

COMSOL Multiphysics®

Physics Builder User's Guide



Physics Builder User's Guide

© 1998–2012 COMSOL

Protected by U.S. Patents 7,519,518; 7,596,474; and 7,623,991. Patents pending.

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

COMSOL, COMSOL Desktop, COMSOL Multiphysics, and LiveLink are registered trademarks or trademarks of COMSOL AB. Other product or brand names are trademarks or registered trademarks of their respective holders.

Version:

May 2012

COMSOL 4.3

Contact Information

Visit www.comsol.com/contact for a searchable list of all COMSOL offices and local representatives. From this web page, search the contacts and find a local sales representative, go to other COMSOL websites, request information and pricing, submit technical support queries, subscribe to the monthly eNews email newsletter, and much more.

If you need to contact Technical Support, an online request form is located at www.comsol.com/support/contact.

Other useful links include:

- Technical Support www.comsol.com/support
- Software updates: www.comsol.com/support/updates
- Online community: www.comsol.com/community
- Events, conferences, and training: www.comsol.com/events
- Tutorials: www.comsol.com/products/tutorials
- Knowledge Base: www.comsol.com/support/knowledgebase

C o n t e n t s

Chapter 1: Introduction

About the Physics Builder	10
What Can You Do With the Physics Builder?	10
Where Do I Access the Documentation and Model Library?	11
Typographical Conventions	13
Overview of the User's Guide	18

Chapter 2: Physics Builder Tools

Overview	20
Using the Physics Builder	20
The Physics Builder Window	21
In This Chapter	22
Creating a Physics Interface	24
Introduction to Creating a Physics Interface.	24
Physics Interface Settings	25
Multiphysics Interface	27
Contained Interface.	28
Physics Interface Component Link	29
Creating Features	31
Feature	31
Periodic Feature	35
Pair Feature.	36
Feature Link.	37
Multiphysics Feature.	38
Contained Feature	39
Device Model Feature	39
Feature Attributes	39

Creating Properties	41
Property	41
Property Link	42
Creating User Inputs	43
Introduction to Creating User Inputs	43
User Input	44
User Input Group	46
Material Property.	46
Feature Input	48
Material List.	49
Activation Condition	50
Additional Requirement	51
Creating Variables	53
About Creating Variables.	53
Variable Declaration	54
Variable Definition	56
Dependent Variable Declaration	58
Dependent Variable Definition.	60
Initial Value	61
Degree of Freedom Initialization	61
Component Settings	62
Specifying Selections	63
Frame Shape	65
Creating Equations	66
Weak Form Equation	66
General Form Equation	67
Coefficient Form Equation	69
Creating Constraints	72
Constraint	72
Creating Device Systems	74
About Device Systems.	74
Device Model	75
Device Feature	76

Port Model	76
Device Variables	77
Device Equations	77
Device Inputs	77
Device Constants	77
Port.	78
Device.	78
Port Connections	78
Input Modifier	79
Creating Operators and Functions	80
Introduction to Operators and Functions	80
Average	80
Integration	80
Maximum.	81
Minimum	81
Function Nodes	81
Creating Components	82
About Creating Components	82
Component.	83
Physics Interface Component	83
Selection Component	83
Component Link	83
Usage Condition	84
Equation Display	86
Building Blocks	88
Components	88
Properties	89
Features	89
Code Editor.	89
External Resources	90
Import.	90
Creating Physics Areas	91
Physics Area	91

Creating Material Property Groups	93
Material Property Group	93
Material Property.	94
Adding Physical Quantities	95
Physical Quantity	95
Creating Override Rules	96
Override Rule	96
The Definitions Library	97
Domain Type Conditions	97
Domain Condition	97
Plot Menu Categories	98
Creating Study and Solver Defaults	99
Field	99
Absolute Tolerance	100
Outer Job Parameters	100
Eigenvalue Transform	100
Study Sequence	101
Creating Result Defaults	102
Plot Defaults	102
Default Scalar Plot	102
Default Vector Plot	103
Default Deformation Plot	103
Default Multi Scalar Plot	103
Default Plot Parameters	103
Creating Mesh Defaults	104
Mesh Size.	104
Entering Names and Expressions	106
Tensor Parser	106
Entering Names	108
Using Coordinate Systems	111
Transformation Between Coordinate Systems	113

Designing the GUI Layout	115
User Inputs and GUI Components	115
User Input Group Options	118
Creating Elements	123
Element	123
GeomDim	123
Src	124
Array	124
Record	124
String	124
Elinv.	125
Elpric	125
Event	125
Degree of Freedom Re-Initialization.	126
The Physics Builder Manager	127
Migration	130
About Backward Compatibility	130
Version	131
Physics Interface	131
Feature	132
Property	132
Change Type	132
Rename Inputs.	132
Migration Links	133
License Settings	133
Adding Comments and Documentation	134
Introduction	134
Physics Interface Documentation.	134
User Documentation	135
Developer Comments	135
The Documentation Node	136
Documentation Text Components	137
The Preview Window	139

Chapter 3: Examples of Custom Physics Interfaces

The Thermoelectric Effect	142
Introduction to the Thermoelectric Effect	142
Equations in the Physics Builder	143
Thermoelectric Effect Implementation	147
Overview.	147
Thermoelectric Effect Physics Interface—Adding It Step by Step	151
Example Model—Thermoelectric Leg	166
Introduction to the Thermoelectric Leg Model	166
Results.	167
Reference	167
Modeling Instructions	168
The Schrödinger Equation	170
Introduction to the Schrödinger Equation	170
Schrödinger Equation Implementation	171
Overview.	171
Schrodinger Equation Interface—Adding It Step by Step	173
Example Model—Hydrogen Atom	182
Introduction to the Hydrogen Atom Model.	182
Results.	182
Modeling Instructions	185

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 User's Guide](#)

About the Physics Builder

In this section:

- [What Can You Do With the Physics Builder?](#)
- [Where Do I Access the Documentation and Model Library?](#)
- [Typographical Conventions](#)

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 these tailored physics interfaces to create your own products and applications.

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

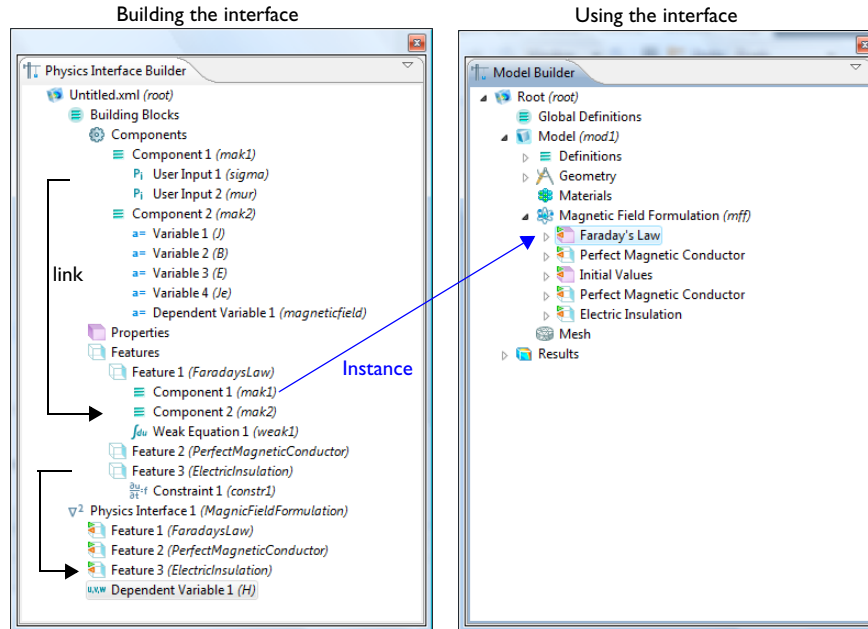


Figure 2-1: The definition of a new physics interface “Magnetic Field Formulation”: in the Physics Builder (left) and the result in the Model Builder (right).

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 interfaces.

Where Do I Access the Documentation and Model Library?

A number of Internet resources provide more information about COMSOL Multiphysics, including licensing and technical information. The electronic

documentation, Dynamic Help, and the Model Library 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 a model or content referenced in a different user's guide. However, if you are using the online help in COMSOL Multiphysics, these links work to other modules, model examples, and documentation sets.

THE DOCUMENTATION

The *COMSOL Multiphysics User's Guide* and *COMSOL Multiphysics Reference Guide* describe all interfaces and functionality included with the basic COMSOL Multiphysics license. These guides also have instructions about how to use COMSOL Multiphysics and how to access the documentation electronically through the COMSOL Multiphysics help desk.

To locate and search all the documentation, in COMSOL Multiphysics:


- Press F1 for Dynamic Help,
- Click the buttons on the toolbar, or
- Select **Help>Documentation** () or **Help>Dynamic Help** () from the main menu

and then either enter a search term or look under a specific module in the documentation tree.


THE MODEL LIBRARY

Each model comes with documentation that includes a theoretical background and step-by-step instructions to create the model. The models are available in COMSOL as MPH-files that you can open for further investigation. You can use the step-by-step instructions and the actual models as a template for your own modeling and applications.

SI units are used to describe the relevant properties, parameters, and dimensions in most examples, but other unit systems are available.

To open the Model Library, select **View>Model Library** () from the main menu, and then search by model name or browse under a module folder name. Click to highlight any model of interest, and select **Open Model and PDF** to open both the model and the documentation explaining how to build the model. Alternatively, click the **Dynamic**

Help button () or select **Help>Documentation** in COMSOL to search by name or browse by module.

The model libraries are updated on a regular basis by COMSOL in order to add new models and to improve existing models. Choose **View>Model Library Update** () to update your model library to include the latest versions of the model examples.

If you have any feedback or suggestions for additional models for the library (including those developed by you), feel free to contact us at info@comsol.com.

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 case number is sent to you by email.

COMSOL WEB SITES



Main Corporate web site	www.comsol.com
Worldwide contact information	www.comsol.com/contact
Technical Support main page	www.comsol.com/support
Support Knowledge Base	www.comsol.com/support/knowledgebase
Product updates	www.comsol.com/support/updates
COMSOL User Community	www.comsol.com/community

Typographical Conventions

All COMSOL user guides use a set of consistent typographical conventions that make it easier to follow the discussion, understand what you can expect to see on the Graphical User Interface (GUI), and know which data must be entered into various data-entry fields.

In particular, these conventions are used throughout the documentation:

- Click text **highlighted in blue** to go to other information in the PDF. When you are using the online help desk in COMSOL Multiphysics, these links also work to other modules, model examples, and documentation sets.

- A **boldface** font indicates that the given word(s) appear exactly that way on the COMSOL Desktop (or, for toolbar buttons, in the corresponding tooltip). For example, the **Model Builder** window () is often referred to and this is the window that contains the model tree. As another example, the instructions might say to click the **Zoom Extents** button () , and this means that when you hover over the button with your mouse, the same label displays on the COMSOL Desktop.
- The names of other items on the COMSOL Desktop that do not have direct labels contain a leading uppercase letter. For instance, the Main toolbar is often referred to—the horizontal bar containing several icons that are displayed on top of the user interface. However, nowhere on the COMSOL Desktop, nor the toolbar itself, includes the word “main.”
- The forward arrow symbol > is instructing you to select a series of menu items in a specific order. For example, **Options>Preferences** is equivalent to: From the **Options** menu, choose **Preferences**.
- A **Code** (monospace) font indicates you are to make a keyboard entry in the user interface. You might see an instruction such as “Enter (or type) 1.25 in the **Current density** field.” The monospace font also is an indication of programming code, or a variable name. An italic *Code* (monospace) font indicates user inputs and parts of names that can vary or be defined by the user.
- An *italic* font indicates the introduction of important terminology. Expect to find an explanation in the same paragraph or in the Glossary. The names of other user guides in the COMSOL documentation set also have an italic font.

THE DIFFERENCE BETWEEN NODES, BUTTONS, AND ICONS

Node: A node is located in the **Model Builder** and has an icon image to the left of it. Right-click a node to open a context menu and to perform actions.


Button: Click a button to perform an action. Usually located on a toolbar (the main toolbar or the Graphics toolbar, for example), or in the upper-right corner of a settings window.

Icon: An icon is an image that displays on a window (for example, the **Model Wizard** or **Model Library**) or displays in a context menu when a node is right-clicked. Sometimes selecting an item with an icon from a node’s context menu adds a node with the same image and name, sometimes it simply performs the action indicated (for example, **Delete**, **Enable**, or **Disable**).

KEY TO THE GRAPHICS

Throughout the documentation, additional icons are used to help navigate the information. These categories are used to draw your attention to the information based on the level of importance, although it is always recommended that you read these text boxes.


Caution

A Caution icon  is used to indicate that the user should proceed carefully and consider the next steps. It might mean that an action is required, or if the instructions are not followed, that there will be problems with the model solution, for example:



This may limit the type of boundary conditions that you can set on the eliminated species. The species selection must be carefully done.


Important

An Important icon  is used to indicate that the information provided is key to the model building, design, or solution. The information is of higher importance than a note or tip, and the user should endeavor to follow the instructions, for example:



Do not select any domains that do not conduct current, for example, the gas channels in a fuel cell.


Note

A Note icon  is used to indicate that the information may be of use to the user. It is recommended that the user read the text, for example:



Undo is not possible for nodes that are built directly, such as geometry objects, solutions, meshes, and plots.

Tip


A Tip icon  is used to provide information, reminders, short cuts, suggestions of how to improve model design, and other information that may or may not be useful to the user, for example:



Tip

It can be more accurate and efficient to use several simple models instead of a single, complex one.

See Also


The See Also icon  indicates that other useful information is located in the named section. If you are working on line, click the hyperlink to go to the information directly. When the link is outside of the current document, the text indicates this, for example:



See Also

- [Theory for the Single-Phase Flow Interfaces](#)
- [The Laminar Flow Interface](#) in the *COMSOL Multiphysics User's Guide*

Model







The Model icon  is used in the documentation as well as in COMSOL Multiphysics from the **View>Model Library** menu. If you are working online, click the link to go to the PDF version of the step-by-step instructions. In some cases, a model is only available if you have a license for a specific module. These examples occur in the *COMSOL Multiphysics User's Guide*. The Model Library path describes how to find the actual model in COMSOL Multiphysics.



Model



- [Acoustics of a Muffler](#): Model Library path **COMSOL_Multiphysics/Acoustics/automotive_muffler**
- If you have the RF Module, see [Radar Cross Section](#): Model Library path **RF_Module/Tutorial_Models/radar_cross_section**

Space Dimension Icons

Another set of icons are also used in the Model Builder—the model space dimension is indicated by 0D , 1D , 1D axial symmetry , 2D , 2D axial symmetry , and 3D  icons. These icons are also used in the documentation to clearly list

the differences to an interface, feature node, or theory section, which are based on space dimension.

The following tables are examples of these space dimension icons.

 3D	3D models often require more computer power, memory, and time to solve. The extra time spent on simplifying a model is time well spent when solving it.
 2D	Remember that modeling in 2D usually represents some 3D geometry under the assumption that nothing changes in the third dimension.

Overview of the User's Guide

This *Physics Builder User's Guide* contains information that helps you to get started with creating custom interfaces 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 User's Guide*.



Tip

As detailed in the section [Where Do I Access the Documentation and Model Library?](#) this information is also searchable from the COMSOL Multiphysics software **Help** menu.

TABLE OF CONTENTS AND INDEX

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

TOOLS

The [Physics Builder Tools](#) chapter provides an overview and 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 Interfaces](#) chapter provide two examples to show how to create custom physics interfaces—[The Thermoelectric Effect](#) and [The Schrödinger Equation](#).

Physics Builder Tools

This chapter provides an overview and a description of the tools in the Physics Builder that you can use to create custom physics interfaces for specific applications. Go to the [Overview](#) section to get started.

Overview

In this section:

- [Using the Physics Builder](#)
- [The Physics Builder Window](#)
- [In This Chapter](#)

Using the Physics Builder

STARTING THE PHYSICS BUILDER

To access the Physics Builder, start COMSOL and select **Options>Preferences**. In the **Preference** window, click **Builder Tools**, and then select the **Enable Physics Builder** check box.

With this preference enabled, the **File** menu gets a **New from Physics Builder** option that starts the **Physics Builder**. The Physics Builder file contains a **Physics Builder** window with a tree similar to the Model Tree in the **Model Builder**. From the **View** menu, you can open the **Physics Builder Manager** that allows you to administer testing of Physics Builder files and deployment of entire Physics Builder packages; see [The Physics Builder Manager](#).


SAVING AND OPENING CUSTOM PHYSICS INTERFACES



Saving Physics Builder files works in the same way as ordinary model files (*.mph) except that they have the extension *.mphdev. Opening a file is also similar to opening MPH-files, just make sure that **Physics Builder File (*.mphdev)** is selected from the **Files of type** list in the **Open** dialog box.

TESTING CUSTOM PHYSICS INTERFACES

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

COMSOL searches and uses the Physics Builder files that are located under the **Development Files** branch (📁). Select **View>Physics Builder Area** (🎨) to open the settings window. 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. Click the **Update Builder Files** toolbar button () in the **Physics Builder Manager** window to reload all physics interfaces listed in the **Development Files** branch (). See [The Physics Builder Manager](#) for more information about organizing development files.





