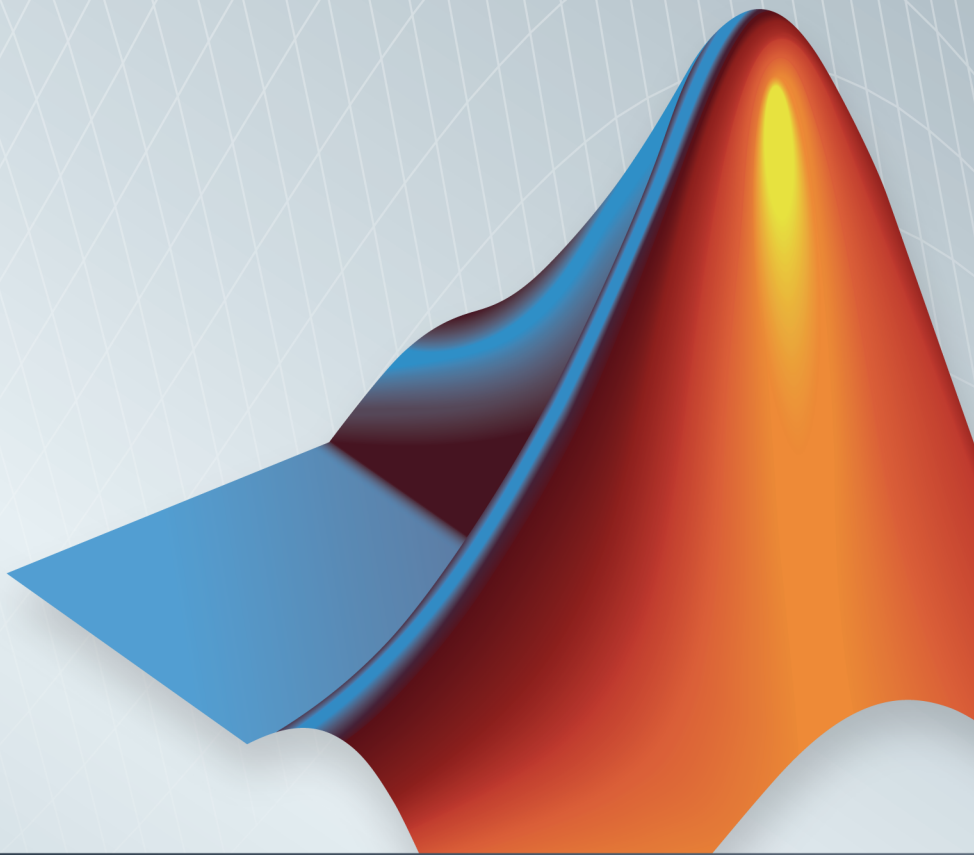


# SimMechanics™ Link

Reference

R2014b



# MATLAB®



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098

*SimMechanics™ Link Reference*

© COPYRIGHT 2003–2014 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

**FEDERAL ACQUISITION:** This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

### **Revision History**

October 2008	Online only	New for Version 3.0 (Release 2008b)
March 2009	Online only	Revised for Version 3.1 (Release 2009a)
September 2009	Online only	Revised for Version 3.1.1 (Release 2009b)
March 2010	Online only	Revised for Version 3.2 (Release 2010a)
September 2010	Online only	Revised for Version 3.2.1 (Release 2010b)
April 2011	Online only	Revised for Version 3.2.2 (Release 2011a)
September 2011	Online only	Revised for Version 3.2.3 (Release 2011b)
March 2012	Online only	Revised for Version 4.0 (Release 2012a)
September 2012	Online only	Revised for Version 4.1 (Release 2012b)
March 2013	Online only	Revised for Version 4.2 (Release 2013a)
September 2013	Online only	Revised for Version 4.3 (Release 2013b)
March 2014	Online only	Revised for Version 4.4 (Release 2014a)
October 2014	Online only	Revised for Version 4.5 (Release R2014b)



## Register and Use the Inventor Add-In

<b>Register SimMechanics Link with Inventor</b> .....	<b>1-2</b>
Software Requirements .....	1-2
Register SimMechanics Link .....	1-2
Register SimMechanics Link with Multiple CAD Installations .....	1-2
Unregister SimMechanics Link .....	1-3
<b>Constraint-Joint Mapping</b> .....	<b>1-4</b>
CAD Constraint – SimMechanics Joint Mapping .....	1-4
Supported Constraint Entity .....	1-4
Supported Constraint Entity Combinations .....	1-5
Supported SimMechanics Joints .....	1-7
<b>Constraint-Joint Mapping in SimMechanics First Generation</b> .....	<b>1-8</b>
Degrees of Freedom in SimMechanics .....	1-8
CAD Constraint – SimMechanics Joint Mapping .....	1-8
Supported Constraint Entity .....	1-9
Supported Constraint Entity Combinations .....	1-9
Supported SimMechanics Joints .....	1-11
Limitations .....	1-12
<b>Configure SimMechanics Link</b> .....	<b>1-13</b>
SimMechanics Link Settings .....	1-13
Dialog Box .....	1-13
<b>Export CAD Assembly from Autodesk Inventor Software</b> .	<b>1-15</b>
Export CAD Assembly .....	1-15
CAD Assembly Export Errors .....	1-16

## Register and Use the Creo Add-In

### 2

<b>Register SimMechanics Link with Creo</b> .....	2-2
Software Requirements .....	2-2
Registration Overview .....	2-2
Add Registration Text to Registry File .....	2-3
Add Registry File Path to Configuration File .....	2-4
SimMechanics Link Registration Example .....	2-5
Unregister SimMechanics Link .....	2-6
<b>Constraint-Joint Mapping</b> .....	2-7
CAD Constraint – SimMechanics Joint Mapping .....	2-7
Supported Constraint Entity .....	2-7
Supported Constraint Entity Combinations .....	2-8
Supported SimMechanics Joints .....	2-10
<b>Constraint-Joint Mapping in SimMechanics First</b>	
<b>Generation</b> .....	2-11
Degrees of Freedom in SimMechanics .....	2-11
CAD Constraint – SimMechanics Joint Mapping .....	2-11
Supported Constraint Entity .....	2-12
Supported Constraint Entity Combinations .....	2-12
Supported SimMechanics Joints .....	2-14
Limitations .....	2-15
<b>Configure SimMechanics Link</b> .....	2-16
SimMechanics Link Settings .....	2-16
Dialog Box .....	2-16
<b>Export CAD Assembly from Creo Software</b> .....	2-19
Export CAD Assembly .....	2-19
CAD Assembly Export Errors .....	2-20

## Register and Use SolidWorks Add-In

### 3

<b>Register SimMechanics Link with SolidWorks</b> .....	3-2
Software Requirements .....	3-2

Register SimMechanics Link .....	3-2
Add SimMechanics Link to SolidWorks Menu bar .....	3-2
Register SimMechanics Link with Multiple SolidWorks Installations .....	3-3
SimMechanics Link Menu .....	3-4
Unregister SimMechanics Link .....	3-4
<b>Mates and Joints .....</b>	<b>3-5</b>
Mates and Entities .....	3-5
Joints and Constraints .....	3-7
Mate-Joint Mapping .....	3-10
Mate-Constraint Mapping .....	3-12
Special Mate Translation Cases .....	3-12
<b>Mate-Joint Mapping in SimMechanics First Generation ..</b>	<b>3-14</b>
Degrees of Freedom in SimMechanics .....	3-14
CAD Mate – SimMechanics Joint Mapping .....	3-14
Supported Constraint Entity .....	3-15
Supported Constraint Entity Combinations .....	3-15
Supported SimMechanics Joints .....	3-18
Limitations .....	3-19
<b>Configure SimMechanics Link .....</b>	<b>3-20</b>
SimMechanics Link Settings .....	3-20
Dialog Box .....	3-20
<b>Export CAD Assembly from SolidWorks Software .....</b>	<b>3-24</b>
Export CAD Assembly .....	3-24
CAD Assembly Export Errors .....	3-25

## Function Reference

4

## API — Alphabetical List

5





# Register and Use the Inventor Add-In

---

This chapter describes how to register SimMechanics™ Link software to the Autodesk Inventor CAD platform as an Inventor add-in tool. You must complete the registration before you can export a CAD assembly in SimMechanics format.

