

The following steps show how to build this model using the Schrodinger Equation interface.

MODEL WIZARD

- 1 Open COMSOL Multiphysics.
- 2 On the **New** page click **Model Wizard** () then click the **2D Axisymmetric** button () on the **Select Space Dimension** page.
- 3 On the **Select Physics** page under **Mathematics>Quantum Mechanics** click **Schrodinger Equation (scheq)** ()
- 4 Click **Add** and then the **Study** button ()
- 5 On the **Select Study** page, under **Preset Studies** click **Eigenvalue** ()
- Click **Done** ()

GEOMETRY MODELING

- 1 Under **Component 1** click **Geometry 1** ()
- 2 Add a **Circle** ()
- In the **Settings** window under **Size and Shape**, in the **Radius** field enter $3e-9$ (3 nm). In the **Sector angle** field enter 180.
- 3 Under **Position** from the **Base** list select **Center**. In the **r** field enter 0 and in the **z** field enter 0, which centers the circle at the origin.
- 4 Under **Rotation Angle**, in the **Rotation** field, enter -90 degrees to create a semicircle in the right half-plane.
- 5 Click **Geometry 1**, and add a **Circle** ()
- In the **Settings** window under **Size and Shape**, in the **Radius** field enter $0.5e-9$ (0.5 nm). In the **Sector angle** field enter 180.
- 6 Under **Position** from the **Base** list select **Center**. In the **r** field enter 0 and in the **z** field enter 0, which centers the circle at the origin.
- 7 Under **Rotation Angle**, in the **Rotation** field, enter -90 degrees to create a semicircle in the right half-plane.
- 8 Build the geometry by pressing F8. The final geometry consists of two semicircles in the $r > 0$ half-plane.

PHYSICS SETTINGS

- 1 Click the **Schrodinger Equation Model** node. In the **Settings** window the default in the **Reduced mass** field is the electron mass, m_e .

- 2 In the **Potential energy** field enter the following expression:





$$-e_const^2 / (4 * pi * epsilon0_const * sqrt(r^2 + z^2))$$

where `e_const` and `epsilon0_const` are built-in physical constants for the electron charge and the permittivity of vacuum, respectively. `sqrt(r^2+z^2)` is the distance r from the origin. This expression is the potential energy in [Equation 4-3](#).

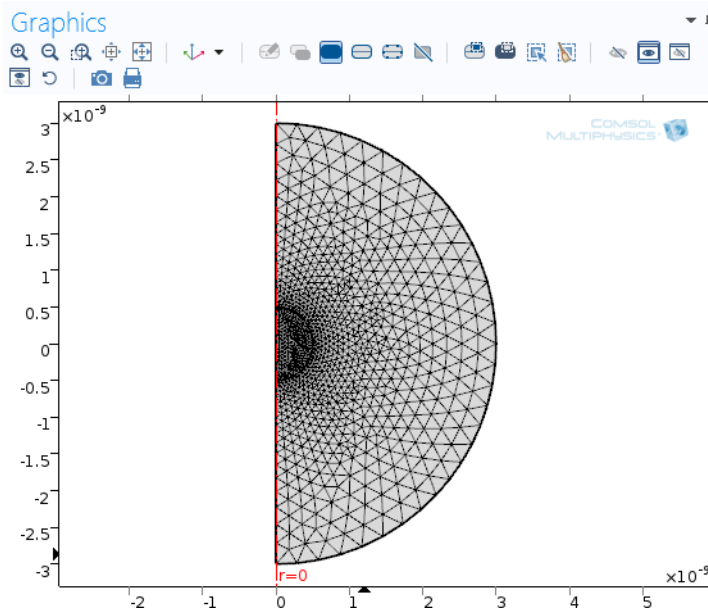
- 3 Verify that the default boundary conditions are correct. Click the **Axial Symmetry** node and confirm it applies to the symmetry boundaries at $r = 0$. Click the **Zero Probability** node to confirm it applies to the exterior boundaries of the geometry.