The Physics Builder Window

THE PHYSICS BUILDER TREE

The **Physics Builder** window shows the tree containing all physics interfaces 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. 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 INTERFACES

Under each physics interface node, any of the following items can be added:

- A feature or a link to a feature in the **Features** branch () under **Building Blocks** ()
- A property or a link to a property in the **Properties** branch () under **Building Blocks** ()
- A dependent variable declaration for the variable used by the physics interface.




Note

This node does not add a dependent variable to the physics interface, it just declares that it exists. See [Dependent Variable Declaration](#) for more information.

- A variable declaration with variable definitions 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 [Creating Variables](#)).

See [Creating a Physics Interface](#) for more information about creating physics interfaces.

DEFINITIONS LIBRARY

In the **Definitions Library** branch () custom material properties group can be added using available or additional material properties. Also create physical properties in addition to the predefined ones in COMSOL Multiphysics. See [Creating Material Property Groups](#) and [Adding Physical Quantities](#) for more information.

In This Chapter

The rest of this chapter describes these features of the Physics Builder:

- [Creating a Physics Interface](#)
- [Creating Features](#)
- [Creating Properties](#)
- [Creating User Inputs](#)
- [Creating Variables](#)
- [Creating Equations](#)
- [Creating Constraints](#)
- [Creating Device Systems](#)
- [Creating Operators and Functions](#)
- [Creating Components](#)
- [Building Blocks](#)
- [External Resources](#)
- [Creating Physics Areas](#)
- [Creating Material Property Groups](#)
- [Adding Physical Quantities](#)
- [Creating Override Rules](#)
- [Creating Study and Solver Defaults](#)
- [Creating Result Defaults](#)
- [Creating Mesh Defaults](#)
- [Entering Names and Expressions](#)
- [Designing the GUI Layout](#)

- [Creating Elements](#)
- [The Physics Builder Manager](#)
- [Migration](#)
- [Adding Comments and Documentation](#)

Creating a Physics Interface

In this section:

- [Introduction to Creating a Physics Interface](#)
- [Physics Interface Settings](#)
- [Multiphysics Interface](#)
- [Contained Interface](#)

Introduction to Creating a Physics Interface




In the Model Builder you can add physics interfaces and physics interface features in the Model Wizard. Depending on your license, the Model Wizard may 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 a new physics interfaces that also show up in the Model Wizard, either in an existing physics branch or in a new physics branch.

THE MAIN STEPS TO CREATE A PHYSICS INTERFACE

The steps below 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 features of the physics interface. Depending on the application 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) using the physics interface.
- 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 to the tree (right-click the root node and choose **Physics Interface**). Enter the settings available in the Physics Interface settings window, initially in the **Identifiers**, **Restrictions**, and **Settings** sections.
- 5** Add the dependent variable declarations for the variables to solve for.

- 6 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.
- 7 Add the properties to the physics interface in the same way as the features.
- 8 Test the physics interface using the **Preview** toolbar button (). 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 [The Physics Builder Manager](#).

Physics Interface Settings

The settings window for the **Physics Interface** () contains 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 [Creating 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

In the **Type** field define a unique string to identify the physics interface. The string should not conflict with other identifiers for the physics interfaces present in the Model Wizard. The entry in the **Default identifier 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 Java-files. The text written in the **Description** field is the text COMSOL Multiphysics displays for the physics interface in the Model Wizard.

In the Model Wizard and for the physics instance in the Model Builder there is an icon displayed for the particular physics interface. Using the **Icon** field you can Browse to an icon to display for the physics interface if the default 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**.

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.

In the **Allowed study types** list, add the study types that the physics interface can create equations for. The most important alternatives are **Stationary**, **Time dependent**, **Frequency domain**, **Eigenfrequency**, and **Eigenvalue**. There are also several other alternatives, and some of these are only for specific interfaces. For more information see [Study Types](#) in the *COMSOL Multiphysics Reference Guide*.

SETTINGS

From the **Top geometric entity level** list, choose the top level for the governing equations of the physics interface—**Global**, **Domain**, **Boundary**, **Edge**, or **Point**. **Domain** is the default and 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.

You can also choose how the physics interface behaves together with mesh deformation using the **Default frame** list. 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 make this interface responsible for splitting frames into a reference frame and a moving frame. You choose the moving frame from the **Moving frame** list. For example, an interface that computes a displacement of the spatial frame have this check box selected, and the **Moving frame** set to **Spatial**. From the **Geometry shape order** list, select **Same as first dependent variable** (the default) or

Custom. The **Geometry shape order** setting controls the order of polynomials used for representing the geometry shape in the moving frame.



See Also

[Curved Mesh Elements](#) in the *COMSOL Multiphysics Reference Guide*

GUI OPTIONS

Select the **Hide interface in the Model Wizard** check box if required. 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.

OVERRIDE RULE

This section summarizes the override rules defined by all features of the interface. If two features use different override rules, you can fill in the table with rules between categories in different override rules. For more information see [Creating Override Rules](#).

DEFAULT FEATURES

This section has no user input. It contains a list of all default features that you declared 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 always adds the default features in this list to the new physics interface.

VARIABLE DECLARATIONS

This section contains read-only information. It lists all variables that are declared directly under the physics interface node; see [Information Sections](#) for more details.

Multiphysics Interface

A **Multiphysics Interface** (∇^2) is a combination of other physics interfaces, behaving like one single interface. The multiphysics interface inherits all features and properties that all its contained physics interfaces have, which are added through [Contained Interface](#) nodes. It is possible to remove features that for some reason do not work in a multiphysics context. The settings window for the multiphysics interface is very similar to a physics interface, but with some settings removed (because they are inherited from the contained interfaces).

IDENTIFIERS

Identical to the [Identifiers](#) section of the [Physics Interface Settings](#) node.

RESTRICTIONS

Choose **Contained interfaces** from the **Intersect space dimensions** list to restrict the space dimensions by intersecting the allowed space dimensions from the contained interfaces. With the **Custom dimensions and interfaces** option, you can add extra restrictions to the space dimensions. From the **Allowed study types** list choose **Intersection of interface types** 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 Settings](#) node. The top geometric entity level for the multiphysics interface is the maximum level among the contained interfaces.

GUI OPTIONS

Identical to the [GUI Options](#) section of the [Physics Interface Settings](#) node.

OVERRIDE RULE

Identical to the [Override Rule](#) section of the [Physics Interface Settings](#) node.


DEFAULT FEATURES

Similar to the [Default Features](#) section of the [Physics Interface Settings](#) 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.

VARIABLE DECLARATIONS

Identical to the [Variable Declarations](#) section of the [Physics Interface Settings](#) node.

Contained Interface

The **Contained Interface** () is actually a link node, very 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. The settings window has the following sections:

SOURCE INTERFACE

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



- **Local interfaces.** Choose from the interfaces defined in the same file. Choose the interface from the **Link** list.
- **Built in.** Choose from COMSOL's built-in interfaces that are available to the Physics Builder. Select the main resource 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

In the **Add default features from interface** list, choose **Use default features** to add the same default features as the contained interface. If **Customized** is selected, choose 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.

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 [Creating Components](#). The settings window of the physics interface component link contains the following additional section:

SOURCE COMPONENT


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

- **Building blocks.** 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 COMSOL's 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.** 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.


Creating 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:

- [Feature](#)
- [Periodic Feature](#)
- [Pair Feature](#)
- [Feature Link](#)
- [Multiphysics Feature](#)
- [Contained Feature](#)
- [Device Model Feature](#)
- [Feature Attributes](#)

Feature

The settings window for a **Feature** node () has the following sections:

IDENTIFIERS

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 identifier and tag** field is used to generate the tag of newly created instances in the Model Builder. The text entered in the **Description** field is the text COMSOL displays for the feature 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.

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 [Creating a Physics Interface](#)). 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 **Boundary** 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 **Boundary** 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 live on faces in 3D and lines (edges) in 2D. The boundary condition to a line, for example, is a point. The choices **Edge** and **Point** always refer to the geometric entities edge and point no matter what the top level is.

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.

Allowed Domain Types

It is possible to limit the types of entities where a feature can be active. For example, a boundary condition may be limited to only interior boundaries. All other selected boundaries 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 an **Allowed domain types** from the list—**From list** (the default) or **From Condition**.

- If **From list** is selected, select one or all of the options: **Exterior**, **Interior**, or **Pair**. Use the buttons under the list to organize as required.
- If **From condition** is selected, select a **Domain condition source**—**Locally defined** (the default) or **Imported from external resource**. If **Locally defined** is selected, select a **Domain condition** from the list. Click the **Go to Source** button  as required.

Override Rule Source

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

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

- If **Locally defined** is selected, also select a **Link** from the list. Click the **Go to Source** button  as required.
- If **Imported from external resource** is selected, select an Imported file from the list. Click the **Go to Source** button  as required. Then select a **Link** from the list.

Override Type ID

The **Override type id** list specifies if the feature is **Exclusive** or **Contributing** to other features. In the Model Builder, an *exclusive feature* (constraints, fixed values) replaces all previous feature instances for intersecting selections whereas a *contributing feature* (loads, sources) adds to other contributing features sharing the same selection.

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. See [Using Coordinate Systems](#) and [Transformation Between Coordinate Systems](#) for more information about using base vector systems.

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). For more information, see [Using Coordinate Systems](#).

When any of the base vector systems choices are **Selected input coordinate system**, you get an option to filter the supported coordinate systems. 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.

PREFERENCES

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).

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

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 domain level** list. Newly created interfaces then add the feature on this level. The **Default domain type** specify the domain type the default feature is added on.

In the **Advanced preferences** table, you can specify special options for the default feature. By default, all 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 to alter this default behavior.

GUI OPTIONS

Select the **Hide feature from context menus** 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.


INFORMATION SECTIONS

This is a group of sections that contains 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 information sections are valid for the feature:

- **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 identifier 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.

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. 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.

The settings window of a periodic feature is very similar to the [Feature](#) settings window.




See Also

[Identifiers, Restrictions, Preferences, GUI Options, and Information Sections](#) for the [Feature](#) node

SETTINGS

Except for a few removed settings, identical to the same section for the [Feature](#) node; see [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 domain type setting, because the only setting that makes sense is periodic conditions on exterior boundaries.

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. The settings window of a pair feature is very similar to the [Feature](#) node's settings window:



See Also

- [Identifiers, Restrictions, and Information Sections](#) for the [Feature](#) node
- [Pairs](#) in the *COMSOL Multiphysics Reference Guide*

SETTINGS

Except for a few removed settings, identical to the same section for the [Feature](#) node; see [Settings](#). 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 has a special type of 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 lies below the pair in the list, contributes with the pair condition. As a result, you cannot set the category for overriding selection for pair features.


PREFERENCES

Identical to the same section for the [Feature](#) node; see [Preferences](#). It is not possible to add a pair feature as a default feature to a physics interface.

GUI OPTIONS

Identical to the same section for the [Feature](#) node; see [GUI Options](#).

Feature Link

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

SOURCE FEATURE

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


- **Building blocks.** Lets you choose among the features listed under the **Features** branch in the **Building Blocks** branch. You choose the feature from the **Link** list.
- **Built in.** Lets you choose among COMSOL's 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.** Lets you 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.

A feature link also has a **Add as default feature** check box; see [Feature](#).



Information Sections for the [Feature](#) node

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. The settings window of the multiphysics feature is very similar to a feature, but with some settings removed as they are inherited from the contained features. The entire **Restrictions** section is gone, 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 [Feature](#) node
 - [Variable Declarations](#) section for the [Physics Interface Settings](#) node
-


SETTINGS

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

PREFERENCES

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

Contained Feature


The **Contained Feature** node () specifies the features that a multiphysics feature inherits from. This is actually a link node, almost identical to the [Feature Link](#) node. The difference is that the settings window only contains the [Source Feature](#) section.



Note

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.

Device Model Feature

A **Device Model Feature** node () is a combination of a [Feature](#) node and a [Device Model](#) node. It behaves identical 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.

Feature Attributes

A feature can have one **Feature Attribute** node that contains a collection of various settings for its parent feature that you usually do not need to change. In its settings window you find 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.


Creating Properties

A Property () contains user inputs and variables that are important to the entire physics interface. A property instance always exist 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.

In this section:

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

Property

The **Property** node () settings window of a feature contains the following sections:

IDENTIFIER


The **Type** is a unique string that identifies the property, which must be unique among all properties supported by a physics interface. 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.

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, typically under the **Building Blocks** branch. The settings window has the following section:

SOURCE PROPERTY

In the **Links from** list you choose where to look for a certain property. Possible 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 COMSOL's 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.

Creating User Inputs

In this section:

- [Introduction to Creating User Inputs](#)
- [User Input](#)
- [User Input Group](#)
- [Material Property](#)
- [Feature Input](#)
- [Material List](#)
- [Activation Condition](#)
- [Additional Requirement](#)

Introduction to Creating User Inputs

A **User Input** (P_i) is a parameter that shows up in the settings window of the feature instance in the Model Builder. A user input always represent a 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 .MPHDEV). 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.

There are also some special versions of the user input, which add predefined declarations of one or several user inputs with a special relation. 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.

- **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.
- **Material List.** Defines a user input that contains a material selection.

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](#).

User Input

A **User Input** (P_i) for a feature or property becomes a data storage editable by the user for the corresponding instance in the Model Builder (see [Creating User Inputs](#) for an overview). 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, and physical quantity of the user input works 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 just 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 about GUI options.

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](#). A user input group node contains the following sections:

DECLARATION

The **Group name** field is an identifier of the group, which cannot be in conflict with any other user input or user input group. In the **Description** field you can enter a text to display for the group. For groups that represent a section, this description becomes the title of the section. 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, 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

In the **GUI Layout** list you can choose how to organize the group members in the window. There are several options here, which are described in more detail in the section [Designing the GUI Layout](#). 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 [Designing the GUI Layout](#) for a detailed description of all options.

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.

DECLARATION

The type of material property can either be a basic material property or a property that belongs to a predefined material property group. In the **Property type** list, you choose the type you want. The basic material properties belongs to a predefined set of quantities that are commonly used in material libraries. You choose the quantity in the **Physical quantity** list. 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. You set this dimension with the **Dimension** list, which can only contain **Scalar** and **Custom** for a scalar material parameter. For a basic material property, you can also specify the name, description, and symbol similar to variable declaration; see [Variable Declaration](#).

A parameter from a predefined material property group can be taken from a locally defined group, a built-in group, or a group defined in an external builder file. In the **Material property group** list, you select the property group to choose parameter from, which you choose in the **Material property** list.

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

OPTIONS

In addition to the options **From material** and **User defined**, you can add extra items to the list in the feature instance, by adding them to the **Extra list items** table.

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 enables you to choose among all materials added to a model. Use the **Couple to material list** check box to specify that this property gets linked to the material selected in the list. You choose what list to couple against in the **Material list**, either by entering a name or choosing a particular material list reference.

GUI OPTIONS

If you want the material property to disappear when it is inactive, you 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.



See Also

- [Designing the GUI Layout](#) for more information about GUI options
- [Materials](#) in the *COMSOL Multiphysics Reference Guide*

Feature Input

The **Feature Input** node (P_i) 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.

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 by adding them to the **Extra list items** table.

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:
 - 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.

GUI OPTIONS

If you want the feature input to disappear when it is inactive, you 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.



See Also

- [Designing the GUI Layout](#) for more information about GUI options
- [Materials](#) in the *COMSOL Multiphysics Reference Guide*

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, it is possible to add a **Material List** node (**P_i**). 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.

MATERIAL LIST

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



See Also

- [Designing the GUI Layout](#) for more information about GUI options
 - [Materials](#) in the *COMSOL Multiphysics Reference Guide*
-

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. The settings window of the activation condition contains 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 may 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.

As a final option, you can select the **Invert condition** check box to invert the entire condition.

Additional Requirement

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

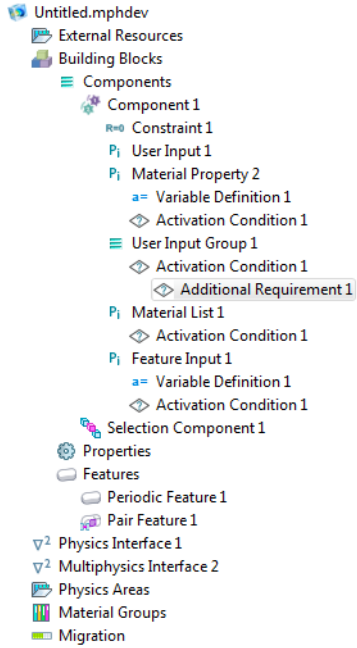


Figure 3-1: A subnode of the Activation Condition node for additional requirements.

Creating Variables

In this section:

- [About Creating Variables](#)
- [Variable Declaration](#)
- [Variable Definition](#)
- [Dependent Variable Declaration](#)
- [Dependent Variable Definition](#)
- [Initial Value](#)
- [Degree of Freedom Initialization](#)
- [Component Settings](#)
- [Specifying Selections](#)
- [Frame Shape](#)

About 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 examples:

- 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 you first declare and then define. 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 that results in 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 you 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 of the physics interface instance in the Model Builder. This means that the user cannot change the name of this variable. Another minor difference is also that it is possible to define the scope of the variable, where the default is physics scope (`<model identifier>.<interface identifier>.<name>`); see [Entering Names and Expressions](#) for information about scopes and variable names. The dependent variable always has a model scope.