- “Register SimMechanics Link with Inventor” on page 1-2
- “Constraint-Joint Mapping” on page 1-4
- “Constraint-Joint Mapping in SimMechanics First Generation” on page 1-8
- “Configure SimMechanics Link” on page 1-13
- “Export CAD Assembly from Autodesk Inventor Software” on page 1-15

## Register SimMechanics Link with Inventor

<b>In this section...</b>
“Software Requirements” on page 1-2
“Register SimMechanics Link” on page 1-2
“Register SimMechanics Link with Multiple CAD Installations” on page 1-2
“Unregister SimMechanics Link” on page 1-3

Before you can export a CAD assembly from the Autodesk Inventor® platform, you must register the SimMechanics Link utility with Inventor. The registration procedure adds a SimMechanics Link add-in tool to the CAD platform. Use the add-in tool to export a CAD assembly in SimMechanics format.

### Software Requirements

Registration requires that the following two products be installed on your computer:

- Inventor
- SimMechanics Link — See “Install and Register SimMechanics Link Software”

### Register SimMechanics Link

To register SimMechanics Link with the CAD platform:

- 1 Start a new MATLAB® session.
- 2 At the MATLAB command line, enter `smlink_linkinv`.
- 3 MATLAB displays a message stating that linking was successful. You can close MATLAB.

On startup, the CAD platform displays a **SimMechanics Link** menu item in the Add-In menu. The menu item appears *only* when a CAD assembly (extension `.iam`) is open.

### Register SimMechanics Link with Multiple CAD Installations

If you have multiple Inventor installations on your computer, the command `smlink_linkinv` registers SimMechanics Link with *all* installations.

You can register *one* SimMechanics Link version as a CAD add-in tool. If you have multiple SimMechanics Link versions, you must unregister the current version before registering a new version.

## Unregister SimMechanics Link

To unregister the SimMechanics Link add-in from a Inventor installation:

- At the MATLAB command line, enter `smlink_unlinkinv`.

The command removes the add-in from the Inventor registry. The SimMechanics Link add-in no longer appears in the menu bar of the CAD platform.

If you have multiple installations of Inventor on your computer, the command `smlink_unlinkinv` removes the SimMechanics Link add-in from *all* installations.

## Constraint-Joint Mapping

<b>In this section...</b>
“CAD Constraint – SimMechanics Joint Mapping” on page 1-4
“Supported Constraint Entity” on page 1-4
“Supported Constraint Entity Combinations” on page 1-5
“Supported SimMechanics Joints” on page 1-7

### CAD Constraint – SimMechanics Joint Mapping

During CAD export, SimMechanics Link maps Inventor constraints between parts to SimMechanics joints between rigid bodies. CAD constraints and SimMechanics joints do not follow a one-to-one correspondence — multiple constraints can map into a single joint. All SimMechanics joints contain a combination of three joint primitives: Prismatic, Revolute, and Spherical. The Weld Joint block contains zero joint primitives, and therefore zero degrees of freedom. The following table identifies the degrees of freedom of each joint primitive.

Primitive	Abbreviation	Motion Type	Number of DoFs
Prismatic	P	Translational	1
Revolute	R	Rotational	1
Spherical	S	Rotational	3

### Supported Constraint Entity

Depending on the constraint combination, SimMechanics Link utility supports the following Inventor constraint entities:

Entity	Description
Circle/Arc	Circular edge/arc sketch segment*
Ellipse/Arc	Elliptical edge/arc sketch segment*
Cone	Conical face
Cylinder	Cylindrical face

Entity	Description
Line	Linear edge/sketch segment/reference axis
Plane	Reference plane or planar face
Point	Vertex/sketch point/reference point

\* A complete circle or ellipse is a special case of a circular or elliptical arc.

## Supported Constraint Entity Combinations

The following sections list the constraint-entity combinations that SimMechanics Link supports for different constraint types.

---

**Note:** If the SimMechanics Link exporter cannot translate a constraint–constraint entity combination into a supported SimMechanics joint with DoFs, it converts the combination into a weld (W) primitive.

---

### Coincident Constraint

The following table identifies supported constraint-entity combinations for the Coincident constraint. A # indicates the combination is supported.

		Constraint-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint-Entity 1	Point	#					
	Line		#	#			
	Plane		#	#			#
	Cylinder				#	#	#
	Cone				#	#	#
	Circle/Arc			#	#	#	#

### Concentric Constraint

The following table identifies supported constraint-entity combinations for the Concentric constraint. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						
	Line					#	#
	Plane						
	Cylinder				#	#	#
	Cone		#		#	#	#
	Circle/Arc		#		#	#	#

**Distance Constraint**

The following table identifies supported constraint-entity combinations for the Distance constraint. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point	#		#			
	Line			#			
	Plane	#	#	#			
	Cylinder						
	Cone						
	Circle/Arc						

**Angle Constraint**

The following table identifies supported constraint-entity combinations for the Angle constraint. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						

	<b>Line</b>		#				
	<b>Plane</b>			#			
	<b>Cylinder</b>						
	<b>Cone</b>						
	<b>Circle/Arc</b>						

## Supported SimMechanics Joints

The SimMechanics Link utility supports the following SimMechanics joint-primitive combinations.

<b>Primitive Combination</b>	<b>SimMechanics Block</b>
P	Prismatic Joint
PP	Rectangular Joint
PPP	Cartesian joint
S	Spherical joint
R	Revolute Joint
PR	Cylindrical Joint
PPR	Planar Joint
PPPS	6-DOF Joint
W	Weld joint

## Constraint-Joint Mapping in SimMechanics First Generation

**In this section...**

- “Degrees of Freedom in SimMechanics” on page 1-8
- “CAD Constraint – SimMechanics Joint Mapping” on page 1-8
- “Supported Constraint Entity” on page 1-9
- “Supported Constraint Entity Combinations” on page 1-9
- “Supported SimMechanics Joints” on page 1-11
- “Limitations” on page 1-12

In Autodesk Inventor, unconstrained parts have six mechanical degrees of freedom (DoFs) that describe how the parts move with respect to each other. Of the six degrees of freedom, three are rotational and three are translational. Applying a constraint between two parts eliminates degrees of freedom between the two parts. Constraints can remove between zero and six degrees of freedom.

### Degrees of Freedom in SimMechanics

SimMechanics First Generation assigns zero degrees of freedom to an unconstrained rigid body. Connecting the rigid body to a joint or constraint block increases the mechanical degrees of freedom available to the rigid body.

Rigid Body Condition	First-Generation DoF
Not connected to joints, constraints, or World Frame	0
Connected to Joints or Constraints blocks	Add degrees of freedom according to joint or constraint

### CAD Constraint – SimMechanics Joint Mapping

During CAD export, SimMechanics Link maps Inventor constraints between parts to SimMechanics joints between rigid bodies. CAD constraints and SimMechanics joints do not follow a one-to-one correspondence — multiple constraints can map into a single joint. All SimMechanics joints contain a combination of three joint primitives: Prismatic,



Revolute, and Spherical. The Weld Joint block contains zero joint primitives, and therefore zero degrees of freedom. The following table identifies the degrees of freedom of each joint primitive.

Primitive	Abbreviation	Motion Type	Number of DoFs
Prismatic	P	Translational	1
Revolute	R	Rotational	1
Spherical	S	Rotational	3

## Supported Constraint Entity

Depending on the constraint combination, SimMechanics Link utility supports the following Inventor constraint entities:

Entity	Description
Circle/Arc	Circular edge/arc sketch segment*
Ellipse/Arc	Elliptical edge/arc sketch segment*
Cone	Conical face
Cylinder	Cylindrical face
Line	Linear edge/sketch segment/reference axis
Plane	Reference plane or planar face
Point	Vertex/sketch point/reference point

\* A complete circle or ellipse is a special case of a circular or elliptical arc.

## Supported Constraint Entity Combinations

The following sections list the constraint-entity combinations that SimMechanics Link supports for different constraint types.

---

**Note:** If the SimMechanics Link exporter cannot translate a constraint–constraint entity combination into a supported SimMechanics joint with DoFs, it converts the combination into a weld (W) primitive.

---

**Coincident Constraint**

The following table identifies supported constraint-entity combinations for the Coincident constraint. A # indicates the combination is supported.

		Constraint-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint-Entity 1	Point	#					
	Line		#	#			
	Plane		#	#			#
	Cylinder				#	#	#
	Cone				#	#	#
	Circle/Arc			#	#	#	#

**Concentric Constraint**

The following table identifies supported constraint-entity combinations for the Concentric constraint. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						
	Line					#	#
	Plane						
	Cylinder				#	#	#
	Cone		#		#	#	#
	Circle/Arc		#		#	#	#

**Distance Constraint**

The following table identifies supported constraint-entity combinations for the Distance constraint. A # indicates the combination is supported.