MESH GENERATION

The reason for the inner circular domain is to use a finer mesh in that part because the solution shows greater variations in the center region than in the outer regions for low-energy eigenvalues.



- 1 Right-click the **Mesh** node () and select **Size** () to add a second **Size** node to define the mesh size in the inner circular domain (Domain 2).
- 2 In the **Settings** window for **Size 1**, select **Domain** from the **Geometric entity level** list.
- 3 Add domain 2 to the **Selection** list.
- 4 In the **Element Size** section click the **Custom** button.
- 5 In the **Element Size Parameters** section, select the **Maximum element size** check box and enter $0.05e-9$ in the corresponding field to use a mesh size no larger than 0.05 nm in domain 2.
- 6 Right-click the **Mesh** node () and select **Free Triangular** () to add a node that meshes domain 2 using the specified mesh size.
- 7 Click the top **Size** node. In its **Settings** window click to expand the **Element Size Parameters** section.
- 8 In the **Maximum element growth rate** field replace the default with 1.1 to make the mesh size grow more slowly toward the perimeter of the geometry.

9 In the Settings window click **Build All** () to create the mesh.




COMPUTING THE SOLUTION

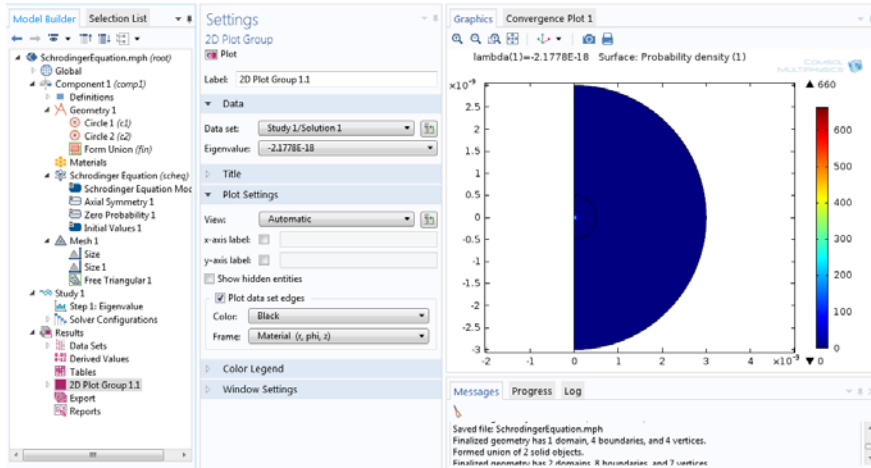
Specify that the eigenvalue solver should solve for 10 eigenvalues, searching around a small negative number where the first energy eigenvalues are located.

- 1 Under **Study I** click **Step 1: Eigenvalue** ().
- 2 In the **Settings** window locate the top **Study Settings** section.
 - In the **Desired number of eigenvalues** field enter 10.
 - In the **Search for eigenvalues around** field enter $-2e-18$.
- 3 In the Settings window click **Compute** () to start the eigenvalue solver.

RESULTS AND VISUALIZATION

By default, COMSOL Multiphysics shows a surface plot of the probability density function $|\Psi|^2$ for the first eigenmode. In the **Settings** window for the **Probability Density** plot, you can also click the **Replace Expression** button () and select **Schrodinger Equation>Wave function (psi)** to plot the variable **psi**, which is the complex-valued wave function for the electron position.

In the **Settings** window for the **2D Plot Group 1**, you can select which eigenmode to plot. Under **Data**, select the corresponding eigenvalue from the **Eigenvalue** list. You can also verify that the eigenvalues correspond to the first energy eigenvalues listed in [Results](#).