Variable Declaration

A **Variable Declaration** (**a=**) specifies various properties of a variable. There are two places to add this node. Go to **Building Blocks>Components>** and right-click **Component**. Then select **Variable Declaration** from the **Variables** list. Right-click **Physics Interface** or **Multiphysics Interface** and select **Variable Declaration** from the **Variables** list.

The settings window for a variable declaration contains the following sections:

DECLARATION

Declare the variable name, description, symbol, dimension, and physical quantity.

The **Variable name** field declares the variable's name, and the **Description** field provides a descriptive text for the variable shown in analysis and variable listings.

It is also possible to declare a symbol that can be used in equation displays and other tools. Enter a LaTeX-encoded string in the **Symbol (LaTeX encoded)** field to define a symbol (`\mu`, for example, to display the Greek letter μ).

The **Dimension** list contains the choices **Scalar**, **Vector (3x1)**, **Matrix (3x3)**, and **Custom**. If you choose **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.

The **Physical quantity** list defines the unit of the variable and is identical to how it works for dependent variable declarations (see [Dependent Variable Declaration](#)).

In [Entering Names and Expressions](#) you can find more detailed information about how you can customize variable names with scopes and parameter values.



PREFERENCES

In the **Preferences** section you specify different options about the variable. These are described briefly below.

- The **Operation between multiple definitions** list. Any variable definition for this variable uses the selected operator to combining 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).
- The **Interpret as right-hand side** check box. Select this 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](#).

- The **Show in plot menu** check box. Select this check box (the default setting) 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. Use the **Category** and **Parent** edit fields to group the variable into a submenu.
- The **Announce variable to feature inputs** check box. 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.

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**. See [Using Coordinate Systems](#) and [Transformation Between Coordinate Systems](#) for more information about selecting base vector 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 [Creating User Inputs](#) for more information). To add this node, go to **Building Blocks>Components>** and right-click **Component**. Then select **Variable Definition** from the **Variables** list.

The settings window for a variable definition contains the following sections:

DEFINITION

For a standalone definition (not being a subnode to a declaration, for example), you type 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. For a subnode definition, there is no **Variable name** field, because it always use the same name as the parent.

In the **Type** list, you specify if this definition is an **Expression** or a **Shape function**. If you choose **Expression**, you can enter an expression in the **Expression** field, according to the rules described in [Entering Names and Expressions](#). For the **Shape function** option, you can choose the shape function from the **Shape function** list. See [Element Types and Discretization](#) in the *COMSOL Multiphysics User's Guide* for more information about the shape functions. The shape function selection becomes disabled 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.

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 about using these lists.

PREFERENCES

By selecting the **Zero out** components check box, you can 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 option **Zero in-plane** does the opposite. This setting has no effect in 3D, for scalars, or non-spatial tensors (length other than 3).

PROTECTION

In this section you can set preferences that enables protection of your entered expression. Select the **Hide expressions 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, and 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 Model 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 (*.mphdev) cannot be read.

ADVANCED

If you have selected **Shape function** in the **Type** list, the **Create a slit for the selected shape** check box becomes editable. When selected, a shape function is not added but instead removed 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.

For shape function definitions, 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 very 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](#).

Dependent Variable Declaration

A **Dependent Variable Declaration** node ([u,v,w](#)) declares a dependent variable used by the 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.

To add this node, right-click **Physics Interface** or **Multiphysics Interface** and select **Dependent Variable Declaration** from the **Variables** list.

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.



The settings window for the dependent variable declaration contains the following sections:

DEFINITION

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 a unique tag (choose **Use physical quantity + tag** and enter a tag in the **Unique tag** field).

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.

It is also possible to choose **None** from the list of physical quantities. This is not a physical quantity, so you have to enter an explicit unit in the **SI unit** field. With the **None** option, it is recommended to use the option **Use physical quantity + tag** in the **Dependent variable reference** list.

The **Default variable name** field declares the default name for the dependent variable, and the **Description** field provides a descriptive text for the variable shown in analysis and variable listings.

It is also possible to declare a symbol that can be used in equation displays and other tools. Enter a LaTeX-encoded string in the **Symbol (LaTeX encoded)** field to define a symbol (μ for example, to display the Greek letter μ).

The **Dimension** list contains the choices **Scalar**, **Vector (3x1)**, **Matrix (3x3)**, and **Custom**. If you choose **Custom**, you can specify a nonstandard dimension (for example, 3x3x3 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

In this section you can select or clear the options, **Show in plot menu** and **Announce variable to feature inputs**. The **Variable Declaration** also uses these options, so you can find a more detailed description under the **Preferences** section for that item.

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**. See **Transformation Between Coordinate Systems** for more information about selecting base vector systems.

Dependent Variable Definition

The **Dependent Variable Definition** node (u,w) is used for the definition of a dependent variable means that you specify the shape function to use and where that shape function is to be used. To add this node, go to **Building Blocks>Components>** and right-click **Component**. Then select **Dependent Variable Definition** from the **Variables** list.

In the settings window of the variable you must first specify the identifier of the dependent variable, which you can select from the **Physical quantity** list if you select **Use physical quantity** (the default) from the **Dependent variable reference** list. As an option, it is also possible to specify an arbitrary unique tag (identifier) by selecting **Use physical quantity + tag** and entering the tag in the **Unique tag** field. It is important that you have a matching dependent variable setting under the physics interface; otherwise you get an error when you try to use the physics interface.

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.

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 about using these lists.

PREFERENCES

The options in this section are identical to the ones in the [Preferences](#) section of the [Variable Definition](#).

ADVANCED

The options in this section are identical to the ones in the [Advanced](#) section of the [Variable Definition](#).

Initial Value

A dependent variable needs an initial value for nonlinear and transient simulations. In the **Initial Value** node ([u,v,w](#)) 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.

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 just becomes `init`, and at solution time this gets model 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, you 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

The options in this section are identical to the ones in the [Preferences](#) section of the [Variable Definition](#).

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 Value](#) 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 this node, go to **Building Blocks>Components>** and right-click **Component**. Then select **Degree of Freedom Initialization** from the **Variables** list.

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 about using these lists.

PREFERENCES

The options in this section are identical to the ones in the [Preferences](#) section of the [Variable Definition](#).

PROTECTION

The options in this section are identical to the ones in the [Protection](#) section of the [Variable Definition](#).

Component Settings

You can override the default names and descriptions for tensor elements with the **Component Settings** node. 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.

The **Create component names by** list contains several options that allows you to create 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
```

you 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
str.append(coord.1,_vel)	#coord.1#-velocity
str.append(coord.2,_vel)	#coord.2#-velocity
str.append(coord.3,_vel)	#coord.3#-velocity



See Also


[Entering Names](#)

Specifying Selections

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, you set up the selection relative to selections that are known in the Model Builder. In the **Selection** list you specify what selection to start from. The list below explains the options in the list,

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 interface, 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.
- **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 [Selections](#) in the *COMSOL Multiphysics Reference Guide*.
- **Component link.** The selection refers to a selection component under the **Building Blocks** branch that defines the selection; see [Selection Component](#).

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 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 variables selection contains the adjacent domains to the selection.
- **Adjacent edges.** The variables selection contains the adjacent edges to the selection.
- **Adjacent points.** The variables selection contains the adjacent points to the selection.

For the option **Adjacent boundaries** you get yet another option to restrict the output boundaries to certain conditions. These 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 just returns all boundaries.
- **Exterior boundaries to the selection.** All boundaries that only has one of the upside domain or downside domain belonging to the domain selection.

- **Interior boundaries to the selection.** The boundaries where the upside domain and downside domain both belong to the domain selection.
- **Exterior boundaries whose upside is in the selection.** Include exterior boundaries that has the upside domain in the domain selection.
- **Exterior boundaries whose downside is in the selection.** Include exterior boundaries that has the downside domain in the domain selection.

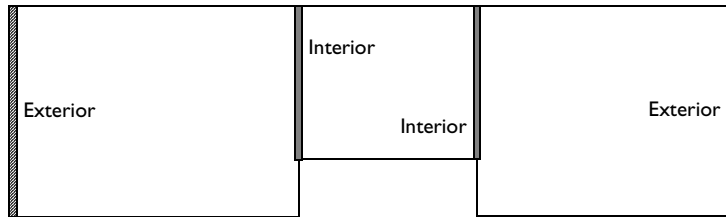


Figure 3-2: A schematic of a domain selection with highlighted exterior and interior boundaries. Note that some exterior boundaries are not highlighted.

Frame Shape

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 this node, right-click a node that can have it as a child node (for example, **Component** or **Feature**). Then select **Frame Shape** from the **Variables** list.

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.

Creating 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 wish to solve for. You can enter the equations in three different 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 this section:

- [Weak Form Equation](#)
- [General Form Equation](#)
- [Coefficient Form Equation](#)

Weak Form Equation

To add the **Weak Form Equation** node () , go to **Building Blocks>Components>** and right-click **Component**. Then select **Weak Form Equation** from the **Equations** list.

The settings window for a **Weak Form Equation** contains 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. For information about writing expressions, see [Entering Names and Expressions](#).

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 about using these lists.

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, see [Using Coordinate 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. See [Using Coordinate Systems](#) for more information about choosing coordinate systems.



See Also

- [Weak Form Modeling](#) in the *COMSOL Multiphysics User's Guide*.
- [Weak Form Theory Background and The Finite Element Method](#) in the *COMSOL Multiphysics Reference Guide*.

General Form Equation

As an alternative to the weak form you can enter the equation using the **General Form Equation** node (\int_{du}). To add this node, go to **Building Blocks>Components>** and right-click **Component**. Then select **General Form Equation** from the **Equations** list.

The settings window for a general form equation contains 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. For information about writing expressions, see [Entering Names and Expressions](#).

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** edit 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 about using these lists.

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, see [Using Coordinate Systems](#).

ADVANCED

By default, the program converts your general form equation to an equivalent weak form before solving a model. It is possible to disable this conversion by selecting the **Use coefficient form element** check box. The only situation when this is necessary, is when you intend to use a special wave form discretization. You must then specify an expression different from NaN in the **Lax-Friedrichs flux parameter** edit field. This wave form also requires that you use the **Nodal discontinuous Lagrange** element in the shape function definition of the dependent variable that the general form refers to.

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. See [Using Coordinate Systems](#) for more information about choosing coordinate systems.



See Also

- [Creating Variables](#)
- [General Form PDE Interface \(g\)](#) in the *COMSOL Multiphysics User's Guide*

Coefficient Form Equation

As an alternative to the weak form you can enter the equation using the **Coefficient Form Equation** node (\int_{du}). To add this node, go to **Building Blocks>Components>** and right-click **Component**. Then select **Coefficient Form Equation** from the **Equations** list.

The settings window for a coefficient form equation contains the following sections:

EQUATION

This section displays the equation that you define by defining the coefficients in the next section.

COEFFICIENTS

There are eight coefficient 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. For information about writing expressions, see [Entering Names and Expressions](#).

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). See [Dependent Variable Definition](#) for more information about specifying dependent variable references.

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 about using these lists.

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, see [Using Coordinate Systems](#).

ADVANCED

By default, the program converts your coefficient form equation to an equivalent weak form before solving a model. It is possible to disable this conversion by selecting the **Use coefficient form element** check box. The only situation when this is necessary, is when you intend to use a special wave form discretization. You must then specify an expression different from NaN in the **Lax-Friedrichs flux parameter** edit field. This wave form also requires that you use the **Nodal discontinuous lagrange** element in the shape function definition of the dependent variable that the coefficient form refers to.

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. See [Using Coordinate Systems](#) for more information about choosing coordinate systems.



See Also

- [Creating Variables](#)
 - [Dependent Variable Definition](#)
 - [Coefficient Form PDE Interface \(c\)](#) in the *COMSOL Multiphysics User's Guide*
-

Creating 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.

Constraints can also be formulated as a Weak Form Equation using an extra degree of freedom. This technique is called weak constraints, which you can read more about in the *COMSOL Multiphysics User's Guide*.

Constraint

To add a **Constraint** node ($R=0$), go to **Building Blocks>Components>** and right-click **Component**. Then select **Constraint** from the **Equations** list.

The settings window for a constraint contains the following sections:

DECLARATION

In the **Expression** field, you enter the constraint expression that the solver forces to zero when solving (for example, the expression $T - 293.15$ [K] makes T equal to 293.15 K). For information about writing expressions, see [Entering Names and Expressions](#). 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 (see [Bidirectional and Unidirectional Constraints](#)). 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 about using these lists.

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 **Use weak constraint** 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; see [Weak Form Equation](#) and [Using Coordinate Systems](#).

Clearing the **Use weak constraint** 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.



See Also

- [Constraint and Using Weak Constraints in the *COMSOL Multiphysics User's Guide*](#)
 - [Group Members Define a Section](#)
-

Creating Device Systems

In this section:

- [About Device Systems](#)
- [Device Model](#)
- [Port Model](#)
- [Device Variables](#)
- [Device Equations](#)
- [Device Inputs](#)
- [Device Constants](#)
- [Port](#)
- [Device](#)
- [Port Connections](#)
- [Input Modifier](#)

About 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

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.

There are various ways to add a **Device Model** node. Go to **Building Blocks>Features>** and add any of these nodes, **Device Model Feature**, **Feature**, or **Periodic Feature**. Then right-click these nodes to add a Device Model from the **Devices** list. Or go to **Building Blocks>Components>** and add a **Component** node. Then right-click the node to add a **Device Model** from the **Devices** list.

The settings window for a device model contains 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 type of 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 will be 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.




See Also

- [Device Feature](#)
- [Device Inputs](#)
- [Device Variables](#)
- [Device Equations](#)
- [Port](#)
- [Port Model](#)
- [Input Modifier](#)

Device Feature

The **Device Feature** node is a special device that can only exist under a **Device Model Feature** node (see [Device Model Feature](#)). 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. Note that this is what 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. The settings window for a **Device Feature** node is identical to a **Device** node.

Port Model

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.

The settings view of the port model contains one section. In the **Type** edit field you 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 the type of that port model in the **Inherited type** edit field.


There are various ways to add a **Port Model** node. Go to **Building Blocks>Features>** and add any of these nodes, **Device Model Feature**, **Feature**, or **Periodic Feature**. Then right-click these nodes to add a Port Model from the **Devices** list. Or go to **Building**

Blocks>Components> and add a **Component** node. Then right-click the node to add a **Port Model** from the **Devices** list

Device Variables

Fill the table with the variables you need for a Device Model. You can use these **Device Variables** (**a=**) in the equations of the same device model or any device model that inherits from it. By specifying an expression for a variable, you directly define an equation for the variable. This is equivalent to define an equation with the right-hand side set to the variable name, and the expression as the left-hand side.

Device Equations

Fill the table with right-hand sides and left-hand sides 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.

Device Inputs


Declares the inputs of a device model. Fill the table in this section with the **Device Inputs** (**P_i**) 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.

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.


Device Constants

Fill the table with the constants you need for a device model. You can use these **Device Constants** (**a=**) in the equations of the same device model or any device model that inherits from it. Enter the name of the constant in the right column, and the constant expression in the left. The expression must be a constant with respect to device variables, but they can depend on ordinary variables declared outside the device context.

Port


A **Port** () is a port model instance that you add to a device model to define the connections that devices use to communicate with each other. In the **Name** edit 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** edit field. If a port model declares a variable named v , and you typed a in the **Name** edit field, you access the variable v by typing $a.v$.

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 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. You enter the name in the **Name** edit field, and in the **Type** edit field you 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 pre-pending 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 $R1.R$.

There are various places where you can add a **Device** node. For example, go to **Building Blocks>Features>** and add any of these nodes, **Device Model Feature**, **Feature**, or **Periodic Feature**. Then right-click these nodes to add a **Device** from the **Devices** list. Or go to **Building Blocks>Components>** and add a **Component** node. Then right-click the node to add a **Device** from the **Devices** list.

Port Connections

The **Port Connections** () defines how devices connect to each other. There are two situations when you can use port connections. The first situation is when you add the port connections to a device model. You use this to define connections between devices that belong to the same device model. Fill the table with the names of the ports that you wish to connect, where the names in the left column connect to the names in the right. 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 .

The other situation is when you add port connections to a device. Then you typically want to define how the ports of that device connects to another device that belongs to another feature. Fill the table with the port name in the left column, and a virtual connection in the right 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 `plus` where you want the user to enter a node that port `p` connects to. Then you enter `p` in the right column, and `par.plus` 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.

Input Modifier

A device model defines a set of device inputs that you define a default value for. If you want to change this default value for a certain device, you do that by adding an **Input Modifier** (`Pi`) to a device. Fill the table with the name of the input you wish to modify, and an expression for the new value.

Creating Operators and Functions

In this section:

- [Introduction to Operators and Functions](#)
- [Average](#)
- [Integration](#)
- [Maximum](#)
- [Minimum](#)
- [Function Nodes](#)

Introduction to Operators and Functions

You can add an operator to a feature, property, or component. Defining an operation is similar to the way it works in the Model Builder but with a few differences.

- The operator names follow 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.
- In the Physics Builder you cannot specify absolute selections as you can in the Model Builder. See [Specifying Selections](#) for more information about this topic.

For more information about the other settings, see [Model Couplings](#) in the *COMSOL Multiphysics Reference Guide*. Not all model couplings are supported as operations in the Physics Builder, and the following sections list the ones supported.

Average

The average operator is almost identical to the average operator you can add in the Model Builder; see [About Model Couplings and Coupling Operators](#) in the *COMSOL Multiphysics Reference Guide*. The main difference is how you specify the operator name and selections, which is described in the [Introduction to Operators and Functions](#) of this chapter.

Integration

The Integration operator is almost identical to the integration model coupling you can add in the Model Builder; see [About Model Couplings and Coupling Operators](#) in the

COMSOL Multiphysics Reference Guide. The main difference is how you specify the operator name and selections, which is described in the [Introduction to Operators and Functions](#) of this chapter.

Maximum

The Maximum operator is almost identical to the Maximum model coupling you can add in the Model Builder; see [About Model Couplings and Coupling Operators](#) in the *COMSOL Multiphysics Reference Guide*. The main difference is how you specify the operator name and selections, which is described in the [Introduction to Operators and Functions](#) of this chapter.

Minimum

The Minimum operator is almost identical to the Minimum model coupling you can add in the Model Builder; see [About Model Couplings and Coupling Operators](#) in the *COMSOL Multiphysics Reference Guide*. The main difference is how you specify the operator name and selections, which is described in the [Introduction to Operators and Functions](#) of this chapter.

Function Nodes


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 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; see [Functions](#) in the *COMSOL Multiphysics Reference Guide*.

Creating Components

In this section:


- [About Creating Components](#)
- [Component](#)
- [Physics Interface Component](#)
- [Selection Component](#)
- [Component Link](#)
- [Usage Condition](#)
- [Equation Display](#)

About Creating Components

As an alternative to define variables, user inputs, and so forth, directly under a feature or property, it is possible to create a collection of such items. These collections are called components, and you find them under the **Components** branch () under **Building Blocks** in the Physics Builder tree. It is convenient to use components 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 simply use a [Component Link](#) in the feature or property that need this section. A component can contain the same items that [Feature](#) and [Property](#) can, with a few exceptions. Here are some examples:

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

Component


In the **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 feature or property. The settings window contains one section for specifying parameters. All other sections contain only read-only information. These sections are identical to these sections for features; see [Information Sections](#).

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.



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. The settings window does not contain any user settings, only read-only information sections. These sections are identical to these sections for physics interfaces; see [Information Sections](#).

Selection Component

With the **Selection Component** node () you create selections that you can use in defining other selections. You can either link to a selection component directly or as part in a operation between selection 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. For more information about specifying a selection, see [Specifying Selections](#).

Component Link

Use a **Component Link** () to include all items defined within a component under the **Building Blocks>Components** branch () as if they were part of the feature containing the link; see [Creating Components](#). The settings window of the component link contains the following additional section:


SOURCE COMPONENT

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




- **Building blocks.** Lets you choose among the components listed under the **Components** branch in the **Building Blocks** branch. You 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. 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.

When the source component has parameters declared, these show up 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.

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 defaults. What kind of conditions you can use differ among contexts because some conditions cannot be evaluated in all contexts. The section description below covers all possible conditions, although some might not be visible depending on the context. The settings window for the **Usage Condition** node contains one section:

USAGE CONDITION

For usage conditions under features or properties you choose the **Explicit** condition in the **Condition** list to get access to the settings described below. With the **And condition** and **Or condition** choices, you can define a usage condition that evaluates as a Boolean operation between other usage conditions. You add those conditions to the **Input condition** list. For usage conditions under **Study/Solver Defaults** () , **Result Defaults** () , and **Mesh Suggestions** () , you always define an explicit condition.

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

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 result defaults and mesh suggestions, the check box is called **Restrict to entity dimensions**. This essentially means the same thing, but the allowed values are instead a subset of the dimensions: **3D**, **2D**, **1D**, and **0D**. The plot and mesh settings are the same for Cartesian geometries and axial symmetry, so these are left out.

When you select the **Restrict to study types** check box, you enable a condition on the study type currently solved for. This is applicable for usage conditions under features, properties, and study/solver defaults. A common example is when you want to define the result of a time derivative, which is