		Constraint Entity 2
--	--	---------------------

		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
<b>Constraint Entity 1</b>	<b>Point</b>	#		#			
	<b>Line</b>			#			
	<b>Plane</b>	#	#	#			
	<b>Cylinder</b>						
	<b>Cone</b>						
	<b>Circle/Arc</b>						

### Angle Constraint

The following table identifies supported constraint-entity combinations for the Angle constraint. A # indicates the combination is supported.

		<b>Constraint Entity 2</b>					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
<b>Constraint Entity 1</b>	<b>Point</b>						
	<b>Line</b>		#				
	<b>Plane</b>			#			
	<b>Cylinder</b>						
	<b>Cone</b>						
	<b>Circle/Arc</b>						

### Supported SimMechanics Joints

The SimMechanics Link utility supports the following SimMechanics joint-primitive combinations.

<b>Primitive Combination</b>	<b>SimMechanics Block</b>
P	Prismatic
PP	In-Plane
PPP	Custom Joint

<b>Primitive Combination</b>	<b>SimMechanics Block</b>
PPPR	Custom Joint
S	Spherical
R-S	Revolute-Spherical
R	Revolute
PR	Cylindrical
PPR	Planar
PPPS	Six-DoF
R-R	Revolute-Revolute
S-S	Spherical-Spherical
W	Weld

**Tips for Specific Constraints**

- The point-point coincident constraint maps onto a spherical joint.
- The point-point distance constraint maps onto a spherical-spherical massless connector.

**Limitations**

The following limitation applies to CAD export from Inventor.

**Weld is Default Joint**

If the SimMechanics Link utility fails to translate a CAD constraint, a Weld joint replaces the constraint.

**Restriction on Point-Point Distance Mate**

For SimMechanics Link to successfully map the CAD point-point distance constraint onto a SimMechanics spherical-spherical massless connector, the constraint must not connect to any other constraint.

## Configure SimMechanics Link

### In this section...

“SimMechanics Link Settings” on page 1-13

“Dialog Box” on page 1-13

### SimMechanics Link Settings

The SimMechanics Link add-in tool provides a Settings option. Use the option to specify:

- Tolerances — linear, angular, and relative

To access the Settings parameters:

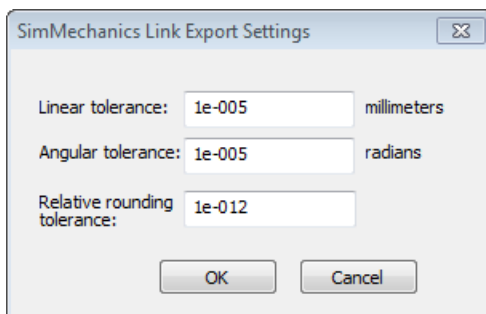
- 1 Open the assembly to export.
- 2 In the menu bar, click **Add-Ins > Settings**.

The Settings dialog box opens.

### Dialog Box

The dialog box contains two panes:

- **Assembly Tolerances** — Specifies linear, angular, and relative tolerances of exported assembly.



Enter the export tolerances for a CAD assembly. During the conversion of CAD constraints to SimMechanics joints, SimMechanics Link compares the spacing, alignment, and relative numerical errors with the export tolerances.

<b>Field</b>	<b>Default Value</b>	<b>Purpose</b>	<b>Default</b>	<b>Unit</b>
<b>Linear tolerance</b>	1e-005	Smallest significant length difference	1e-5	Unit used in assembly. The default is mm
<b>Angular tolerance</b>	1e-005	Smallest significant angle difference	1e-5	Unit used in assembly. The default is rad
<b>Relative roundoff tolerance</b>	1e-012	Smallest significant relative numerical difference	1e-12	—

# Export CAD Assembly from Autodesk Inventor Software

## In this section...

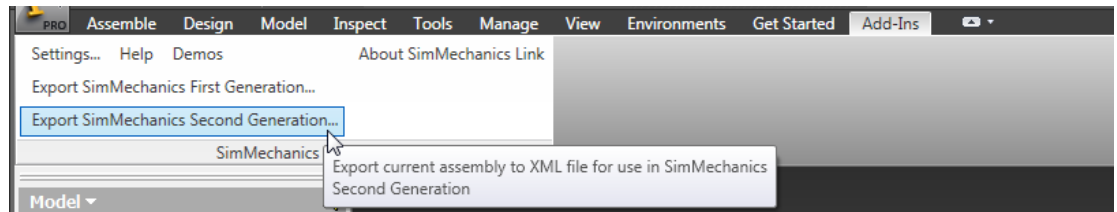
“Export CAD Assembly” on page 1-15

“CAD Assembly Export Errors” on page 1-16

## Export CAD Assembly

To export a CAD assembly:

- 1 In the menu bar of the CAD platform, click **Add-Ins**.
- 2 Click **Export SimMechanics <Generation>**, where <Generation> identifies the desired SimMechanics generation.



- 3 In the dialog box, enter the file name and select a convenient file directory.

SimMechanics Link generates:

- One XML import file.

The file contains the structure and parameters of the CAD assembly. During CAD import, SimMechanics uses the structure and parameters to autogenerate a SimMechanics model.

- A set of STL files.

Each STL file specifies the 3-D surface geometry of one CAD part. The STL files are not required to generate the model, but they are required for visualization. If you import a model without the STL files, during model update and simulation Mechanics Explorer displays a blank screen.

## **CAD Assembly Export Errors**

In the event that a CAD export error occurs:

- A dialog box displays an error message. The message identifies the CAD constraints that SimMechanics Link could not translate into joints.
- SimMechanics Link generates an error log file. Refer to the log for more information about the CAD export error. The error message identifies the name and location of an error log file.
- SimMechanics Link generates the XML file. You can import the file to generate a valid SimMechanics model, but the model may not accurately represent the original CAD assembly.
- If SimMechanics Link cannot export one or more STL files, the error message identifies the CAD parts associated with the STL files.



# Register and Use the Creo Add-In

---

This chapter describes how to register SimMechanics Link software to the Creo™ (Pro/ENGINEER) CAD platform as a Pro/TOOLKIT application. You must complete the registration before you can export a CAD assembly in SimMechanics format.

- “Register SimMechanics Link with Creo” on page 2-2
- “Constraint-Joint Mapping” on page 2-7
- “Constraint-Joint Mapping in SimMechanics First Generation” on page 2-11
- “Configure SimMechanics Link” on page 2-16
- “Export CAD Assembly from Creo Software” on page 2-19

## Register SimMechanics Link with Creo

### In this section...

- “Software Requirements” on page 2-2
- “Registration Overview” on page 2-2
- “Add Registration Text to Registry File” on page 2-3
- “Add Registry File Path to Configuration File” on page 2-4
- “SimMechanics Link Registration Example” on page 2-5
- “Unregister SimMechanics Link” on page 2-6

Once you have successfully downloaded and installed the SimMechanics Link utility, you must complete registration with the CAD platform. Registration adds the SimMechanics Link utility to the CAD platform as an add-in tool. In earlier versions of Pro/ENGINEER software, the add-in tool appears directly on the toolbar. In Creo software, the add-in tool appears as a menu item in the **Tools** menu.

Registration is manual. You must create or modify registry and configuration files for the CAD platform.

### Software Requirements

Registration requires that two products be installed on your computer:

- CAD platform — Creo or Creo predecessor Pro/ENGINEER
- SimMechanics Link

### Registration Overview

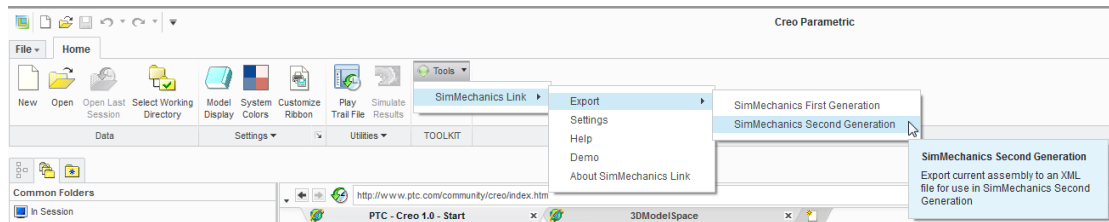
The complete registration procedure contains two steps:

Step	Purpose	For more information, see...
1. Add registration text to registry file	Registers the SimMechanics Link utility with the CAD platform.	• “Add Registration Text to Registry File” on page 2-3
2. Add registry file path to configuration file	Allow the CAD platform to load the SimMechanics Link add-in at startup.	• “Add Registry File Path to Configuration File” on page 2-4

For information about registry and configuration files, consult the documentation that accompanies your CAD platform installation.