I n d e x

- 2D circle, extra dimension 199
- 2D rectangle, extra dimension 199
- 3D sphere, extra dimension 200
- A**
 - absolute tolerance 209
 - activating allowed values 141
 - activation condition 139
 - allowed values 140
 - announcing variables 149
 - applications
 - using physics in 55
 - archives, comparing 54
 - array 229
 - auxiliary elements 228
 - auxiliary settings, feature nodes 111
 - auxiliary settings, multiphysics couplings 113
 - auxiliary settings, physics interface 87
 - average 188
- B**
 - backward compatibility 216
 - Bohr radius 285
 - Boolean input 124
 - boundary condition node 97
 - boundary element equation 171
 - boundary feature 98
 - boundary multiphysics coupling 109
 - building blocks 58
 - built-in physics interfaces 79
 - button 142
 - buttons 127
- C**
 - cards, for GUI layout 27
 - change type, migration 218
 - code editor 60
 - coefficient form equation 169
 - colon product 39
 - comments nodes 222
 - comparing archives 54
 - compile to archive folder 52
 - compiling builder files 51
 - component 65
 - component link 72
 - component settings 162
 - components 58, 64
 - components, for GUI design 23
 - constraint 175
 - constraint settings section 128
 - contact pair feature 100
 - contained feature 110
 - contained interface 85, 192
 - contained multiphysics coupling 191
 - contravariant 44
 - contributing boundary condition 244
 - contributing features 94, 124
 - coupling type contribution 110
 - covariant 44
 - cross product 36
- D**
 - default deformation plot 214
 - default multi scalar plot 215
 - default plot parameters 215
 - default scalar plot 214
 - default vector plot 214
 - definitions library 63
 - degree of freedom initialization 161
 - degree of freedom re-initialization 232
 - del operator 242
 - delimiter, operator for 36
 - Dependencies window 60
 - dependent variable declaration 154
 - dependent variable definition 153
 - dependent variable definitions section 61
 - developer comments 222
 - development files 52

- device 182
- device constants 182
- device equations 184
- device feature 185
- device inputs 182
- device model 180
- device model feature 101
- device variables 183
- DG Wave Element, General Form node 231
- disable allowed study types 89
- disable in solvers 160
- discontinuous Galerkin method 231
- discretization levels 157
- documentation (node) 223
- documentation, HTML format 223
- documentation, Microsoft Word 224
- domain condition node 96
- domain feature 98
- domain multiphysics coupling 109
- dot product, entering 36
- double dot product 39
- double-level inputs 25
- E**
 - edge feature 98
 - edge multiphysics coupling 109
 - eigenvalue transform 211
 - Einstein summation notation 43
 - electron charge 272
 - electron mass 272
 - element (node) 228
 - elements, creating 228
 - elinv 230
 - elpric 230
 - emailing COMSOL 16
 - equation display 70
 - equation display, auxiliary definitions 205
 - equations
 - referencing 71
- event 231
- exclusive boundary condition 244
- exclusive features 94, 124
- external resources 62
- extra dimension link 73
- extra dimension selection 196
- extra dimensions, branch 198
- F**
 - feature 218
 - feature input 136
 - feature input, rename 219
 - feature link 105
 - features (branch) 59
 - field 209
 - files, importing 62
 - frame shape 164
 - Frobenius inner product 39
 - functions 187
- G**
 - general extrusion 189
 - general form equation 167
 - generic feature 92
 - generic multiphysics coupling 107
 - geomdim 228
 - geometric nonlinearity 113
 - global feature 97
 - global multiphysics coupling 108
 - gradient operator 39
 - GUI layout 23
- H**
 - Hamiltonian 270
 - hide in GUI 159
 - HTML 223
 - hydrogen atom example 284
- I**
 - ID interval, ex 198
 - import 62
 - importing files 62
 - initial values 159
 - input modifier 183
 - integer values check 143