```
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 several other alternatives, and some of these are only for very specific interfaces. For more information on study types, see [Study Types](#) in the *COMSOL Multiphysics Reference Guide*.





The final condition can depend on user inputs, either in the parent feature, parent property, or some property. You enable this condition by selecting the **User input** check box. If the usage condition is under a feature or property, which might 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 the type of condition in the **User input condition** list. The options in this list differ depending on the type of user input you refer to, but the condition can either check if the user input is active, or if it has a certain value.

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 usage conditions under study/solver defaults, result defaults, and mesh suggestion, the **By name** option is the only way to refer to a user input, so these nodes do have that choice. 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.

As a final option, you can select the **Invert condition** check box to invert the entire condition.

Equation Display

With the **Equation Display** () node you can enter pretty-print equations in LaTeX that shows up in the **Equations** section of a physics interface or feature in the Model builder. Enter the LaTeX-encoded expressions in the **Enter equation in LaTeX syntax** field (see [Mathematical Symbols and Special Characters](#) in the *COMSOL Multiphysics Reference Guide* for information about all available LaTeX symbols and commands). There are a couple of tools that you can use to get assistance with entering specific LaTeX commands. Press Ctrl+Space to get a list of predefined operations to choose from. You can also click the **Add Expression** () or **Replace Expression** () toolbar buttons for the same list of operations. Use the **Add Expression** toolbar button to concatenate expressions to the entered expression, and **Replace Expression** to overwrite. Use the **Expression Converter** button () to display the **Expression Converter** dialog box. In this dialog box you can enter an expression and convert it to an equivalent expression using LaTeX commands. Click **Add** or **Replace** to concatenate or overwrite the existing expression in the **Enter equation in LaTeX syntax** field. You can see a preview of your entered expression below the **Equation preview** divider. Each **Equation Display** node has an identifier that the user defines by entering text in the **Name** field. This identifier is necessary when creating equation expressions that link to other expressions.

REFERENCES IN EQUATION EXPRESSIONS

Sometimes it is useful to reuse parts of equations or combine multiple equation expressions into one equation. This can be achieved by 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 in the definitions library.

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 the following syntax can be used to access the current user-defined name of that dependent variable:

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

Building Blocks

Under the root of the tree there is a branch called **Building Blocks** (). Under this branch you can create a library of components, features, or properties that you can build physics interfaces with.


- **Components** (). A list of [Component](#) and [Selection Component](#) nodes.
- **Properties** (). A list of [Property](#) nodes
- **Features** (). A list of [Feature](#) nodes.




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


In this section:

- [Components](#)
- [Properties](#)
- [Features](#)


Components

The **Components** branch () contains 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.
- **Selection Component** (). A selection definition that can either be an explicit selection or a selection being the result of a Boolean operation.
- **Code Editor** (). Opens a text editor window for Java code, providing a possibility to enter coded methods in Java.

The components in this list 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 builder file are also available, if included as an **Import** node under the **External Resources** branch (.


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.

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.



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


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

A **Code Editor** node works exactly like a **Component** node, so you include it through a **Component Link** node. Contact COMSOL support to learn more about the supported 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.


Creating 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](#). When you create your own physics interface you can always put it in an existing category; see [Physics Interface Settings](#) for more information how you do this.

Physics Area

A **Physics Area** () contains a group of physics interfaces for more intuitive and quick access. The settings window of the physics area node contains 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.

You can also control the position of your 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.



[The Model Wizard and Model Builder](#) in the *COMSOL Multiphysics User's Guide*


Creating Material Property Groups

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 () under the **Definitions Library** branch (). A material property group contains a set of material properties, which can come from the list of basic material properties or be custom material properties. When you define a material property group, physics interfaces can use its properties through a material property, and you can also add values for each property in a material library.



In this section:

- [Material Property Group](#)
- [Material Property](#)

Material Property Group

A **Material Property Group** () contains a set of [Material Property](#) nodes that you can refer to when using material properties in your interfaces. The settings window for has the following sections:

PARENT CATEGORIES FOR GROUPS

Each material property group can be categorized under one or several group categories. This is only to simplify access in the **Material** node of the Model Builder. You create a new parent for the current property group by clicking on the **Create New Parent** button () below the tree view. You can also click the **Set As Parent** button () to choose an existing category as parent. For categories that you created, you can change their name in the field under **Category**.

The category tree view also contains COMSOL's built-in property groups () and categories (). You can use a built-in category as parent, but you cannot change the name of it.

MATERIAL PROPERTY GROUP

In the **Name** field you define a unique string that identifies the property group, which must be unique among all property groups available to materials. The text you write

in the **Description** field is the text COMSOL Multiphysics displays for the property group in the **Material properties** section of the **Material** node in the Model builder.



See Also

[Materials](#) in the *COMSOL Multiphysics Reference Guide*


Material Property

The **Material Property** node (P_i) is a subnode to a **Material Property Group** and declares a new property for a node. It is similar to the **Material Property** node used in features or properties but with fewer options. The settings window has the following section:


MATERIAL PROPERTY

In the **Property type** list, you choose if the type of material property is a basic material property or a custom material property. For the option **Basic material property**, you only need to select a material quantity from the **Physical quantity** list. If you choose **Custom material property** you must declare name, description, unit, and dimension, similar to how you do it for the [Variable Declaration](#) node.

Adding Physical Quantities

In addition to the predefined physical quantities that you can use for variable declarations and definitions of material properties, you can add physical quantities to a Physics Builder file. To do this, right-click the **Definitions Library** node () and add a **Physical Quantity** node. To use the added physical quantity in a **Variable Declaration** node, 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.

Physical Quantity

The settings window for a **Physical Quantity** node () contains the following sections:

PHYSICAL QUANTITY

In this section you define the physical quantity using the following properties:

- Type a **Name** for the physical quantity (`seebeck_coefficient`, for example).
- Type a **Description** for the physical quantity (`Seebeck coefficient`, for example).
- Type 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 Guide*.
- You 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.

To use the added physical quantity in a **Material Property** node, 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.



Creating Override Rules

When creating a model in the Model builder, you add several feature instances as children to a physics interface. If these features supports selections, they obey certain rules about how a feature of one types overrides ore 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 categories, and the rules apply between categories. The two standard built-in categories are *exclusive* and *contributing*. Exclusive means that a feature in this category override all other categories. Contributing on the other hand, does not override any other feature, with one exception. The exception is for entity levels below the top-entity level, where all categories overrides all default features.

You add an override rule by right-click the **Definitions Library** node, add an **Override Rule** node. 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 **Override Rule** node () settings window has a table that displays all categories as a matrix with the type identifiers as rows, and what category they overwrite as columns. A selected cell means that the row category overwrites the column category. You can add new categories by clicking the **Add** () button.

The Definitions Library

The **Definitions Library** branch () contains definitions of material property groups, physical quantities, and other definitions that are used by but are not part of a physics interface.

Domain Type Conditions

The **Domain Type Condition** node defines a domain type condition that a feature can link to by choosing **From condition** in the **Allowed domain types** list in the **Settings** section of a feature's window (see [Feature](#)). The conditions are defined in the **Domain Condition** subnodes; see [Domain Condition](#).

Right-click the **Definitions Library** node () to add a **Domain Type Conditions** node (). Right-click to add a **Domain Condition** subnode.

The settings window for a **Domain Type Conditions** node () contains the following section:

SETTINGS

The table displays a list of all **Domain Condition** nodes under this **Domain Type Condition** node. For each row (domain condition) select **Take complement** and choose **Operation** to use with the next condition. The choices in this table produce a complete condition sequence without precedence between operations. For example, if the table has the following three rows:



CONDITIONS	TAKE COMPLEMENT	OPERATION WITH NEXT
Condition 1	X	Intersection
Condition 2	√	Union
Condition 3	X	Intersection


The contents of this table represent the following logical expression: Condition 1 *and not* (Condition 2) *or* Condition 3.

Domain Condition

A **Domain Condition** node defines a single condition of override rules (specified through their feature groups) and a list of allowed domain types; see [Override Rule](#). The condition represents a geometry selection given by the selection of the features that

belong to the given feature groups. The specified domain types define what kind of selection it should be—for example, exterior boundaries to the features selection.


Under the **Definitions Library** node () right-click **Domain Type Conditions** to add the **Domain Condition** subnode ().

The settings window for a **Domain Condition** node () contains the following section:

CONDITIONS

Fill the **Feature group ids** list with the feature groups to use for this condition. Then add the **Allowed domain types**—**Exterior**, **Interior**, or **Pair**—to the list.

Plot Menu Categories

A **Plot Menu Categories** node defines the categories that you can group related variables into; see [Variable Declaration](#). A category shows up as a submenu in the **Replace Expression** () menu of the result nodes in the Model Builder. Specify another category as parent if you need several levels of submenus; otherwise leave the parent field blank. The category of a variable is specified under the variable declaration node below the **Show in plot** menu option. Using plot menu categories you can create a structure where related variables are grouped into separate categories.


Right-click the **Definitions Library** node () to add a **Plot Menu Categories** node ().

The settings window for a **Plot Menu Categories** node () contains the following section:

CATEGORIES

In the table under each column, enter the **Name**, **Description**, and **Parent** to define a new category.


Creating Study and Solver Defaults

To define **Study/Solver Defaults** () for a physics interface, create a solver sequence or part of a solver sequence. This sequence is used as a solver suggestion when solving a model in the Model Builder that includes the interface. There are many alternatives to create a solver default, but the most important rule is to try to specify as little as possible. You can also vary the solver defaults using the [Usage Condition](#) node—for example, to specify one default for 3D 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; see [Solver Features](#) in the *COMSOL Multiphysics Reference Guide*. 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](#)
- [Outer Job Parameters](#)
- [Eigenvalue Transform](#)
- [Study Sequence](#)

Field

Use the **Field** node () to set the default scaling for a dependent variable. 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 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.

SCALING

Set the scaling method in the **Method** list. See [Dependent Variables](#) in the *COMSOL Multiphysics Reference Guide* for more information about scaling.

Absolute Tolerance

Use the **Absolute Tolerance** node if you want to set the default absolute tolerance for a dependent variable. 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 Guide* for more information about setting absolute tolerances.

Outer Job Parameters

If the interface uses predefined parameters that must be in an outer sweep, add these to the **Outer Job Parameters** node (). Enter those 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. The settings window has the following sections:

DECLARATION

In the **Study types** list, choose among the study types that uses the eigenvalue solver. Enter a unique name and description for the transform in the Name and Description edit fields.

RELATION BETWEEN EIGENVALUES

Enter the name of the variable name in the **Eigenvalue name** edit field. This name is used in the following transform definitions. In the **Transform to lambda** edit field, enter the expression that computes the internal eigenvalue, `lambda`, when the transform's

eigenvalue is known. Enter the inverse transform in the **Transform from lambda** edit 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. Right-click the node to add the study steps that are part of the sequence. Note that 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 **Study Sequence** node's settings window contains the following sections:

IDENTIFIERS

Enter the **Name**, **Description**, and **Icon** for the study sequence in the corresponding text fields.

RESTRICTIONS

Select the **Restrict to space dimensions** check box to only allow the study sequence for the chosen space dimensions.

Creating Result Defaults

To define results defaults for a physics interface, you create plot groups as you want them to look after you solve a model in the Model Builder. You can vary the result defaults using the [Usage Condition](#) node. 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 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.



See Also


[Results Analysis and Plots](#) in the *COMSOL Multiphysics Reference Guide*

Plot Defaults


All child nodes to the **Plot Defaults** branch () defines the default expression of variables to use for newly created plots of certain types. The supported types are:

- [Default Scalar Plot](#)
- [Default Vector Plot](#)
- [Default Deformation Plot](#)
- [Default Multi Scalar Plot](#)
- [Default Plot Parameters](#)


Default Scalar Plot

For the **Default Scalar Plot** () , enter the expression in the **Expression** field. The expression follows the rules outlined in [Entering Names and Expressions](#) and 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 in the **Description** field.


Default Vector Plot

For the **Default Vector Plot** () , enter a valid vector variable name in the **Variable name** field. The name follows the rules outlined in [Entering Names](#), and 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 name follows the rules outlined in [Entering Names](#), and 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. The expressions follow the rules outlined in [Entering Names and Expressions](#), and are 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 you enter here.

Creating Mesh Defaults

Right-click a **Physics Interface** or **Multiphysics Interface** node to add a **Mesh Suggestion** node . You can customize the mesh generation if your physics interface is the interface controlling the mesh. If you need to vary the mesh generation depending on space dimension or user input value, you use the [Usage Condition](#) node. This section describes the [Mesh Size](#).

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. Fill in the table with suitable values for the following settings:

- **Maximum element size**
- **Minimum element size**
- **Resolution of curvature**
- **Maximum element growth rate**
- **Resolution of narrow regions**

You have to define these values for each of the predefined mesh sizes:

- **Extremely fine**
- **Extra fine**
- **Finer**
- **Fine**
- **Normal**
- **Coarse**
- **Coarser**
- **Extra coarse**
- **Extremely coarse**



See Also

[Meshing](#) in the *COMSOL Multiphysics User's Guide*.

Entering Names and Expressions

In this section:

- [Tensor Parser](#)
- [Entering Names](#)
- [Using Coordinate Systems](#)
- [Transformation Between Coordinate Systems](#)

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. The other special symbols used by expressions are the (inner) dot product, $A \cdot B$, and the nabla operator, ∇A . You can use predefined key bindings to type them into an expression (see the following table).