Registration adds a SimMechanics Link menu item to the CAD platform. The following figure shows the menu item in a Creo installation.

**Note:** In Pro/ENGINEER, the SimMechanics Link add-in appears as a separate item in the menu bar.



## Add Registration Text to Registry File

Registration requires access to registry file `protk.dat`. The registry file should exist in the `protoolkit` directory of your CAD platform. If you cannot locate the file, create a new registry file. Save the file in a convenient directory as `<filename>.dat`, where `<filename>` is a name of your choice.

To add registration text to the registry file:

- 1 Open the Pro/TOOLKIT application registry file. In the registry file, each line contains a predefined parameter followed by a value.
- 2 Add the SimMechanics Link registration text appropriate to your CAD platform.

## Creo, Pro/ENGINEER Wildfire 4.0 and Later Versions

In the registry file, insert:

```
NAME SimMechanics Link
STARTUP dll
EXEC_FILE $matlabroot/bin/arch/c1_proe2sm.dll
```

```
TEXT_DIR $matlabroot/toolbox/physmod/smlink/cad_systems/proe/  
UNICODE_ENCODING false  
END
```

## Pro/ENGINEER Wildfire 3.0 and Earlier Versions

As above. Omit the line `UNICODE_ENCODING false`:

```
NAME SimMechanics Link  
STARTUP dll  
EXEC_FILE $matlabroot/bin/arch/cl_proe2sm.dll  
TEXT_DIR $matlabroot/toolbox/physmod/smlink/cad_systems/proe/  
END
```

- 3 Save the registry file.

### Add Registry File Path to Configuration File

The configuration file of your CAD platform must contain the location of the registry file. This step allows the CAD platform to load the SimMechanics Link utility as an add-in tool on startup.

The configuration file has name `config.pro`. Look for the file in the following directories.

## Creo

```
<creoroot>/<arch>/Common Files/F000/text
```

- `<creoroot>` is the root directory of your Creo installation.
- `<arch>` is the CAD platform architecture. For example, `x64` for 64-bit architectures.

## Pro/ENGINEER

```
<proeroot>/<arch>/text
```

- `<proeroot>` is the root directory of your Pro/ENGINEER installation.
- `<arch>` is the CAD platform architecture. For example, `x64` for 64-bit architectures.

---

**Note:** The directory of file `config.pro` may differ from the directories provided.

---

If you cannot locate the configuration file for your CAD platform, create a new file. Use a text editor of your choice. Save the new file as `<filename>.pro`, where `<filename>` is a name of your choice. You must save the file in one of two folder:

- CAD platform folder
- Startup folder

To find the startup folder, open the CAD platform and click **File > Open**.

- 1 Open the configuration file for your CAD platform.
- 2 At the bottom of the configuration file, add a new line with the absolute path to the registry file:

```
toolkit_registry_file <absolute_path>/<registry_filename>.dat
```

`toolkit_registry_file` is a predefined parameter that specifies the location of the registry file. `<absolute_path>` is the absolute path to the registry file. `<registry_filename>` is the name of the registry file.

## SimMechanics Link Registration Example

This example illustrates the registration procedure for a Pro/ENGINEER CAD platform. In the example, you create new registry and configuration files:

- The registry file provides registration information for the SimMechanics Link utility.
- The configuration platform provides the location of the registry file. This step allows the CAD platform can load the SimMechanics Link add-in on startup.

After successful completion of the example, the CAD platform loads SimMechanics Link as an add-in at startup.

### Create Registry File

To create the registry file:

- 1 Create directory `C:\data`
- 2 With a text editor, create a new file.
- 3 In the new file, enter:

```
NAME SimMechanics Link
STARTUP dll
EXEC_FILE C:\Program Files\MATLAB\bin\win32\cl_proe2sm.dll
TEXT_DIR C:\Program Files\MATLAB\toolbox\physmod\smlink\cad_systems\proe\text
UNICODE_ENCODING false
END
```

- 4 Save the file in directory `C:\data` with name `smlink.dat`. This is the registry file that contains the registration information for SimMechanics Link.

### Create Configuration File

To create the configuration file:

- 1 With a text editor, create a new file.
- 2 In the new file, enter:  

```
toolkit_registry_file c:\data\smlink.dat
```
- 3 Save the file as `config.pro` in one of two directories:

- CAD installation directory:

```
<proeroot>/<arch>/text
```

`proeroot` is the root of the Pro/ENGINEER installation. For example,

```
C:/Program Files/Proe
```

`arch` is the architecture of the CAD installation. For example, `x64`.

- CAD startup directory — the directory a CAD session starts in.

### Unregister SimMechanics Link

To unregister SimMechanics Link with either Creo or Pro/ENGINEER CAD platforms, follow these steps.

If you...	Then...
Created a new registry file for your CAD platform	Delete the registry file path from the configuration file
Added registration text to an existing registry file	Delete the registration text from the registry file

## Constraint-Joint Mapping

### In this section...

“CAD Constraint – SimMechanics Joint Mapping” on page 2-7

“Supported Constraint Entity” on page 2-7

“Supported Constraint Entity Combinations” on page 2-8

“Supported SimMechanics Joints” on page 2-10

### CAD Constraint – SimMechanics Joint Mapping

During CAD export, SimMechanics Link maps Creo constraints between parts to SimMechanics joints between rigid bodies. CAD constraints and SimMechanics joints do not follow a one-to-one correspondence — multiple constraints can map into a single joint. All SimMechanics joints contain a combination of three joint primitives: Prismatic, Revolute, and Spherical. The Weld Joint block contains zero joint primitives, and therefore zero degrees of freedom. The following table identifies the degrees of freedom of each joint primitive.

Primitive	Abbreviation	Motion Type	Number of DoFs
Prismatic	P	Translational	1
Revolute	R	Rotational	1
Spherical	S	Rotational	3

### Supported Constraint Entity

Depending on the constraint combination, SimMechanics Link utility supports the following Creo constraint entities:

Entity	Description
Circle/Arc	Circular edge/arc sketch segment*
Ellipse/Arc	Elliptical edge/arc sketch segment*
Cone	Conical face
Cylinder	Cylindrical face

Entity	Description
Line	Linear edge/sketch segment/reference axis
Plane	Reference plane or planar face
Point	Vertex/sketch point/reference point

\* A complete circle or ellipse is a special case of a circular or elliptical arc.

## Supported Constraint Entity Combinations

The following sections list the constraint-entity combinations that SimMechanics Link supports for different constraint types.

---

**Note:** If the SimMechanics Link exporter cannot translate a constraint–constraint entity combination into a supported SimMechanics joint with DoFs, it converts the combination into a weld (W) primitive.

---

### Coincident Constraint

The following table identifies supported constraint-entity combinations for constraints:

- Align without offset
- Mate without offset
- Point on Line
- Edge on Surface
- Point on Surface

A # indicates the combination is supported.

		Constraint-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint-Entity 1	Point	#					
	Line		#	#			
	Plane		#	#			#



	<b>Cylinder</b>				#	#	#
	<b>Cone</b>				#	#	#
	<b>Circle/Arc</b>			#	#	#	#

### Insert Constraint

The following table identifies supported constraint-entity combinations for the Insert constraint. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						
	Line					#	#
	Plane			#			
	Cylinder		#		#	#	#
	Cone		#		#	#	#
	Circle/Arc		#		#	#	#

### Align or Mate Constraint with Translational Offset

The following table identifies supported constraint-entity combinations for the Align or Mate constraints with translational offset. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point	#		#			
	Line			#			
	Plane	#	#	#			
	Cylinder						
	Cone						
	Circle/Arc						

### Align or Mate with Rotational Offset

The following table identifies supported constraint-entity combinations for the Align or Mate constraints with rotational offset. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						
	Line		#				
	Plane			#			
	Cylinder						
	Cone						
	Circle/Arc						

### Supported SimMechanics Joints

The SimMechanics Link utility supports the following SimMechanics joint-primitive combinations.

Primitive Combination	SimMechanics Block
P	Prismatic Joint
PP	Rectangular Joint
PPP	Cartesian joint
S	Spherical joint
R	Revolute Joint
PR	Cylindrical Joint
PPR	Planar Joint
PPPS	6-DOF Joint
W	Weld joint

## Constraint-Joint Mapping in SimMechanics First Generation

### In this section...

“Degrees of Freedom in SimMechanics” on page 2-11

“CAD Constraint – SimMechanics Joint Mapping” on page 2-11

“Supported Constraint Entity” on page 2-12

“Supported Constraint Entity Combinations” on page 2-12

“Supported SimMechanics Joints” on page 2-14

“Limitations” on page 2-15

In Pro/ENGINEER, unconstrained parts have six mechanical degrees of freedom (DoFs) that describe how the parts move with respect to each other. Of the six degrees of freedom, three are rotational and three are translational. Applying a constraint between two parts eliminates degrees of freedom between the two parts. Constraints can remove between zero and six degrees of freedom.