- integrated help
 - for documentation 224
- integration 188
- integration over extra dimension 189
- internet resources 15
- J** Joule heating 236
- K** knowledge base, COMSOL 16
- L** LaTeX encoding 71
 - low-level elements 228
- M** material list 132, 135
 - material list, rename 219
 - material parameter, rename 219
 - material property 130
 - material property group 201
 - material property, auxiliary definitions 202
 - matrix symmetry, options for 24
 - maximum 189
 - menu 89
 - menu item 89
 - mesh defaults 207
 - mesh generation 207
 - mesh size 207
 - Microsoft Word format 224
 - migration 216–217
 - minimum 189
 - model inputs 95
 - model object API 216
 - multiphysics coupling 106
 - multiphysics coupling selection filter 195
 - multiphysics couplings
 - at points 110
 - generic 107
 - in domains 109
 - on boundaries 109
 - on edges 109
 - multiphysics couplings (branch) 59
 - multiphysics feature 105
 - multiphysics interface 84
 - multiple ID intervals, extra dimension 198
 - N** nabla operator 36, 272
 - named group member 144
 - natural boundary conditions 239
 - necessary property user inputs section 61
 - necessary variables section 61
 - normal sign, variable for 33
 - O** ODE states collection 164
 - one-particle system 270, 284
 - operators 187
 - outer job parameters 210
 - override rule 203
 - override rule filter 194
 - P** pair feature 99
 - Peltier effect 236
 - periodic feature 104
 - permittivity of vacuum 272
 - physical quantity 202
 - physics area 190
 - physics areas (branch) 190
 - Physics Builder Manager 20, 51
 - Physics Builder window 21
 - physics interface 81
 - physics interface component 66
 - physics interface component link 86
 - physics interface, migration 218
 - physics symbol 114
 - Planck constant 270
 - plot defaults 214
 - plot menu definition 205
 - plug-ins, alternative location of 55
 - point feature 99
 - point multiphysics coupling 110

- port 184
 - port connections 185
 - port model 181
 - predefined multiphysics 191
 - preview 227
 - of documents 223
 - preview, of the physics interface 262, 282
 - probability density function 286
 - for electron's position 273, 284
 - properties (branch) 59
 - property 76
 - property link 77
 - property, migration 218
- Q** quantum numbers 285
- R** radio buttons 29
- real values check 143
 - record 229
 - reduced mass 270
 - reduced Planck constant 270
 - referencing equations 71
 - register development files 52
 - regular expression check 144
 - renaming inputs, migration branch 219
 - result defaults 213
- S** scalar multiplication 272
- scalar product 242, 272
 - Schrödinger equation 270
 - scope, for variables 31
 - section 128
 - Seebeck effect 236
 - segregated step 210
 - selectable input 121
 - selection 193
 - selection component filter 195
 - selection filter sequence 193
 - selection input 121
 - selections (branch) 193
 - selections, specifying 46
 - shape functions 154
 - shape functions, DOF 146
 - shape interpolation element 233
 - shared quantity definition 173
 - singleton features 95
 - solver defaults 208
 - SRC 229
 - stationary 212
 - string 229
 - study sequence 211
 - sublayouts 27
 - SVN repository, comparing against 54
- T** technical support, COMSOL 16
- tensor operations 37
 - tensor parser 36
 - testing a physics interface 51
 - text label 126
 - thermoelectric effect 236
 - Thomson effect 236
 - time-dependent 212
 - transformations, between coordinates
 - 43
- U** Unicode standard 36
- usage condition 66
 - user documentation 221
 - user input 119
 - user input group 125
 - user input, creating 117
 - user input, rename 219
 - user inputs
 - accessing 31
 - user inputs section 61
- V** variable declaration 146
- variable declarations section 61
 - variable definition 150
 - variable definitions section 61

- variables
 - entering 31
 - vs. user inputs 117
- version (branch) 217
- versions, for migration 217

W weak constraint 177

- weak form equation 166
- websites, COMSOL 16
- widgets 23