SYMBOL	KEY BINDING	UNICODE (HEXADECIMAL)
×	Ctrl+Alt+Keypad*	00D7
.	Ctrl+Alt+Period	00B7
∇	Ctrl+Alt+Comma	2207

The system font must support the special symbols to display them properly, otherwise the expression might not look correct. You can also press Ctrl+Space to get a list of the supported operations that includes any special characters. As a last resort, 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 function that you can use to perform tensor operations—for example, the transpose of a matrix or the inverse of a matrix. 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>
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>

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.

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 model 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.lambda` to access the eigenvalues from the solver.

You may 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.	<code>phys.A => root.mod1.es.A</code>
mod	Replaced by the model scope.	<code>mod.u => root.mod1.u</code>
root	Used as is to define root scope.	<code>root.h => root.h</code>
coord	Identifies the coordinate, and is used together with number-dot	
item	Replaced by a scope that represents the full path to the feature or property.	<code>item.V0 => root.mod1.es.gnd1</code>
par	The subsequent identifier is an input, and the value of that input replaces the entire sequence.	<code>par.sigma => 12[S/m]</code>
mat	The subsequent identifier is a material property, and the material property value, either from the material or the user defined, replaces the sequence.	<code>mat.rho => root.mod1.mat2.def.rho(root.mod1.T)</code>

PREFIX	DESCRIPTION	EXAMPLE
map	The subsequent identifier is an extrusion operator of a pair or a periodic condition.	<code>map.src2dst(c1) => root.mod1.src2dst_p1(c1)</code>
rot	The subsequent identifier defines a rotation of a vector or matrix variable in a periodic condition.	<code>rot.src2dst(A) => {cos(30)*Ax+sin(30)*Ay, sin(30) *Ax-cos(30)*Ay}</code>
minput	The subsequent identifier is a valid model input variable name, and the value of the model input parameter replaces the entire sequence.	<code>minput.T => root.mod1.ht.T</code>
entity	The subsequent identifier is a valid get method from the current feature or property.	<code>entity.tag => init1</code>
dev	Replaced by the current device scope.	<code>dev.v => root.mod1.cir.R1_v</code>