### Degrees of Freedom in SimMechanics

SimMechanics First Generation assigns zero degrees of freedom to an unconstrained rigid body. Connecting the rigid body to a joint or constraint block increases the mechanical degrees of freedom available to the rigid body.

Rigid Body Condition	First-Generation DoF
Not connected to joints, constraints, or World Frame	0
Connected to Joints or Constraints blocks	Add degrees of freedom according to joint or constraint

### CAD Constraint – SimMechanics Joint Mapping

During CAD export, SimMechanics Link maps Pro/ENGINEER constraints between parts to SimMechanics joints between rigid bodies. CAD constraints and SimMechanics joints do not follow a one-to-one correspondence — multiple constraints can map into a single joint. All SimMechanics joints contain a combination of three joint primitives: Prismatic, Revolute, and Spherical. The Weld Joint block contains zero joint primitives, and therefore zero degrees of freedom. The following table identifies the degrees of freedom of each joint primitive.

Primitive	Abbreviation	Motion Type	Number of DoFs
Prismatic	P	Translational	1
Revolute	R	Rotational	1
Spherical	S	Rotational	3

## Supported Constraint Entity

Depending on the constraint combination, SimMechanics Link utility supports the following Creo constraint entities:

Entity	Description
Circle/Arc	Circular edge/arc sketch segment*
Ellipse/Arc	Elliptical edge/arc sketch segment*
Cone	Conical face
Cylinder	Cylindrical face
Line	Linear edge/sketch segment/reference axis
Plane	Reference plane or planar face
Point	Vertex/sketch point/reference point

\* A complete circle or ellipse is a special case of a circular or elliptical arc.

## Supported Constraint Entity Combinations

The following sections list the constraint-entity combinations that SimMechanics Link supports for different constraint types.

---

**Note:** If the SimMechanics Link exporter cannot translate a constraint–constraint entity combination into a supported SimMechanics joint with DoFs, it converts the combination into a weld (W) primitive.

---

### Coincident Constraint

The following table identifies supported constraint-entity combinations for constraints:

- Align without offset
- Mate without offset
- Point on Line
- Edge on Surface
- Point on Surface

A # indicates the combination is supported.

		Constraint-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint-Entity 1	Point	#					
	Line		#	#			
	Plane		#	#			#
	Cylinder				#	#	#
	Cone				#	#	#
	Circle/Arc			#	#	#	#

### Insert Constraint

The following table identifies supported constraint-entity combinations for the Insert constraint. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						
	Line					#	#
	Plane			#			
	Cylinder		#		#	#	#
	Cone		#		#	#	#
	Circle/Arc		#		#	#	#

### Align or Mate Constraint with Translational Offset

The following table identifies supported constraint-entity combinations for the Align or Mate constraints with translational offset. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point	#		#			
	Line			#			
	Plane	#	#	#			
	Cylinder						
	Cone						
	Circle/Arc						

### Align or Mate with Rotational Offset

The following table identifies supported constraint-entity combinations for the Align or Mate constraints with rotational offset. A # indicates the combination is supported.

		Constraint Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Constraint Entity 1	Point						
	Line		#				
	Plane			#			
	Cylinder						
	Cone						
	Circle/Arc						

### Supported SimMechanics Joints

The SimMechanics Link utility supports the following SimMechanics joint-primitive combinations.

<b>Primitive Combination</b>	<b>SimMechanics Block</b>
P	Prismatic
PP	In-Plane
PPP	Custom Joint
PPPR	Custom Joint
S	Spherical
R-S	Revolute-Spherical
R	Revolute
PR	Cylindrical
PPR	Planar
PPPS	Six-DoF
R-R	Revolute-Revolute
S-S	Spherical-Spherical
W	Weld

## **Limitations**

The following limitation applies to CAD export from Pro/ENGINEER.

### **Weld is Default Joint**

If the SimMechanics Link utility fails to translate a CAD constraint, a Weld joint replaces the constraint.

## Configure SimMechanics Link

In this section...
“SimMechanics Link Settings” on page 2-16
“Dialog Box” on page 2-16

### SimMechanics Link Settings

The SimMechanics Link add-in tool provides a Settings option. Use the option to specify:

- Tolerances — linear, angular, and relative
- Coordinate systems to export

To access the Settings parameters:

- 1 Open the assembly to export.
- 2 In the menu bar, click **Tools > SimMechanics Link**.
- 3 Click **Settings**.

The Settings dialog box opens.

### Dialog Box

The dialog box contains two panes:

- **Assembly Tolerances** — Specifies linear, angular, and relative tolerances of exported assembly.
- **Export Coordinate Systems** — Determines what coordinate systems to export.



### Assembly Tolerances

Enter the export tolerances for a CAD assembly. During the conversion of CAD constraints to SimMechanics joints, SimMechanics Link compares the spacing, alignment, and relative numerical errors with the export tolerances.

Field	Default Value	Purpose	Default	Unit
<b>Linear tolerance</b>	1e-005	Smallest significant length difference	1e-5	Units used in assembly
<b>Angular tolerance</b>	1e-005	Smallest significant angle difference	1e-5	Units used in assembly
<b>Relative roundoff tolerance</b>	1e-012	Smallest significant relative numerical difference	1e-12	—

### Export Coordinate Systems

Specify which reference coordinate systems to export. The coordinate systems are independent of constraints between parts. Options include:

- **Do not export coordinate systems** — Export no coordinate systems.
- **Export only CSs with this prefix** — Export only coordinate systems with the specified name prefix. If the prefix field is empty, SimMechanics Link exports all reference coordinate systems.

## Export CAD Assembly from Creo Software

### In this section...

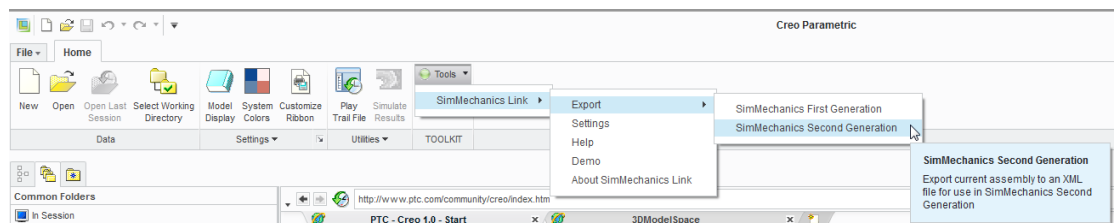
“Export CAD Assembly” on page 2-19

“CAD Assembly Export Errors” on page 2-20

## Export CAD Assembly

To export a CAD assembly:

- 1 In the menu bar of the CAD platform, click **Tools**.
- 2 Click **SimMechanics Link > Export**.



- 3 Click **SimMechanics <Generation>**, where <Generation> identifies the desired SimMechanics generation.
- 4 In the dialog box, enter the file name and select a convenient file directory.

SimMechanics Link generates:

- One XML import file.

The file contains the structure and parameters of the CAD assembly. During CAD import, SimMechanics uses the structure and parameters to autogenerate a SimMechanics model.

- A set of STL files.

Each STL file specifies the 3-D surface geometry of one CAD part. The STL files are not required to generate the model, but they are required for visualization. If you import a model without the STL files, during model update and simulation Mechanics Explorer displays a blank screen.

## **CAD Assembly Export Errors**

In the event that a CAD export error occurs:

- A dialog box displays an error message. The message identifies the CAD constraints that SimMechanics Link could not translate into joints.
- SimMechanics Link generates an error log file. Refer to the log for more information about the CAD export error. The error message identifies the name and location of an error log file.
- SimMechanics Link generates the XML file. You can import the file to generate a valid SimMechanics model, but the model may not accurately represent the original CAD assembly.
- If SimMechanics Link cannot export one or more STL files, the error message identifies the CAD parts associated with the STL files.

# Register and Use SolidWorks Add-In

---

This chapter describes how to register SimMechanics Link software to the SolidWorks CAD platform as a SolidWorks add-in. You must complete the registration before you can export a CAD assembly in SimMechanics format.