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 model scope, so unscoped names get this scope. Secondly, the user can change their names, so you always specify them by their default name. The default-name lookup has precedence over other variables, so if you want to use a variable with physics scope that has the same name as the default name of a dependent variable, you must use the `phys` 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.mod1.ph` is the physics scope)

```
root.mod1.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>

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 `mod` prefix, which is interpreted as model 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 model scope, then root scope, so it is possible to change the meaning of the function name `sin` if you want.

Using Coordinate Systems

When creating a feature or property, you can define two coordinate systems:

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**. 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.
Material frame system	Uses the coordinate system for the material frame.
Mesh frame system	Uses the coordinate system for the rarely used mesh frame.

The feature determines its frame from the **Material type** list, which has the options **Solid**, **Non-solid**, or **Selectable by user**. The **Solid** option corresponds to the material frame, and the **Non-solid** (typically fluids) option corresponds to the spatial frame. For the **Selectable by user** option, the material type depends on user choice or material setting during a Model builder session. See [Frames](#) in the *COMSOL Multiphysics User's Guide* to get more information about frames.

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 **Base vector type** to **Covariant** or **Contravariant** in a table where each row corresponds to a tensor index.



Note

By default, all tensor indices are contravariant, and this setting is only important for non-scalar, 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; see [The Base Vector System](#). 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.

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_i 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, may 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 two tensors 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

1. G. B. Arfken, H. J. Weber, *Mathematical Methods for Physicists*, Academic press, 1995.

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 Options](#)

User Inputs and GUI Components

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

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 1x1	Any	field (text box)
Scalar or 1x1	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

DIMENSION	ALLOWED VALUES	GUI COMPONENT
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**, and **Anisotropic**, and controls the cells that the user can edit.

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 1x1	Any	Behaves as a single-array vector
Vector (3x1)	Scalar or 1x1	From list	Several combo boxes when the input is a member to a special input group, see User Input Group Options . Otherwise, behaves as a single-array vector with restrictions what you can enter in the table cells.
Matrix (3x3)	Scalar or 1x1	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 Options . Otherwise, behaves as a single-array vector with restrictions what you can enter in the table cells.



Note

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.



See Also

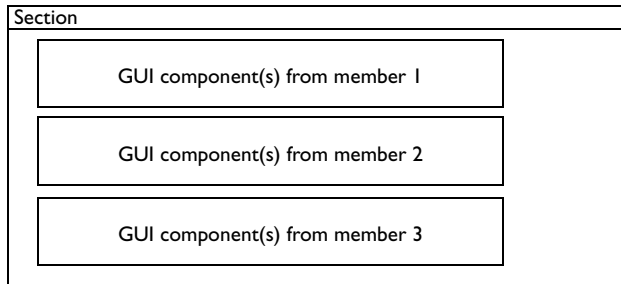
[User Input](#)

User Input Group 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. You can choose among the following layouts:

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.



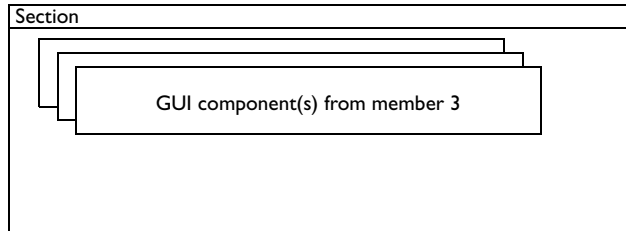
*Figure 3-3: A schematic of the layout of the option **Group members below each other**.*

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 3-4: 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.

If you do not specify a section, there is a default section that may be good enough for very 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.

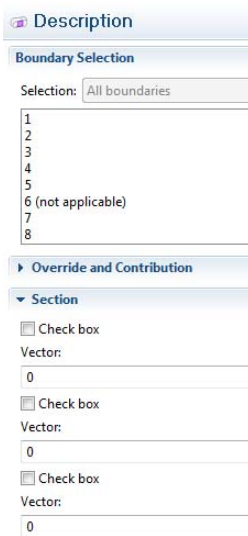


The section that constraint nodes usually adds is not always shown. You must show advanced physics options to see it.

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 an 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 screen shot below.

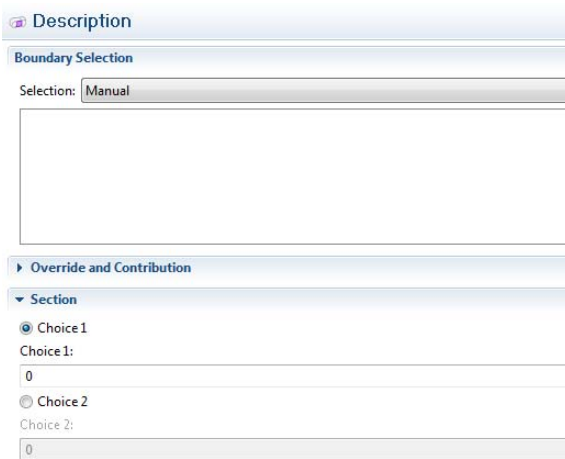


*Figure 3-5: A screen shot of a window created with the layout option **Create a widget for each vector component**.*

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 3-6](#) that

displays an example with two radio buttons. It needs three user inputs, where the first one has two allowed values.



*Figure 3-6: 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 wish 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 a couple of 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
Table data can be saved to file and loaded	Select to add toolbar buttons for saving and loading table content

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



See Also

[User Input Group](#)

Creating 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 as very 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](#)
- [Degree of Freedom Re-Initialization](#)

Element

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.


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 about specifying selections.


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 about specifying selections.


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 child nodes 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>**. 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 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.

Elnv

The **Elnv** 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 **Elnv** from the **Elements** menu.

Elpric

The **Elpric** node () is a special element for computing the eigenvectors and eigenvalues of square 3-by-3 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.

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 User's Guide*. 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.

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 Also

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.


The Physics Builder Manager

With the **Physics Builder Manager** window () you manage testing and compilation of your Physics Builder files. Testing is when you temporarily include one or more development files (*.mphdev) 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, from the main menu select **View>Physics Builder Manager**.

THE DEVELOPMENT FILES

The files listed are included in your COMSOL session. This means that your interfaces pops up 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, you right-click the **Development Files** node () and choose **Add Builder File**. If you right-click on 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 **Update Builder Files** toolbar button () to re-read all files into the current session.



Note

If you save a model file (*.mph) that uses one of your new interfaces, you must make sure that the same 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. 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 COMSOL to identify the description.

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.



Caution

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 an compressed archive (*.jar). Furthermore, they might also contain encrypted expressions that you cannot read or change.

THE BUILDER ARCHIVES

Under the **Builder 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.

You use the option **Export As Plugin** 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.
Finally, you have to restart COMSOL before you can use the new plug-in.

Migration

In this section:

- [About Backward Compatibility](#)
- [Version](#)
- [Physics Interface](#)
- [Feature](#)
- [Property](#)
- [Change Type](#)
- [Rename Inputs](#)
- [Migration Links](#)
- [License Settings](#)

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 a COMSOL Multiphysics model files created in the old version of your 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.

Another situation is if the users has saved his model as a Model Java-file. The change you made in the interface can then break that Model Java-file, 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 is situations when you cannot avoid breaking old Model Java-files.



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 if 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.

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 a older version to the next version node.

Physics Interface


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 very similar manner.


Feature

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

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 the fact 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 Java-file 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.

License Settings

If you define a feature that is available only if the license includes one or some of the modules in the COMSOL Multiphysics product suite, add a **License Settings** node () to any of the feature nodes. The settings window contains the following section:

Adding Comments and Documentation

Introduction

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 implementations, 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. In the main **Documentation** nodes you can add the “topic description,” which is the concise description that the **Help** window displays when you click a node. For the in-depth documentation, you can 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** in the **Builder Tools** section. 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 **User Documentation** node () contains the end-user documentation for a physics interface node, feature node, or other builder component that needs 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 required.

In the **Section Heading** section, you add the following:

- 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 features, are references to the contents of the **Description** fields in those nodes' settings windows.
- A **Topic description**, which is the concise description of the node that appears in the context help in the **Help** window when you click the node. You can use the **Label** font above and the characters and dashes below the text field to format parts of the text for the topic description. The same references as for the **Heading** appears in the default text.





Developer Comments

The **Comments** node () contains developer comments about the implemented feature or other builder component. 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.

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.
- **Help plug-in**, for creating the document as a plug-in of Eclipse Help format 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 as Help Plug-in

When creating a help plug-in, you need to specify the following plug-in location details and properties

- Enter the output directory for the help plug-in in the **Plug-in output directory** field, or click **Browse** to open the **Specify Plug-in Output Directory** dialog box and browse to the desired location.
- Enter a **Plug-in prefix** to use a common name space for your help plug-ins. You can specify a standard plug-in prefix for your organization on the **Builder Tools** page in the **Preferences** dialog box.

- Enter the plug-in's name and title in the **Plug-in name** and **Plug-in title** fields, respectively. The complete plug-in name will be formed from the plug-in prefix and plug-in name. The plug-in title is the title that will appear in the **Contents** tree on the left side of the standalone **Help** window or on the **Contents** page of the **Help** window for Dynamic help integrated in the COMSOL Desktop.
- From the **Add to** list, select **None** (the default) to not link the plug-in to the contents of any COMSOL product, or select COMSOL Multiphysics or any of its modules or LiveLink products to add the documentation to one of the products. As an advanced option, you can select **Custom** to link to a custom plug-in that you specify in the **Link to plug-in named** field and the **Link to anchor ID** field. The custom plug-in must extend the org.eclipse.help.toc extension point and contain the specified anchor for your documentation plug-in to link to.
- In the **Vendor** field you can enter the name of your organization. This name will appear in the manifest file inside the generated plug-in.
- Enter the version of the plug-in in the **Plug-in version** field. The default version is 1.0.0. Like the vendor name, this number is written to the plug-in's manifest file.

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**. The following table lists all documentation components:

TABLE 3-1: 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.

TABLE 3-1: DOCUMENTATION COMPONENTS













DOCUMENTATION COMPONENT	ICON	DESCRIPTION
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
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 3-1: DOCUMENTATION COMPONENTS

DOCUMENTATION COMPONENT	ICON	DESCRIPTION
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 Guide* 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.

Examples of Custom Physics Interfaces

The two examples in this chapter show how to create custom physics interfaces for two 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 (MPHDEV-files) and model examples (MPH-files) for these two custom physics interfaces are included in the COMSOL Multiphysics installation. You find the files in the `demo/builder` directory under your COMSOL Multiphysics installation directory.

The Thermoelectric Effect

In this section:

- [Introduction to the Thermoelectric Effect](#)
- [Equations in the Physics Builder](#)

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:

$$\mathbf{E} = -\nabla V$$

$$Q = \mathbf{J} \cdot \mathbf{E}$$

where \mathbf{E} is the electric field and Q is the Joule heating.

Conservation of heat energy and current gives:

$$\rho C \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{q} = Q$$

$$\nabla \cdot \mathbf{J} = \frac{\partial \rho_c}{\partial t}$$

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:

$$\nabla \cdot \mathbf{q} = Q$$

$$\nabla \cdot \mathbf{J} = 0$$

More explicitly, the thermoelectric equations become:

$$\nabla \cdot (-k \nabla T + P(-\sigma \nabla V - \sigma S \nabla T)) = (-\sigma \nabla V - \sigma S \nabla T) \cdot (-\nabla V)$$

$$\nabla \cdot (-\sigma \nabla V - \sigma S \nabla T) = 0$$

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:

$$\begin{aligned}
-\int_D \mathbf{q} \cdot \nabla v_T + \int_B \mathbf{n} \cdot \mathbf{q} v_T &= \int_D Q v_T \\
-\int_D \mathbf{J} \cdot \nabla v_V + \int_B \mathbf{n} \cdot \mathbf{J} v_V &= 0
\end{aligned}$$

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 \mathbf{J}_0 , respectively, the equations become:

$$\begin{aligned}
-\int_D \mathbf{q} \cdot \nabla v_T + \int_B q_0 v_T &= \int_D Q v_T \\
-\int_D \mathbf{J} \cdot \nabla v_V + \int_B \mathbf{J}_0 v_V &= 0
\end{aligned}$$

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:

$$\begin{aligned}
0 &= \mathbf{q} \cdot \nabla v_T + Q v_T \\
0 &= \mathbf{J} \cdot \nabla v_V
\end{aligned}$$



Note

The COMSOL convention collects all terms on the right side.

Using Physics Builder syntax, the right-side expressions become:

$$\begin{aligned}
\mathbf{q} \cdot \text{test}(\nabla T) + Q \text{test}(T) \\
\mathbf{J} \cdot \text{test}(\nabla V)
\end{aligned}$$

and this is what you type into the weak form text fields when creating the thermoelectric 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. This implementation in this example bundles these two conditions into one single insulation boundary condition. As a matter of fact, 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 Physics Interface—Adding 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 an identifier, `ThermoelectricEffect`, which is used by the Java and 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 type (and perhaps even other study types for more exotic applications). In this example, stationary is the only study type.

THE EQUATIONS

The first equation is called a Heat Transfer Model and is represented by the following weak 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 represents the *dot product (scalar product)*.

The second equation is called the Thermoelectric Model and is represented by the following weak equation:

$$\mathbf{q} \cdot \text{test}(\nabla T) + Q \text{test}(T) \\ \mathbf{J} \cdot \text{test}(\nabla V)$$

The weak formulation using the COMSOL tensor syntax becomes

$$\mathbf{q} \cdot \text{test}(\nabla T) + Q * \text{test}(T) \\ \mathbf{J} \cdot \text{test}(\nabla V)$$

where $*$ is ordinary multiplication between scalars.

A number of parameters is defined in order to efficiently use the COMSOL tensor syntax for defining the weak 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 $1.6[\text{W}/(\text{m} \cdot \text{K})]$, 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 $1.1 \times 10^5[\text{S}/\text{m}]$, 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 $200 \times 10^{-6}[\text{V}/\text{K}]$, 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 physics 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.

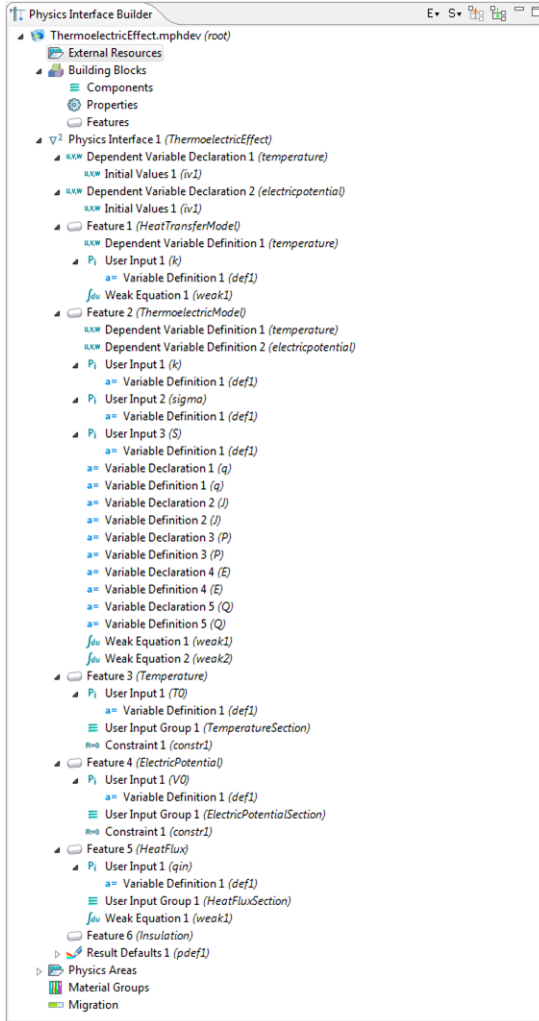
- A heat flux

$$q_{in}$$

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 Physics Interface—Adding 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 Start COMSOL Multiphysics.
- 2 From the **Options** menu, 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>Physics Builder**. A **Physics Builder** window then replaces the **Model Builder** window on the COMSOL Desktop.
- 4 Right-click the root node (**Untitled.mphdev**) and select **Physics Interface**. This adds a **Physics Interface** node (). Move to the settings window for the **Physics Interface 1** node.
- 5 In the **Identifiers** section, type **ThermoelectricEffect** in the **Type** edit field, **tee** in the **Default identifier and tag** edit field, and **Thermoelectric Effect** in the **Description** edit field. If you have a custom icon for the interface, you can 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 leave the **Allowed space dimensions** list with the default contents, which includes all space dimensions except 0D. 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 allowed.
- 7 In the **Settings** section, verify that **Domain** is selected in the **Top domain 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**).
- 8 It is good practice to save the physics interface after completing some steps. From the **File** menu, choose **Save** and create a physics interface file, **ThermoelectricEffect.mphdev** 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 Right-click the **Physics Interface** node and select **Variables>Dependent Variable Declaration** (.
- 2 In the **Dependent Variable Declaration** node's settings window, locate the **Declaration** section.

- 3 Leave the setting in the **Dependent variable ID** list as **Use physical quantity**. Type T in the **Default variable name** edit field. Type Temperature in the **Description** edit field. Type T in the **Symbol (LaTeX encoded)** edit field. Leave the **Size** list's default setting, **Scalar**, because T is a scalar field. Select **Temperature [K]** from the **Physical quantity** list.
- 4 In the **Preferences** section, leave the default settings that makes the dependent variable available for plotting (**Plot**) and for use as an input in other physics interfaces (**Announce variable to feature inputs**).
- 5 Once again, right-click the **Physics Interface** node and select **Variables>Dependent Variable Declaration (u,v,w)**.
- 6 In the **Dependent Variable Declaration** settings window, locate the **Declaration** section.
- 7 Leave the setting in the **Dependent variable ID** list as **Use physical quantity**. Type V in the **Default variable name** edit field. Type Electric Potential in the **Description** edit field. Type V in the **Symbol (LaTeX encoded)** edit field. Leave the **Size** list's default setting, **Scalar**, because V is a scalar field. Select **Electric Potential [V]** from the **Physical quantity** list.
- 8 In the **Preferences** section, leave the default settings.

ADDING THE HEAT TRANSFER MODEL

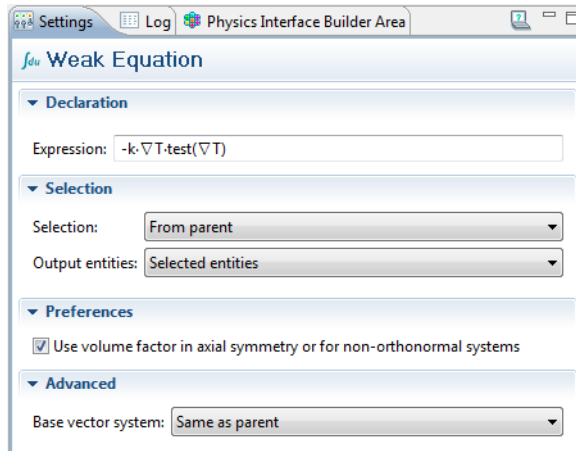
Next add the Heat Transfer Model as a domain feature:

- 1 Right-click the **Physics Interface** node and select **Feature** ()
- 2 In the **Feature 1** settings window, locate the **Identifiers** section.
- 3 Type HeatTransferModel in the **Type** edit field. Type htm in the **Default identifier and tag** edit field. Type Heat Transfer Model in the **Description** edit field.
- 4 In the **Restrictions** section, leave 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 interface.
- 5 In the **Settings** section, select the default **One level below top level (Boundary condition)** geometry level in the **Supported geometric entity level (relative to top geometric entity level)** list and click the **Delete** button () underneath the list. Then click the **Add** button () and select **Same as top level (Domain condition)** from the **Supported domain levels (relative to top domain level)** list; then click **OK**.
- 6 Leave **Active** in the **Allowed domain types** list.

- 7 Leave the **Category for overwriting selections** at the default setting, **Exclusive**.
- 8 From the **Material type** list, select **Solid**.
- 9 Leave the rest of the settings in the **Feature** settings window at their default values.
- 10 Right-click the **Feature 1 (HeatTransferModel)** node and select **Variables>Dependent Variable Definition** (P_1). In the settings window, locate the **Definition** section.
- 11 Select **Temperature [K]** from the **Physical quantity** list.
- 12 Leave the other settings at their default values.
- 13 Right-click the **Feature 1 (HeatTransferModel)** node and select **Inputs>User Input** (P_1). In the settings window, locate the **Declaration** section.
- 14 Type k in the **Input name** edit field and Thermal conductivity in the **Description** edit field. Type k in the **Symbol (LaTeX encoded)** edit field. Select **Thermal conductivity [W/(m*K)]** from the **Physical quantity** list.
- 15 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type 1.6 (typical value for Bismuth telluride) in the **Default value** edit field.
- 16 Leave all other default values.
- 17 Right-click the **User Input** node and select **Variable Definition** ($a=$). This user input then becomes available as a variable $tee.k$ in, for example, the predefined expressions for results evaluation. You can also refer to k directly in the weak equation.
- 18 Leave all other settings at their default values.
- 19 Right-click the **Feature 1 (HeatTransferModel)** node and select **Weak Equation** ($\int du$). This adds a **Weak Equation** node where you specify an equation for the domains in the physics interface. In the settings window, locate the **Declaration** section.

- 20 In the **Expression** edit field, type
 $-k \cdot \nabla T \cdot \text{test}(\nabla T)$

This expression implements the heat equation formulation for this interface. See [Entering Names and Expressions](#) for the keyboard entries to create the del operator (∇) and the dot product (\cdot).



- 21 Leave all other settings at their default values.

ADDING THE THERMOELECTRIC MODEL

Next add the Thermoelectric Model as a domain feature:

- 1 Right-click the **Physics Interface** node and select **Feature** ().
- 2 In the **Feature 2** settings window, locate the **Identifiers** section.
- 3 Type `ThermoelectricModel` in the **Type** edit field. Type `tem` in the **Default identifier and tag** edit field. Type `Thermoelectric Model` in the **Description** edit field.
- 4 In the **Settings** section, select the default **One level below top level (Boundary condition)** geometry level in the **Supported geometric entity level (relative to top geometric entity level)** list and click the **Delete** button () underneath the list. Then click the **Add** button () and select **Same as top level (Domain condition)** from the **Supported domain levels (relative to top domain level)** list; then click **OK**.
- 5 Leave **Active** in the **Allowed domain types** list.
- 6 Leave the **Category for overwriting selections** at the default setting, **Exclusive**.
- 7 From the **Material type** list, select **Solid**.

- 8 In the **Preferences** section, select the **Add as default feature** check box. Keep the settings in the **Default domain level** and **Default domain type** lists to make this a default physics model in all domains.
- 9 Leave the rest of the settings in the **Feature** settings window at their default values.
- 10 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Dependent Variable Definition(P_i)**. In the settings window, locate the **Definition** section.
- 11 Select **Temperature [K]** from the **Physical quantity** list.
- 12 Leave the other settings at their default values.
- 13 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Dependent Variable Definition(P_i)**. In the settings window, locate the **Definition** section.
- 14 Select **Electric Potential [V]** from the **Physical quantity** list.
- 15 Leave the other settings at their default values.

ADDING USER INPUTS

- 1 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Inputs>User Input (P_i)**. In the settings window, locate the **Declaration** section.
- 2 Type k in the **Input name** edit field and **Thermal conductivity** in the **Description** edit field. Type k in the **Symbol (LaTeX encoded)** edit field. Select **Thermal conductivity [W/(m*K)]** from the **Physical quantity** list.
- 3 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type 1.6 (typical value for Bismuth telluride) in the **Default value** edit field.
- 4 Leave all other default values.
- 5 Right-click the **User Input** node and select **Variable Definition ($a=$)**. This user input then becomes available as a variable $tee.k$ in, for example, the predefined expressions for results evaluation. You can also refer to k directly in the weak equation.
- 6 Leave all other settings at their default values.
- 7 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Inputs>User Input (P_i)**. In the settings window, locate the **Declaration** section.
- 8 Type σ in the **Input name** edit field and **Electric conductivity** in the **Description** edit field. Type σ in the **Symbol (LaTeX encoded)** edit field. Select **Electric conductivity [S/m]** from the **Physical quantity** list.
- 9 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar

number. Type $1.1e5$ (typical value for Bismuth telluride) in the **Default value** edit field.

- 10 Leave all other default values.
- 11 Right-click the **User Input** node and select **Variable Definition (a=)**. This user input then becomes available as a variable `tee.sigma` in, for example, the predefined expressions for results evaluation. You can also refer to `sigma` directly in the weak equation.
- 12 Leave all other settings at their default values.
- 13 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Inputs>User Input (P_i)**. In the settings window, locate the **Declaration** section.
- 14 Type `S` in the **Input name** edit field and `Seebeck coefficient` in the **Description** edit field. Type `S` in the **Symbol (LaTeX encoded)** edit field. Select **None** from the **Physical quantity** list and enter `V/K` as the SI unit. This is needed since the Seebeck coefficient is not a built-in material property.
- 15 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type $200e-6$ (typical value for p-type Bismuth telluride) in the **Default value** edit field.
- 16 Leave all other default values.
- 17 Right-click the **User Input** node and select **Variable Definition (a=)**. This user input then becomes available as a variable `tee.S` in, for example, the predefined expressions for results evaluation. You can also refer to `S` directly in the weak equation.
- 18 Leave all other settings at their default values.

ADDING VARIABLES

- 1 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Declaration (P_i)**. In the settings window, locate the **Declaration** section.
- 2 Type `q` in the **Variable name** edit field and `Heat flux` in the **Description** edit field. Type `q` in the **Symbol (LaTeX encoded)**. As **Size**, select **Vector (3x1)**. Select **Inward heat flux [A/m²]** from the **Physical quantity** list.
- 3 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Definition (P_i)**. In the settings window, locate the **Definition** section.
- 4 In the **Expression** edit field, type
$$P \cdot J - k \cdot \nabla T$$

- 5 Leave all other settings at their default values.
- 6 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Declaration (P_i)**. In the settings window, locate the **Declaration** section.
- 7 Type J in the **Variable name** edit field and **Current density** in the **Description** edit field. Type J in the **Symbol (LaTeX encoded)**. As **Size**, select **Vector (3x1)**. Select **Current density [A/m²]** from the **Physical quantity** list.
- 8 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Definition (P_i)**. In the settings window, locate the **Definition** section.
- 9 In the **Expression** edit field, type

$$-\text{sigma}*(\nabla V+S*\nabla T)$$
- 10 Leave all other settings at their default values.
- 11 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Declaration (P_i)**. In the settings window, locate the **Declaration** section.
- 12 Type P in the **Variable name** edit field and **Peltier coefficient** in the **Description** edit field. Type P in the **Symbol (LaTeX encoded)**. As **Size**, select **Scalar**. Select **Electric Potential [V]** from the **Physical quantity** list.
- 13 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Definition (P_i)**. In the settings window, locate the **Definition** section.
- 14 In the **Expression** edit field, type

$$S*T$$
- 15 Leave all other settings at their default values.
- 16 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Declaration (P_i)**. In the settings window, locate the **Declaration** section.
- 17 Type E in the **Variable name** edit field and **Electric field** in the **Description** edit field. Type E in the **Symbol (LaTeX encoded)**. As **Size**, select **Vector (3x1)**. Select **Electric field [V/m]** from the **Physical quantity** list.
- 18 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Definition (P_i)**. In the settings window, locate the **Definition** section.
- 19 In the **Expression** edit field, type

$$-\nabla V$$
- 20 Leave all other settings at their default values.
- 21 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Declaration (P_i)**. In the settings window, locate the **Declaration** section.

- 22 Type Q in the **Variable name** edit field and Joule heating in the **Description** edit field. Type Q in the **Symbol (LaTeX encoded)**. As **Size**, select **Scalar**. Select **Heat source $[W/m^3]$** from the **Physical quantity** list.
- 23 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Variables>Variable Definition (P_1)**. In the settings window, locate the **Definition** section.
- 24 In the **Expression** edit field, type $J \cdot E$
- 25 Leave all other settings at their default values.


ADDING THE WEAK EQUATIONS

- 1 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Weak Equation ($\int du$)**. In the settings window, locate the **Declaration** section.
- 2 In the **Expression** edit field, type $q \cdot \text{test}(\nabla T) + Q \cdot \text{test}(T)$
- 3 Right-click the **Feature 2 (ThermoelectricModel)** node and select **Weak Equation ($\int du$)**. In the settings window, locate the **Declaration** section.
- 4 In the **Expression** edit field, type $-J \cdot \text{test}(E)$
Note that here $-E$ 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.

The Temperature Boundary Condition

- 1 Right-click the **Physics Interface** node and select **Feature ()**.
- 2 In the **Feature 3** node's settings window, locate the **Identifiers** section.
- 3 Type Temperature in the **Type** edit field. Type tp in the **Default identifier and tag** edit field. Type Temperature in the **Description** edit field.
- 4 In the **Restrictions** section, leave the default setting, **Same as parent**, for the **Allowed space dimensions** and **Allow study types** lists.
- 5 In the **Settings** section, keep the default **One level below top level (Boundary condition)** type in the **Supported domain levels (relative to top domain level)** list. Leave **Exterior**, **Interior**, and **Pair** in the **Allowed domain types** list. This makes the boundary condition available for all those boundary types.
- 6 Change the **Category for overwriting selections** to **Contributing**.





- 7 Leave the settings in the **Input base vector system** and **Base vector system** lists to the default setting: **Frame system compatible with material type**.
- 1 Right-click the **Feature 3 (Temperature)** node and select **Inputs>User Input (P_i)**. In the settings window, locate the **Declaration** section.
- 2 Type T0 in the **Input name** edit field and Temperature in the **Description** edit field. Type T in the **Symbol (LaTeX encoded)** edit field. Select **Temperature [K]** from the **Physical quantity** list.
- 3 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type 293.15 in the **Default value** edit field, corresponding to room temperature (in K).
- 4 Leave all other default values.
- 5 Right-click the **User Input** node and select **Variable Definition ($a=$)**. This user input then becomes available as a variable `tee.T0` in, for example, the predefined expressions for results evaluation. You can also refer to T0 directly in the weak equation.
- 6 Leave all other settings at their default values.
Now define a Section in the settings window for the boundary condition:
- 7 Right-click the **Feature 3 (Temperature)** node and select **Inputs>User Input Group(P_i)**. In the settings window, locate the **Declaration** section.
- 8 Type TemperatureSection in the **Group name** edit field. Type Temperature in the **Description** edit field.
- 9 Add **User Input1 (T0)** to the **Group members** list. Use the + button to get a selection list of available user inputs.
- 10 Locate the **GUI Options** section. Change the **GUI layout** to **Group members define a section**.
Now define the constraint equation:
- 11 Right-click the **Feature 3 (Temperature)** node and select **Constraint(P_i)**. In the settings window, locate the **Declaration** section.
- 12 In the **Expression** edit field, type
T0-T
- 13 Locate the **Shape Declaration** section and select **Temperature [K]** from the **Physical quantity** list.
- 14 Leave all other default values.

The Electric Potential Boundary Condition

- 1 Right-click the **Physics Interface** node and select **Feature** ().
- 2 In the **Feature 4** settings window, locate the **Identifiers** section.
- 3 Type `ElectricPotential` in the **Type** edit field. Type `ep` in the **Default identifier and tag** edit field. Type `Electric Potential` in the **Description** edit field.
- 4 Change the **Category for overwriting selections** to **Contributing**.
- 5 Leave all other default values.
- 6 Right-click the **Feature 4 (ElectricPotential)** node and select **Inputs>User Input** (). In the settings window, locate the **Declaration** section.
- 7 Type `V0` in the **Input name** edit field and `Electric Potential` in the **Description** edit field. Type `V` in the **Symbol (LaTeX encoded)** edit field. Select **Electric Potential [V]** from the **Physical quantity** list.
- 8 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type `0` in the **Default value** edit field.
- 9 Leave all other default values.
- 10 Right-click the **User Input** node and select **Variable Definition** (). This user input then becomes available as a variable `tee.V0` in, for example, the predefined expressions for results evaluation. You can also refer to `V0` directly in the weak equation.
- 11 Leave all other settings at their default values.
Now define a Section in the settings window for the boundary condition:
- 12 Right-click the **Feature 4 (ElectricPotential)** node and select **Inputs>User Input Group** (). In the settings window, locate the **Declaration** section.
- 13 Type `ElectricPotentialSection` in the **Group name** edit field. Type `Electric Potential` in the **Description** edit field.
- 14 Add **User Input1 (V0)** to the **Group members** list. Use the + button to get a selection list of available user inputs.
- 15 Locate the **GUI Options** section. Change the **GUI layout** to **Group members define a section**.
Now define the constraint equation.
- 16 Right-click the **Feature 4 (ElectricPotential)** node and select **Constraint** (). In the settings window, locate the **Declaration** section.


- 17 In the **Expression** edit field, type $V_0 - V$
- 18 Locate the **Shape Declaration** section and select **Electric Potential [V]** from the **Physical quantity** list.
- 19 Leave all other default values.

The Heat Flux Boundary Condition


- 1 Right-click the **Physics Interface** node and select **Feature** ().
- 2 In the **Feature 5** settings window, locate the **Identifiers** section.
- 3 Type HeatFlux in the **Type** edit field. Type hf in the **Default identifier and tag** edit field. Type Heat Flux in the **Description** edit field.
- 4 Change the **Category for overwriting selections** to **Contributing**.
- 5 Leave all other settings at their default values.
- 1 Right-click the **Feature 5 (HeatFlux)** node and select **Inputs>User Input** (). In the settings window, locate the **Declaration** section.
- 2 Type q_{in} in the **Input name** edit field and Normal Heat Flux in the **Description** edit field. Type q_{in} in the **Symbol (LaTeX encoded)** edit field. Select **Inward heat flux [W/m^2]** from the **Physical quantity** list.
- 3 Keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type 0 in the **Default value** edit field.
- 4 Leave all other default values.
- 5 Right-click the **User Input** node and select **Variable Definition** (). This user input then becomes available as a variable `tee.qin` in, for example, the predefined expressions for results evaluation. You can also refer to q_{in} directly in the weak equation.
- 6 Leave all other settings at their default values.
Now define a Section in the settings window for the boundary condition:
- 7 Right-click the **Feature 5 (HeatFlux)** node and select **Inputs>User Input Group** (). In the settings window, locate the **Declaration** section.
- 8 Type HeatFluxSection in the **Group name** edit field. Type Heat Flux in the **Description** edit field.
- 9 Add **User Input1 (q_{in})** to the **Group members** list. Use the + button to get a selection list of available user inputs.

- 10 Locate the **GUI Options** section. Change the **GUI layout** to **Group members define a section**.

Now define the weak equation:










- 11 Right-click the **Feature 5 (HeatFlux)** node and select **Weak Equation** (). In the settings window, locate the **Declaration** section.
- 12 In the **Expression** edit field, type $q_{in} \cdot \text{test}(T)$
For a theoretical explanation of this expression, see the earlier theory section.
- 13 Leave all other default values.

The Insulation Condition

- 1 Right-click the **Physics Interface** node and select **Feature** ().
- 2 In the **Feature 6** settings window, locate the **Identifiers** section.
- 3 Enter *Insulation* in the **Type** edit field. Enter *in* in the **Default identifier and tag** edit field. Enter *Insulation* in the **Description** edit field.
- 4 This time, keep the **Category for overwriting selections** as **Exclusive**.
- 5 In the **Preferences** section, select the **Add as default feature** check box. Keep the settings in the **Default domain level** and **Default domain type** lists to make this a default boundary condition on exterior boundaries only.
- 6 Leave all other settings at their default values.

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 **Physics Interface 1 (ThermoelectricEffect)**, select **Results Defaults** ().
- 2 Right-click **Results Defaults** () and select **Plot Group 3D** ().
- 3 Right-click the **Plot Group 3D** () node and select **Surface** ().
- 4 In the **Surface** () settings window, type *T* in the **Expression** edit field. Type *K* in the **Unit** edit field.
- 5 Leave all other settings at their default values.
- 6 Right-click the **Surface** () node and select **Rename**. In the **Rename Surface** dialog box, type *Temperature* in the **New name** edit field. Click **OK** to confirm.
- 7 Right-click **Results Defaults 1** () and select **Plot Defaults** ().

- 8 Right-click the **Plot Defaults** (🖌️) node and select **Default Scalar Plot** (🖌️). In the **Expression** edit field, type T. In the Description edit field, type Temperature. This makes temperature the default for all scalar plots.

MAKING A THERMOELECTRIC DEVICES PHYSICS AREA

To add the Thermoelectric Effect physics interface to a new physics area for Thermoelectric Devices under the Heat Transfer branch, do the following steps:



- 1 Right-click the **Physics Areas** node (📁) and select **Physics Area** (🌐).
- 2 In the **Parent Area** window for the **Physics Area** node, select the **Heat Transfer** folder.
- 3 In the **Physics Area Settings** section, enter ThermoelectricDevices in the **Name** edit field, Thermoelectric Devices in the **Description** edit field. The default icon is physics.png (🔧), which is appropriate for this physics interface. Otherwise, click **Browse** to use another icon. Enter 10 in the Weight edit field 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). This completes the definition of this Thermoelectric Effect physics interface. Save the file that contains the physics interface as ThermoelectricEffect.mphdev.

TESTING THE PHYSICS 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, **Physics Interface 1**) 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 new physics interface appears in the Model Wizard and that the functionality and settings appear as expected.

- 1 From the **View** menu, choose **Physics Builder Area**.
- 2 Under **Builder Files**, right-click the **Development Files** node (📁) and select **Add Builder File**. Browse to locate ThermoelectricEffect.mphdev, select it, and click **Open**.
- 3 Click the **New Model** button (📄) on the main toolbar.
- 4 Select any space dimension. In the **Model Wizard's Add Physics** page, select **Heat Transfer>Thermoelectric Devices>Thermoelectric Effect (tee)**.

- 5 Click **Next** ().
- 6 On the **Study Type** page, verify that **Stationary** is the only available study type under **Preset Studies**. Select it and click **Finish** ().
- 7 In the Model Builder, verify that the default nodes appear as expected and that their settings windows contain the user inputs that you specified.
- 8 Proceed by building an example model (see the example model below) 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.
- 9 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 interface definitions to see how it behaves after applying some extensions or corrections. The Thermoelectric Effect physics 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:

$$\begin{aligned} \text{Joule heating} &= \sigma\nabla V \cdot \nabla V = \mathbf{J} \cdot \mathbf{E} \\ \text{Thomson heating} &= \sigma S\nabla T \cdot \nabla V = \mathbf{J} \cdot S\nabla T \end{aligned}$$

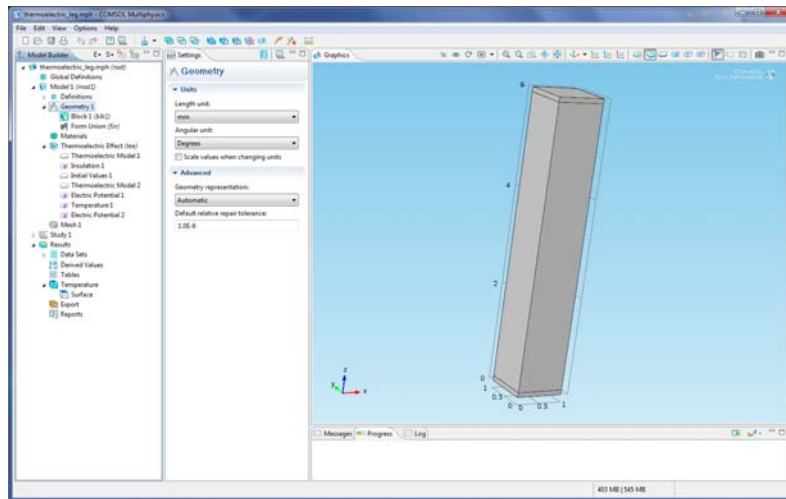
Example Model—Thermoelectric Leg

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](#):

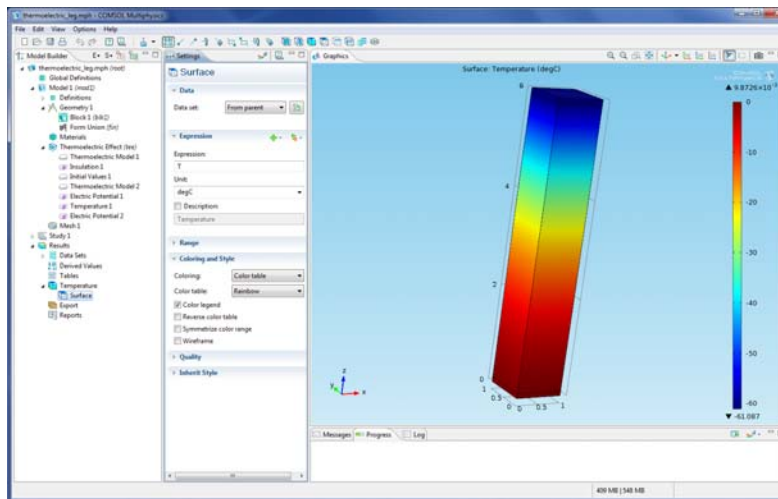
PROPERTY	SYMBOL AND UNIT	BISMUTH TELLURIDE	COPPER
Thermal conductivity	k [W/ (m*K)]	1.6	350

PROPERTY	SYMBOL AND UNIT	BISMUTH TELLURIDE	COPPER
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 that 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 physics interface.

MODEL WIZARD

- 1 Start COMSOL Multiphysics.
- 2 In the **Model Wizard**, click the **3D** button on the **Select Space Dimension** page. Then click the **Next** button ().
- 3 On the **Add Physics** page, select **Heat Transfer>Thermoelectric Devices>Thermoelectric Effect**. Click the **Next** button ().
- 4 On the **Select Study Type** page, select **Preset Studies>Stationary**. Click the **Finish** button ().

GEOMETRY MODELING

- 1 In the settings window of the **Geometry** node, change the **Length unit** to mm.
- 2 Add a block of width 1 mm, depth 1 mm, and height 6 mm.
- 3 Still in the **Block Settings** page, locate the **Layers section** and add two layers: **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.
- 2 For this second Thermoelectric Model, select (left-click right click) the copper electrodes: domains 1 and 3.
- 3 For the **Thermal conductivity** enter 350.
- 4 For the **Electric conductivity** enter 5.9×10^8 .
- 5 For the **Seebeck coefficient** enter 6.5×10^{-6} .
- 6 The default **Thermoelectric Model** already has the correct values for Bismuth telluride and is now assigned to domain 2.
- 7 Right-click the **Thermoelectric Effect** node and select Electric Potential.
- 8 Select boundary 3 (bottom surface) and keep the default 0 V.
- 9 Right-click the **Thermoelectric Effect** node and select Temperature.
- 10 Select boundary 3 again and set the temperature to 273.15 K (0 degrees C).
- 11 Right-click the **Thermoelectric Effect** node and select Electric Potential.
- 12 Select boundary 10 (top surface) and set the electric potential to 0.05 V.

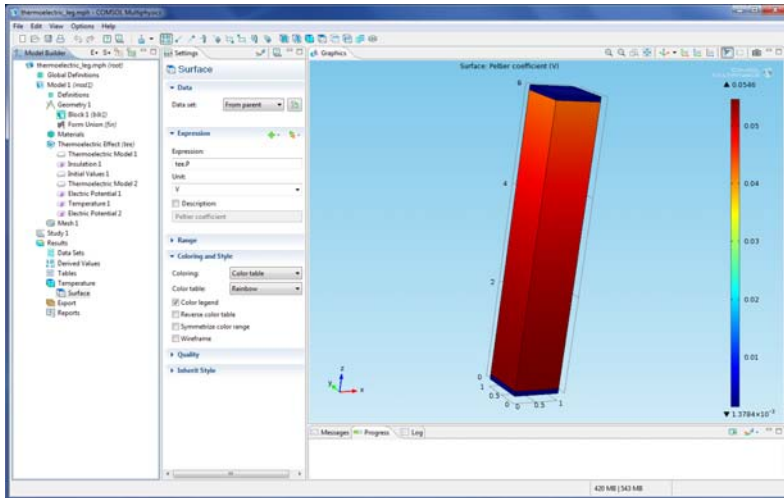
MESH GENERATION AND COMPUTING THE SOLUTION

- 1 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.
- 2 Right-click the **Study** node and select **Compute**.

RESULTS AND VISUALIZATION

The default plot is for the temperature T . Change the unit to degC to verify the 61 degree C temperature drop.

Also try visualizing some of the other predefined expressions such as the **Electric potential V** and **Peltier coefficient P** (see the following figure).



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:

$$\begin{aligned} E\Psi &= \hat{H}\Psi \\ i\frac{\hbar}{2\pi}\frac{\partial\Psi}{\partial t} &= \hat{H}\Psi \end{aligned}$$

where Ψ is the quantum mechanical wave function (the probability amplitude for different configurations of the system) and \hat{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](#)
- [Schrödinger Equation Interface—Adding 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 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.

NAME AND DESCRIPTION

The name of this interface is *Schrodinger Equation* (avoiding using the character “ö” in the user interface). The short name is `scheq`. There is also an identifier, `SchrodingerEq`, which is used by the Java and 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 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—Adding It Step by Step