- “Register SimMechanics Link with SolidWorks” on page 3-2
- “Mates and Joints” on page 3-5
- “Mate-Joint Mapping in SimMechanics First Generation” on page 3-14
- “Configure SimMechanics Link” on page 3-20
- “Export CAD Assembly from SolidWorks Software” on page 3-24

## Register SimMechanics Link with SolidWorks

### In this section...

“Software Requirements” on page 3-2

“Register SimMechanics Link” on page 3-2

“Add SimMechanics Link to SolidWorks Menu bar” on page 3-2

“Register SimMechanics Link with Multiple SolidWorks Installations” on page 3-3

“SimMechanics Link Menu” on page 3-4

“Unregister SimMechanics Link” on page 3-4

Before you can export a CAD assembly from the SolidWorks platform, you must register the SimMechanics Link utility with SolidWorks. The registration procedure adds a SimMechanics Link add-in tool to the CAD platform. Use the add-in tool to export a CAD assembly in SimMechanics format.

### Software Requirements

Registration requires that the following two products be installed on your computer:

- SolidWorks
- SimMechanics Link — See “Install and Register SimMechanics Link Software”

### Register SimMechanics Link

To register SimMechanics Link with the CAD platform:

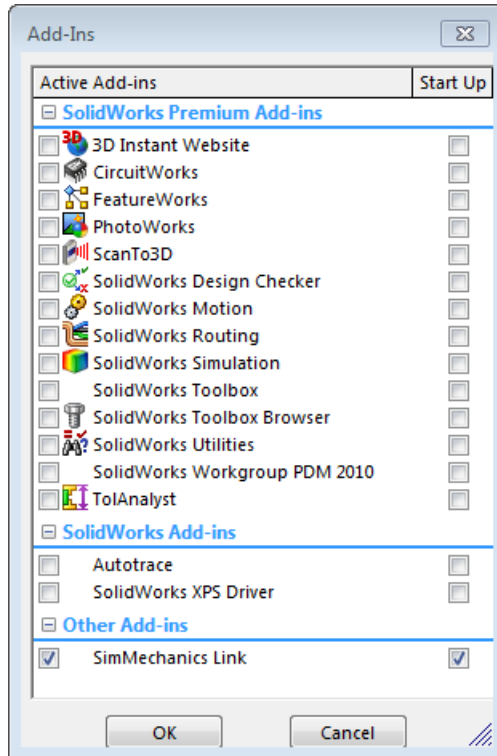
- 1 Start a new MATLAB session.
- 2 At the MATLAB command line, enter `smlink_linksw`.
- 3 MATLAB displays a message stating that linking was successful. You can close MATLAB.

### Add SimMechanics Link to SolidWorks Menu bar

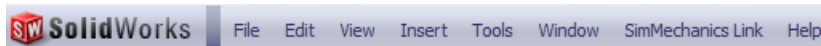
SimMechanics Link is now registered as a SolidWorks add-in tool. To use the tool, select the tool as an active add-in:

- 1 Open SolidWorks.

- 2 In the menu bar, select **Tools > Add-Ins**.
- 3 In the **Add-Ins** dialog box, select **SimMechanics Link**.



The menu bar of the CAD platform displays a **SimMechanics Link** menu item.



## Register SimMechanics Link with Multiple SolidWorks Installations

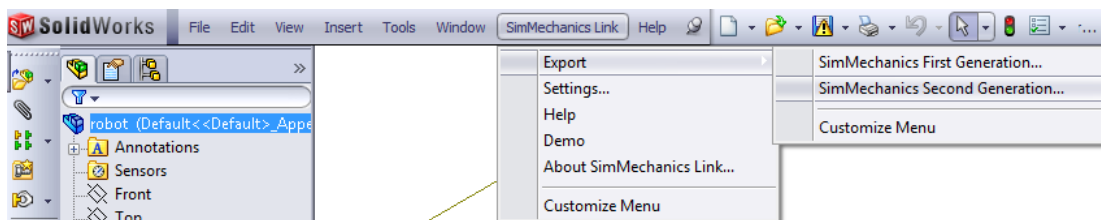
If you have multiple SolidWorks installations on your computer, the command `smlink_linksw` registers SimMechanics Link with *all* installations.

To add **SimMechanics Link** to the CAD platform menu bar, you must still open each installation of SolidWorks, and select **SimMechanics Link** from the **Tools > Add-Ins** menu.

You can register *one* SimMechanics Link version as a CAD add-in tool. If you have multiple SimMechanics Link versions, you must remove the current registration before adding a new one.

## SimMechanics Link Menu

Following registration, the menu bar of the CAD platform displays a **SimMechanics Link** menu item. The menu item appears *only* when a CAD assembly (extension .sldasm) is open.



## Unregister SimMechanics Link

To unregister the SimMechanics Link add-in from a SolidWorks installation:

- At the MATLAB command line, enter `smlink_unlinksw`.

The command removes the add-in from the SolidWorks registry. The SimMechanics Link add-in no longer appears in the menu bar of the CAD platform.

If you have multiple installations of SolidWorks on your computer, the command `smlink_unlinksw` removes the SimMechanics Link add-in from *all* installations.



# Mates and Joints

## In this section...

“Mates and Entities” on page 3-5

“Joints and Constraints” on page 3-7

“Mate-Joint Mapping” on page 3-10

“Mate-Constraint Mapping” on page 3-12

“Special Mate Translation Cases” on page 3-12

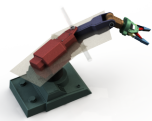
## Mates and Entities

In a SolidWorks assembly, you connect parts using mates. Each mate applies a geometric relationship between mate entities on different parts. Mates include parallel, concentric, and coincident types. Mate entities include points, lines, and surfaces.







Consider the connection between the upper-arm and forearm parts of a robotic arm assembly. This connection allows the two parts to rotate with respect to each other about a single axis and therefore has one rotational degree of freedom. You specify this degree of freedom by applying mates such as:

- One concentric mate between the cylindrical hinge surfaces of the two parts. This mate reduces the joint degrees of freedom to two—one translational, along the cylindrical axis, and one rotational, about the same axis.
- One coincident mate between two planes normal to the cylindrical axis. This mate removes the translational degree of freedom between the two parts. These can now only rotate about the common hinge axis.







The figure shows the mated surfaces in a CAD robotic arm assembly.



For the purposes of CAD import, SimMechanics software supports these mate entities:

- Circle/arc 
- Cone 
- Cylinder 
- Line 
- Plane 
- Point 

SimMechanics also supports these mates:

- Angle 
- Coincident 
- Concentric 
- Distance 
- Parallel 
- Perpendicular 

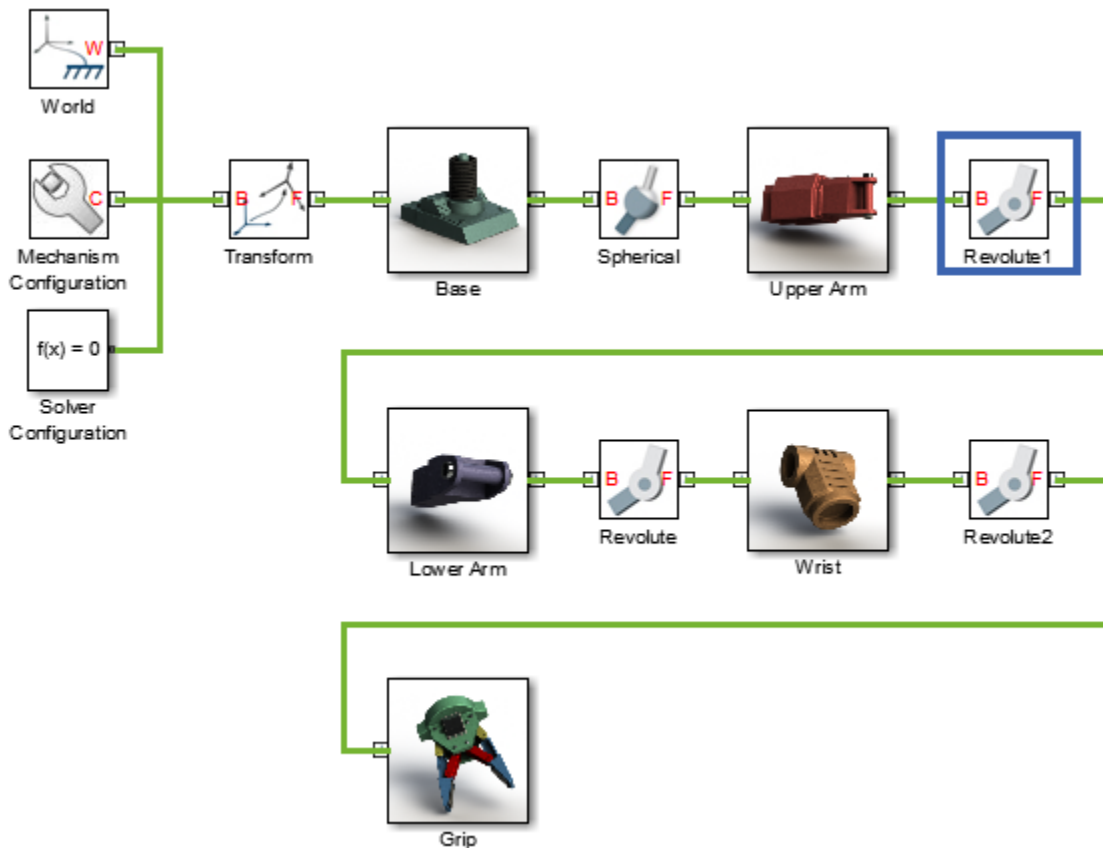
Supported mates are valid only for certain entity pairs. The table shows the entity pairs compatible with the supported mates. This table is symmetric with respect to the diagonal row.