The following steps show how to defined the Schrodinger Equation interface using the implementation defined in the previous section.

CREATING THE BASICS

- 1 Start COMSOL Multiphysics.
- 2 From the **Options** menu, 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>Physics Builder**. A **Physics Builder** window then replaces the **Model Builder** window on the COMSOL Desktop.
- 4 Right-click the root node (**Untitled.mphdev**) and select **Physics Interface**. This adds a **Physics Interface** node (). Move to the settings window for the **Physics Interface** node.
- 5 In the **Identifiers** section, type `SchrodingerEq` in the **Type** edit field, `scheq` in the **Default identifier and tag** edit field, and `Schrodinger Equation` in the **Description** edit field. If you have a custom icon for the interface, you can 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, so leave the **Allowed space dimensions** list with the default contents, which includes all space dimensions except 0D. In the **Allowed study types** list, select the default study types (**Stationary** and **Transient**) and click the **Delete** button () underneath the list. Then click the **Add** button () and select **Eigenvalue** from the **Allowed study types** list; then click **OK**.
- 7 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**).
- 8 It is good practice to save the physics interface after completing some steps. From the **File** menu, choose **Save** and create a physics interface file, `SchrodingerEquation.mphdev` 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 FEATURES



First declare the dependent variable Ψ :

- 1 Right-click the **Physics Interface** node and select **Variables>Dependent Variable Declaration (u,v,w)**.
- 2 In the **Dependent Variable Declaration** settings window, locate the **Declaration** section.
- 3 Leave the setting in the **Dependent variable ID** list as **Use physical quantity**. Type `psi` in the **Default variable name** edit field. Type `Wave function` in the **Description** edit field. Type `\psi` (the LaTeX syntax for the Greek letter ψ) in the **Symbol (LaTeX encoded)** edit field. Leave the **Size** list's default setting, **Scalar**, because Ψ is a scalar field. Select **Dimensionless [1]** from the **Physical quantity** list.
- 4 In the **Preferences** section, leave the default settings that makes the dependent variable available for plotting (**Plot**) and for use as an input in other physics interfaces (**Announce variable to feature inputs**).

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 24 below).

Next add the Schrödinger equation in a domain feature:

- 1 Right-click the **Physics Interface** node and select **Feature** ()
- 2 In the **Feature** settings window, locate the **Identifiers** section.

- 3 Type SchrodingerEqu in the **Type** edit field. Type schequ in the **Default identifier and tag** edit field. Type Schrodinger equation model in the **Description** edit field.
- 4 In the **Restrictions** section, leave 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 interface.
- 5 In the **Settings** section, select the default **One level below top level (Boundary condition)** geometry level in the **Supported geometric entity levels (relative to top geometric entity level)** list and click the **Delete** button () underneath the list. Then click the **Add** button () and select **Same as top level (Domain condition)** from the **Supported geometric entity levels (relative to top geometric entity level)** list; then click **OK**.
- 6 Leave **Active** in the **Allowed domain types** list.
- 7 Leave the **Category for overwriting selections** at the default setting, **Exclusive**.
- 8 From the **Material type** list, select **Selectable by user**.
- 9 Leave the settings in the **Input base vector system** and **Base vector system** lists to the default setting: **Frame system compatible with material type**.
- 10 In the **Preferences** section, select the **Add as default feature** check box to make this the default feature for all domains.
- 11 Right-click the **Feature 1 (SchrodingerEqu)** node and select **Inputs>User Input (P₁)**. In the settings window, locate the **Declaration** section.
- 12 Type μ in the **Input name** edit field and Reduced mass in the **Description** edit field. Type $\backslash\mu$ (LaTeX syntax for the Greek letter μ) in the **Symbol (LaTeX encoded)** edit field. Select **Mass [kg]** from the **Physical quantity** list.
- 13 In the **Settings** section, keep the default settings in the **Levels (Single)** and **Size (Scalar)** lists for a basic scalar quantity. Also leave the **Allowed values** setting to **Any** for an entry of a general scalar number. Type m_e_const (the electron mass, which is a built-in physical constant) in the **Default value** edit field.
- 14 Right-click the **User Input** node and select **Variable Definition (a=)**. 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 μ directly in the weak equation.
- 15 Leave all other settings at their default values.

- 16 Right-click the **Feature 1 (SchrodingerEqu)** node and select **Inputs>User Input (P_i)**. In the settings window, locate the **Declaration** section.
- 17 Type V in the **Input name** edit field and Potential energy in the **Description** edit field. Type V in the **Symbol (LaTeX encoded)** edit field. Select **Energy [J]** from the **Physical quantity** list.
- 18 In the **Settings** section, keep the default settings in the **Levels (Single)** and **Size (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** edit field.
- 19 Right-click the **User Input** node and select **Variable Definition ($a=$)**. 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.
- 20 Leave all other settings at their default values.
- 21 Right-click the **Feature 1 (SchrodingerEqu)** node and select **Weak Equation ($\int_{\Omega} du$)**. This adds the **Weak Equation** node where you specify the equation for the domains in the physics interface. In the settings window, locate the **Declaration** section.
- 22 In the **Expression** edit field, type

$$-\hbar_{\text{const}}^2 / (2 * \mu) * \nabla \psi \cdot \text{test}(\nabla \psi) - (V - \lambda) * \psi \cdot \text{test}(\psi)$$
 This expression implements the Schrodinger equation formulation for this interface. See [Entering Names and Expressions](#) for the keyboard entries to create the del operator (∇) and the dot product (\cdot).
- 23 In the **Selection** section, leave the **Selection** list setting (**From parent**) as is to inherit the selection from the top node. Also leave the **Output entities** list to its default setting (**Selected entities**) to use the selected domains as the output. Leave the remaining settings at their default values.
- 24 Right-click the **Feature 1 (SchrodingerEqu)** node and select **Variables>Dependent Variable Definition (u,v,w)**. In the settings window, locate the **Definition** section.
- 25 Select **Dimensionless [1]** from the **Physical quantity** list. Leave the default shape function (element type) as **Lagrange** in the **Shape function** list. Lagrange elements are the most widely used and are suitable for this physics interface. The default order becomes 2.


Add the default boundary condition:



- 1 Right-click the **Physics Interface** node and select **Feature ()**.

- 2 In the **Feature** settings window, locate the **Identifiers** section.
- 3 Type ZeroProb in the **Type** edit field. Type zpb in the **Default identifier and tag** edit field. Type Zero probability in the **Description** edit field.
- 4 In the **Restrictions** section, leave the default setting, **Same as parent**, for the **Allowed space dimensions** and **Allow study types** lists.
- 5 In the **Settings** section, keep the default **One level below top level (Boundary condition)** type in the **Supported geometric entity levels (relative to top geometric entity level)** list. Leave **Exterior**, **Interior**, and **Pair** in the **Allowed domain types** list. This makes the boundary condition available for all those boundary types.
- 6 Leave the **Category for overwriting selections** at the default setting, **Exclusive**.
- 7 Leave the settings in the **Input base vector system** and **Base vector system** lists to the default setting: **Frame system compatible with material type**.
- 8 In the **Preferences** section, select the **Add as default feature** check box. Keep the settings in the **Default domain level** and **Default domain type** lists to make this a default boundary condition on exterior boundaries only.
- 9 Right-click the **Feature 2 (ZeroProb)** node and select **Constraint (R=0)**. In the **Constraint** settings window, locate the **Declaration** section.
- 10 Type 0-psi in the **Expression** edit field to make $\Psi = 0$ on the boundary.
- 11 In the **Shape Declaration** section, select **Dimensionless [1]** from the **Physical quantity** list to declare the correct dimension for the constrained variable Ψ .



Notice that the equation model and the Zero Probability boundary condition now appear in the **Default Features** list in the **Physics Interface** settings window.

Add the Wave Function Value boundary condition $\Psi = \Psi_0$:

- 1 Right-click the **Physics Interface** node and select **Feature** ()
- 2 In the **Feature** settings window, locate the **Identifiers** section.
- 3 Type WaveFunc in the **Type** edit field. Type wvfcn in the **Default identifier and tag** edit field. Type Wave function value in the **Description** edit field.
- 4 In the **Restrictions** section, leave the default setting, **Same as parent**, for the **Allowed space dimensions** and **Allow study types** lists.
- 5 In the **Settings** section, keep the default **One level below top level (Boundary condition)** type in the **Supported geometric entity levels (relative to top geometric entity level)** list. Leave **Exterior**, **Interior**, and **Pair** in the **Allowed domain types** list. This makes the boundary condition available for all those boundary types.
- 6 Leave the **Category for overwriting selections** at the default setting, **Exclusive**.

- 7 Leave the settings in the **Input base vector system** and **Base vector system** lists to the default setting: **Frame system compatible with material type**.
- 8 Right-click the **Feature 3 (WaveFunc)** node and select **Inputs>User Input** (). In the **User Input** settings window, locate the **Declaration** section.
- 9 Type `psi0` in the **Input name** edit field and `Wave function value` in the **Description** edit field. Type `\psi_0` in the **Symbol (LaTeX encoded)** edit field for the symbol Ψ_0 . Select **Dimensionless [1]** from the **Physical quantity** list. Use the default values in the **Settings** section, which provide a scalar quantity with the default value 0.
- 10 Right-click the **Feature 3 (WaveFunc)** node and select **Constraint** (). In the **Constraint** settings window, locate the **Declaration** section.
- 11 Type `par.psi0-psi` in the **Expression** edit 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.
- 12 In the **Shape Declaration** section, select **Dimensionless [1]** from the **Physical quantity** list to declare the correct dimension for the constrained variable Ψ .


This feature uses an 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, so it is necessary to specify a section for the user input.


- 13 Right-click the **Feature 3 (WaveFunc)** node and select **Inputs>User input group** (). In the **User Input Group** settings window, locate the **Declaration** section.
- 14 Type `WaveFunc_section` in the **Group name** edit field and `Wave function` in the **Description** edit field.
- 15 Beneath the **Group members** list, click the **Add** button () and select **User input 1 (psi0)** from the **Group members** list; then click **OK**.
- 16 In the **GUI Options** section, select **Group members define a section** in the **GUI Layout** list.

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 **Physics Interface** node and select **Variable>Variable Declaration** ().
- 2 In the **Variable Declaration** settings window, locate the **Declaration** section.

- 3 Type **probdens** in the **Variable name** edit field, type Probability density function in the **Description** edit field, and type $\{\mid\psi\mid\}^2$ in the **Symbol (LaTeX encoded)** edit field.
- 4 Select **Dimensionless [I]** from the **Physical quantity** list.
- 5 Leave the settings in the **Preferences** section at their default values. The **Show in plot menu** check box must be selected for this variable to appear as a predefined expression in plots.
- 6 Right-click the **Variable Declaration** node and select **Variables>Variable Definition** ().
- 7 In the **Variable Definition** settings window, locate the **Definition** section.
- 8 Type $\text{abs}(\psi)^2$ in the **Expression** edit field.
Notice that the variable **probdens** now appears in the **Variable Declarations** list in the **Physics Interface** settings window.




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 **Physics Interface** node and select **Results Defaults** ().
- 2 Right-click **Results Defaults** () and select **Plot Group 2D** ().
- 3 Right-click the **Plot Group 2D** () node and select **Surface** ().
- 4 In the **Surface** () settings window, type **probdens** in the **Expression** edit field.
- 5 Right-click the **Surface** () node and select **Rename**. In the **Rename Surface** dialog box, type Probability density in the **New name** edit field. Click **OK**.
- 6 Right-click **Results Defaults** () and select **Plot Defaults** ().
- 7 Right-click the **Plot Defaults** () node and select **Default Scalar Plot** (). In the **Expression** edit field, type **probdens**. This makes the probability density function the default for all scalar plots.

MAKING A QUANTUM MECHANICS PHYSICS AREA


To add the Schrodinger Equation to a new physics area for Quantum Mechanics under the Mathematics branch, do the following steps:

- 1 Right-click the **Physics Areas** node () and select **Physics Area** ().
- 2 In the settings window for the **Physics Area** node, enter QuantumMechanics in the **Name** edit field, Quantum mechanics in the **Description** edit field. The default icon is **physics.png** (), which is appropriate for this physics area. Otherwise, click





Browse to use another icon. Enter 10 in the **Weight** edit field 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 areas). This completes the definition of this Schrodinger Equation interface. Save the file that contains the physics interface, `SchrodingerEquation.mphdev`.

- 3 Locate the **Physics Area** section, and select the **Mathematics** node. Beneath the tree of physics areas, click the **Set As Parent** button (). This moves the new physics area to the **Mathematics** node.
- 4 Select the **Physics Interface** node.
- 5 Locate the **Physics Area** section, expand it, and select the **Mathematics>Quantum mechanics** node. Beneath the tree of physics areas, click the **Set As Parent** button (). This places the Schrodinger Equation interface under the Quantum mechanics physics area

TESTING THE PHYSICS INTERFACE

At any time during the implementation of a physics interface using the Physics Builder, you can launch an updated preview of the 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, **Physics Interface 1**) 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 **View** menu, choose **Physics Builder Manager**.
- 2 Under **Builder Files**, right-click the **Development Files** node () and select **Add Builder File**. Browse to locate `SchrodingerEquation.mphdev`, select it, and click **Open**.
- 3 Click the **New** button () on the main toolbar.
- 4 Select any space dimension. In the **Model Wizard's Add Physics** page, select **Mathematics>Quantum Mechanics>Schrodinger Equation (scheq)**.
- 5 Click **Next** ().
- 6 On the **Study Type** page, verify that **Eigenvalue** is the only available study type under **Preset Studies**. Select it and click **Finish** ().

- 7 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.
- 8 Proceed by building an example model (see the Hydrogen Atom model below) 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.
- 9 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 interface 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 gives 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

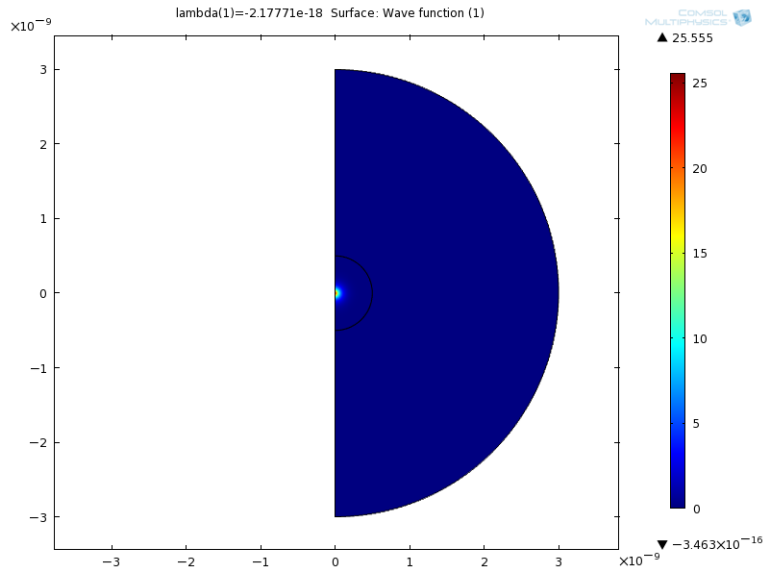


Figure 4-1: The wave function ψ for the first energy eigenvalue.

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 $(2,0,0)$, $(2,1,0)$ and $(3,0,0)$, $(3,1,0)$, $(3,2,0)$. 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 Start COMSOL Multiphysics.
- 2 In the **Model Wizard**, click the **2D axisymmetric** button on the **Select Space Dimension** page. Then click the **Next** button (➡).
- 3 On the **Add Physics** page, select **Mathematics>Quantum Mechanics>Schrodinger Equation (scheq)**. Click the **Next** button (➡).
- 4 On the **Select Study Type** page, select **Preset Studies>Eigenvalue**. Click the **Finish** button (🏁).

GEOMETRY MODELING

- 1 Add a semicircle centered at the origin (0, 0) with a radius of $3 \cdot 10^{-9}$ m (3 nm). Use a sector angle of 180 degrees and a rotation of -90 degrees to create a semicircle in the right half-plane.
- 2 Add another semicircle centered at the origin (0, 0) with a radius of $0.5 \cdot 10^{-9}$ m (0.5 nm). Use the same sector angle and rotation as for the first semicircle.
- 3 Build the geometry by pressing F8, for example. The final geometry consists of two semicircles in the $r > 0$ half-plane.

PHYSICS SETTINGS

- 1 In the **Schrodinger Equation Model** node, the default in the **Reduced mass** edit field is the electron mass, m_e . Type the following expression into the **Potential energy** edit field:
$$-e_const^2 / (4 \cdot \pi \cdot \epsilon_{0_const} \cdot \sqrt{r^2 + z^2})$$
where e_const and ϵ_{0_const} are built-in physical constants: 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](#).
- 2 Verify that the default boundary conditions are correct: the Axial Symmetry condition applies to the symmetry boundaries at $r = 0$, and the Zero Probability condition 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 an extra **Size** node for defining the mesh size in the inner circular domain (Domain 2). In the **Size** settings window, select **Domain** from the **Geometric entity level** list and then add Domain 2 to the **Selection** list.
- 2 In the **Element Size** section, click the **Custom** button.
- 3 In the **Element Size Parameters** section, select the **Maximum element size** check box and type $0.05e-9$ in the corresponding edit field to use a mesh size no larger than 0.05 nm in Domain 2.
- 4 Right-click the **Mesh** node and select **Free Triangular** to add a node that meshes Domain 2 using the specified mesh size.
- 5 Select the top **Size** node, and in its settings window change the value in the **Maximum element growth rate** edit field to 1.1 to make the mesh size grow more slowly toward the perimeter of the geometry.
- 6 Press F8 or right-click the **Mesh** node and select **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 Select **Study 1 > Step 1: Eigenvalue**, and in its settings window, locate the top **Study Settings** section.
- 2 Enter 10 in the **Desired number of eigenvalues** edit field and $-2e-18$ in the **Search for eigenvalues around** edit field.
- 3 Right-click the **Study 1** node and select **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 **2D Plot Group 1**, you can select which eigenmode to plot by selecting the corresponding eigenvalue from the **Eigenvalue** list. You can also verify that the eigenvalues correspond to the first energy eigenvalues listed in [Results](#).

In the settings window for the **Surface** plot, you can click the **Replace Expression** button () and select **Schrodinger Equation>Wave function** to plot the variable ψ_i , which is the complex-valued wave function for the electron position.

I n d e x

- A** allowed values, for inputs 44
announcing variables 55
- B** backward compatibility 130
Bohr radius 183
Boolean inputs 44
builder archive, compiling 127
Building Blocks 88
built-in physics interfaces 24
- C** cards 118
cards, for GUI layout 118
code editor 89
compiling builder archive 127
compiling builder files 127
component links 83
component settings 62
components 88
 creating 82
components, for GUI design 115
conditions, adding 84
constraints 72
contained features 39
contravariant 113
contributing boundary condition 150
contributing features 33
covariant 113
cross product 106
- D** del operator 148
documentation, finding 12
dot product 106, 148, 172
double dot product 107
double-level inputs 117
- E** Eclipse Help plug-in, for documentation 136
Einstein summation notation 113
electron charge 172
electron mass 172
elements, creating 123
emailing COMSOL 13
equations
 referencing 86
equations, displaying 86
exclusive boundary condition 150
exclusive feature 33
exclusive features 33
external resources 90
- F** feature input, rename 133
feature inputs 48, 55
feature links 37
features 88
files, importing 90
Frobenius inner product 107
functions 81
- G** GUI layout 115
- H** Hamiltonian 170
HTML format
 for documentation 136
hydrogen atom example 182
- I** importing files 90
initial values 61
input groups 118
Internet resources 11
- J** Joule heating 142
- K** knowledge base, COMSOL 13
- L** LaTeX encoding 54, 59, 86
low-level elements 123
- M** material list, rename 133
material parameter, rename 132
material properties 49, 93

- matrix symmetry, options for 116
 - mesh generation 104
 - mesh sizes, predefined 104
 - migration 130
 - model inputs 34
 - Model Library 12
 - Model Object API 130
 - Model Wizard 91
 - MPH-files 12
 - multiphysics features 38
- N** nabla operator 106, 172
- O** one-particle system 170, 182
- operators 80
- P** Peltier effect 142
- permittivity of vacuum 172
 - physical quantities 95
 - physics areas 91
 - Physics Builder Manager window 127
 - Physics Builder window 21
 - Planck constant 170
 - plot groups, creating 102
 - preview
 - of documents 136
 - preview, of the physics interface 164, 180
 - probability density function 184
 - for electron's position 173, 182
 - properties 21, 41
 - property links 42
- Q** quantum numbers 183
- R** radio buttons 120
- reduced mass 170
 - reduced Planck constant 170
 - referencing equations 86
 - renaming inputs, migration branch 132
 - results defaults 102
- S** scalar multiplication 172
- scalar product 148, 172
 - Schrödinger equation 170
 - scope, for variables 108
 - Seebeck effect 142
 - selections 83
 - selections, specifying 63
 - shape functions 54, 60
 - singleton features 34
 - solver defaults 99
 - sublayouts 118
- T** technical support, COMSOL 13
- tensor operations 107
 - tensor parser 106
 - testing a physics interface 127
 - thermoelectric effect 142
 - Thomson effect 142
 - transformations, between coordinates
 - 113
 - typographical conventions 13
- U** Unicode standard 106
- use conditions 84
 - user community, COMSOL 13
 - user input 44
 - user input, rename 132
 - user inputs 43, 118
 - accessing 108
- V** Variable Declarations section 35
- Variable Definitions section 35
 - variables
 - adding 53
 - entering 108
 - vs. user inputs 43
 - versions, for migration 131
- W** weak constraints 72
- weak equations 66
 - web sites, COMSOL 13
 - widgets 115