		Mate Entity 2					
		○	▲	◐	/	▭	*
Mate Entity 1	○						
	▲						
	◐						
	/						
	▭						
	*						

## Joints and Constraints

Joint and Constraint blocks are the SimMechanics equivalent of SolidWorks mates. They apply between two bodies the kinematic constraints that determine how they can move. A Revolute Joint block is an example. This block removes five degrees of freedom between two bodies, allowing them only to rotate about a common axis.

Consider the connection between the upper-arm and forearm parts of the robotic arm assembly. This connection provides a single rotational degree of freedom and therefore is a revolute joint. During CAD import, the mates between the two parts translate into a Revolute Joint block between two rigid body subsystems. The figure shows this joint block in an imported model (polished for clarity).



Joint blocks are combinations of joint primitives. Each primitive functions as a building block, providing the joint with a specific number and type of degrees of freedom. Each primitive also provides any modeling capability unique to its degrees of freedom—initial state targets, internal mechanics, actuation, and sensing. Joint primitives can be:

- Prismatic — Joint primitive with one translational DOF. This primitive enables the connected bodies to translate with respect to each other along a single standard axis (X, Y, or Z). The figure shows an example of a prismatic joint.



- Revolute — Joint primitive with one rotational DOF. This primitive enables the connected bodies to rotate with respect to each other about a single standard axis (X, Y, or Z). The figure shows an example of a revolute joint.



- Spherical — Joint primitive with three rotational DOFs. This primitive enables the connected bodies to rotate with respect to each other about an arbitrary [X Y Z] axis. The figure shows an example of a spherical joint.



A Joint block can have up to three revolute primitives or, alternatively, one spherical primitive. Each revolute primitive aligns with a different axis in the joint base frame—X, Y, or Z. These are denoted Rx, Ry, and Rz, respectively. The spherical primitive, denoted S, enables rotation about a general axis [X, Y, Z] in the joint base frame.

Similarly, a Joint block can also have up to three prismatic primitives. Each primitive aligns with a different axis in the joint base frame—X, Y, or Z. These are denoted Px, Py, and Pz, respectively. The table shows the Joint blocks that CAD mates can translate into, the joint primitives the blocks contain, and the degrees of freedom they provide. T and R denote translational and rotational DOFs.

Joint Block	Joint Primitives	Joint DOFs
6-DOF Joint	Px Py Pz Rx Ry Rz S	3 T + 3 R
Cartesian Joint	Px Py Pz Rx Ry Rz S	3 T + 0 R
Cylindrical Joint	Px Py Pz Rx Ry Rz S	1 T + 1 R
Planar Joint	Px Py Pz Rx Ry Rz S	2 T + 1 R
Prismatic Joint	Px Py Pz Rx Ry Rz S	1 T + 0 R
Rectangular Joint	Px Py Pz Rx Ry Rz S	2 T + 0 R
Revolute Joint	Px Py Pz Rx Ry Rz S	0 T + 1 R
Spherical Joint	Px Py Pz Rx Ry Rz S	0 T + 3 R
Telescoping Joint	Px Py Pz Rx Ry Rz S	1 T + 3 R
Universal Joint	Px Py Pz Rx Ry Rz S	0 T + 2 R
Weld Joint	Px Py Pz Rx Ry Rz S	0 T + 0 R

By defining the relative degrees of freedom between two bodies, Joint blocks partially determine how these bodies can move with respect to each other. Constraint blocks enable you to impose additional restrictions on their motion. CAD mates can translate into these Constraint blocks:

- Angle Constraint
- Distance Constraint

### Mate-Joint Mapping

The table shows the Joint blocks that different mate combinations map into. Different mate combinations can map into the same joint. This happens if the mate combinations provide the same degrees of freedom between the parts they join. For a legend of the icons in the table, see “Mates and Entities” on page 3-5.

		CAD Mates				
		Mate I		Mate II		Additional Conditions
		Type	Entity Pair	Type	Entity Pair	
SimMechanics Joint	Cartesian					
	Cylindrical					
	Planar					
	Prismatic			or		1
				or		
				or		2
				or		
		or		or		3
	Rectangular	or		, , or		
		or				
	Revolute			or		4
				or		5
	Spherical					
	Universal					

Mate pairs marked with a note number must satisfy additional requirements. This list outlines these requirements:

- 1 Cylinder axes in mate I must be parallel to planes in mate II.

- 2 Lines in mate I must be parallel to planes in mate II.
- 3 Planes in mate I must not be parallel to planes in mate II.
- 4 Cylinder axes in mate I must be perpendicular to planes in mate II.
- 5 Lines in mate I must be perpendicular to planes in mate II.

## Mate-Constraint Mapping

The table shows the Constraint blocks that different mate combinations map into. Different mates map into the same Constraint block if they provide the same degrees of freedom. Angle mates must have values of 0, 90, 180, or 270 degrees. Other mate settings are not supported. For a legend of the icons in the table, see “Mates and Entities” on page 3-5.

		CAD Mate		
		Type	Entity Pairs	Setting
SimMechanics Constraint	Angle (Perpendicular)		//	90° or 270°
	Angle (Parallel)		//	0°
			//	
	Distance			

## Special Mate Translation Cases

The lack of mates between parts, combinations of mates that fully constrain two parts, and unsupported mates are special translation cases. Here is how SimMechanics software handles these cases:

- Unsupported mates between parts translate into rigid connections between rigid bodies. The rigid connections can be in the form of Weld Joint blocks or direct frame connection lines between the rigid bodies. These connections are meant to be temporary. After CAD import, search your model for rigid connections and, if appropriate, replace them with other Joint and Constraint blocks.



- Mate combinations that fully constrain two parts translate into rigid connections between two rigid bodies. The rigid connections can be in the form of Weld Joint blocks or direct frame connection lines between the rigid bodies. These rigid connections accurately model the degrees of freedom between the two bodies and do not need to be replaced.
- The absence of a mate between a part and the rest of the assembly translates into a 6-DOF Joint block between a rigid body and the World frame. This joint block provides the rigid body the six degrees of freedom that the CAD part has within the CAD assembly.

## Mate-Joint Mapping in SimMechanics First Generation

In this section...
“Degrees of Freedom in SimMechanics” on page 3-14
“CAD Mate – SimMechanics Joint Mapping” on page 3-14
“Supported Constraint Entity” on page 3-15
“Supported Constraint Entity Combinations” on page 3-15
“Supported SimMechanics Joints” on page 3-18
“Limitations” on page 3-19

In SolidWorks, unmated parts have six mechanical degrees of freedom (DoFs) that describe how the parts can move with respect to each other. Of the six degrees of freedom, three are rotational and three are translational. Applying a mate between two parts eliminates degrees of freedom between the two parts. Mates can remove between zero and six degrees of freedom.

### Degrees of Freedom in SimMechanics

SimMechanics assigns six degrees of freedom to an unconstrained rigid body. The unconstrained rigid body behaves as a free body — it can rotate and translate, about or along three mutually orthogonal axes. The following table lists the degrees of freedom of a rigid body in different configurations.

Rigid Body Condition	Degrees of Freedom
Not connected to joints, constraints, or World Frame	0
Connected to Joints or Constraints blocks	Add degrees of freedom as specified by joint or constraint

### CAD Mate – SimMechanics Joint Mapping

During CAD export, SimMechanics Link maps SolidWorks mates between parts to SimMechanics joints between rigid bodies. CAD mates and SimMechanics joints do not follow a one-to-one correspondence — multiple mates can map into a single joint. All SimMechanics joints contain a combination of three joint primitives: Prismatic, Revolute,

and Spherical. The Weld Joint block contains zero joint primitives, and therefore zero degrees of freedom. The following table identifies the degrees of freedom of each joint primitive.

Primitive	Abbreviation	Motion Type	Number of DoFs
Prismatic	P	Translational	1
Revolute	R	Rotational	1
Spherical	S	Rotational	3

## Supported Constraint Entity

Depending on the constraint combination, SimMechanics Link utility supports the following Inventor constraint entities:

Entity	Description
Circle/Arc	Circular edge/arc sketch segment*
Ellipse/Arc	Elliptical edge/arc sketch segment*
Cone	Conical face
Cylinder	Cylindrical face
Line	Linear edge/sketch segment/reference axis
Plane	Reference plane or planar face
Point	Vertex/sketch point/reference point

\* A complete circle or ellipse is a special case of a circular or elliptical arc.

## Supported Constraint Entity Combinations

The following sections list the constraint-entity combinations that SimMechanics Link supports for different constraint types.

---

**Note:** If the SimMechanics Link exporter cannot translate a constraint–constraint entity combination into a supported SimMechanics joint with DoFs, it converts the combination into a weld (W) primitive.

---

**Coincident Constraint**

The following table identifies supported constraint-entity combinations for the Coincident constraint. A # indicates the combination is supported.

		Mate-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Mate-Entity 1	Point	#					
	Line		#	#			
	Plane		#	#			#
	Cylinder				#	#	#
	Cone				#	#	#
	Circle/Arc			#	#	#	#

**Concentric Mate**

The following table identifies supported constraint-entity combinations for the Concentric mate. A # indicates the combination is supported.

		Mate-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Mate-Entity 1	Point						
	Line				#	#	#
	Plane			#			
	Cylinder		#		#	#	#
	Cone		#		#	#	#
	Circle/Arc		#		#	#	#

**Perpendicular Mate**

The following table identifies supported constraint-entity combinations for the Perpendicular mate. A # indicates the combination is supported.

		Mate-Entity 2					
--	--	---------------	--	--	--	--	--

		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
<b>Mate-Entity 1</b>	<b>Point</b>						
	<b>Line</b>		#	#			
	<b>Plane</b>		#	#			
	<b>Cylinder</b>						
	<b>Cone</b>						
	<b>Circle/Arc</b>						

### Parallel Mate

The following table identifies supported constraint-entity combinations for the Parallel mate. A # indicates the combination is supported.

		<b>Mate-Entity 2</b>					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
<b>Mate-Entity 1</b>	<b>Point</b>						
	<b>Line</b>		#	#			
	<b>Plane</b>		#	#			
	<b>Cylinder</b>				#		
	<b>Cone</b>					#	
	<b>Circle/Arc</b>						

### Distance Mate

The following table identifies supported constraint-entity combinations for the Distance mate. A # indicates the combination is supported.

		<b>Mate-Entity 2</b>					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
<b>Mate-Entity 1</b>	<b>Point</b>	#		#			
	<b>Line</b>			#			

	<b>Plane</b>	#	#	#			
	<b>Cylinder</b>						
	<b>Cone</b>						
	<b>Circle/Arc</b>						

### Angle Mate

The following table identifies supported constraint-entity combinations for the Angle mate. A # indicates the combination is supported.

		Mate-Entity 2					
		Point	Line	Plane	Cylinder	Cone	Circle/ Arc
Mate-Entity 1	Point						
	Line		#				
	Plane			#			
	Cylinder						
	Cone						
	Circle/Arc						

### Supported SimMechanics Joints

The SimMechanics Link utility supports the following SimMechanics joint-primitive combinations.

Primitive Combination	SimMechanics Block
P	Prismatic
PP	In-Plane
PPP	Custom Joint
PPPR	Custom Joint
S	Spherical
R-S	Revolute-Spherical
R	Revolute

Primitive Combination	SimMechanics Block
PR	Cylindrical
PPR	Planar
PPPS	Six-DoF
R-R	Revolute-Revolute
S-S	Spherical-Spherical
W	Weld

### Tips for Specific Mates

- The point-point coincident mate maps onto a spherical joint.
- The point-point distance mate maps onto a spherical-spherical massless connector.

### Limitations

The following limitation applies to CAD export from SolidWorks.

#### Weld is Default Joint

If the SimMechanics Link utility fails to translate a CAD constraint, a Weld joint replaces the constraint.

#### Restriction on Point-Point Distance Mate

For SimMechanics Link to successfully map the CAD point-point distance mate onto a SimMechanics spherical-spherical massless connector, the mate must not connect to any other mates.

## Configure SimMechanics Link

<b>In this section...</b>
“SimMechanics Link Settings” on page 3-20
“Dialog Box” on page 3-20

### SimMechanics Link Settings

The SimMechanics Link add-in tool provides a Settings option. Use the option to specify:

- Tolerances — linear, angular, and relative
- Coordinate systems to export

To access the Settings parameters:

- 1 Open the assembly to export.
- 2 In the menu bar, click **SimMechanics Link > Settings**.

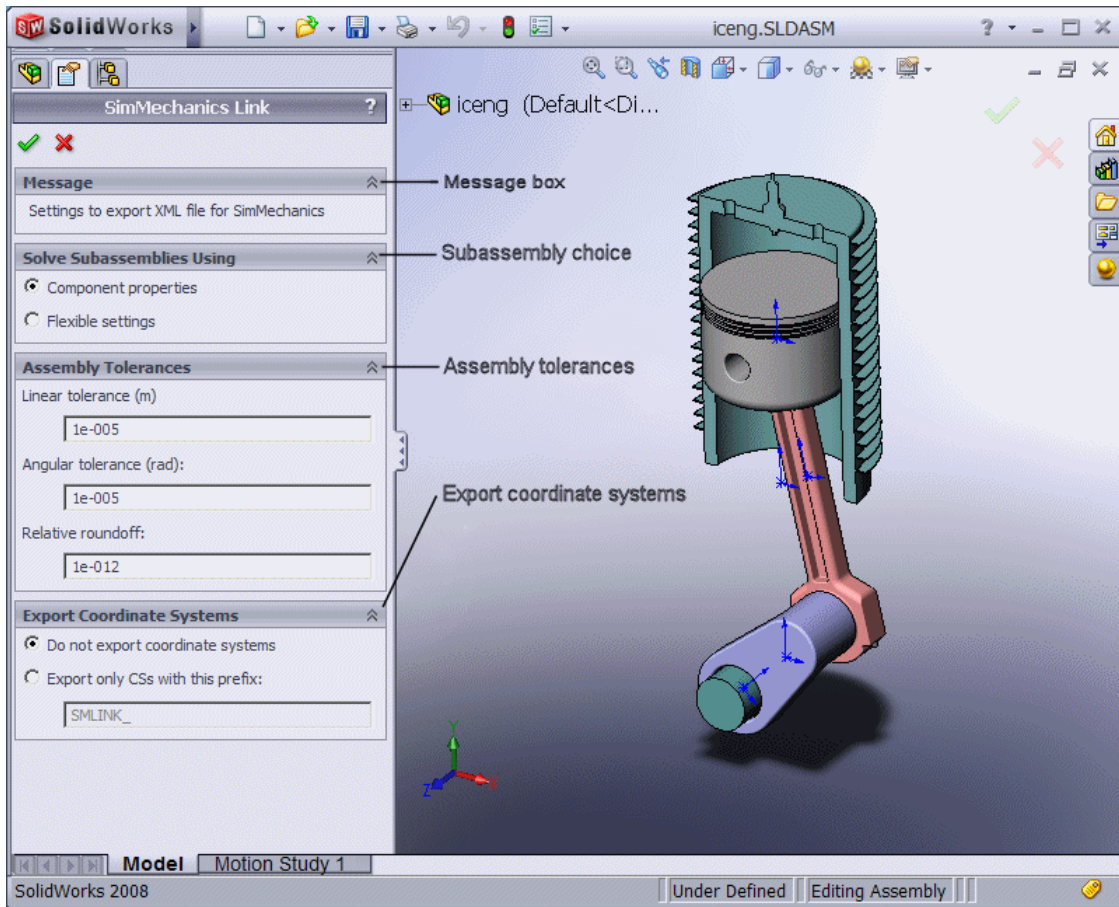
The Settings dialog box opens.

### Dialog Box




The dialog box contains four panes:

- **Message** — Describes the purpose of the dialog box. The Message box is inactive.
- **Solve Subassemblies Using** — Determines whether to export a subassembly as a rigid or flexible system.
- **Assembly Tolerances** — Specifies linear, angular, and relative tolerances of exported assembly.
- **Export Coordinate Systems** — Determines what coordinate systems to export.





### Save, Close, and Help Buttons

Click...	To...
	Save your settings and close the settings dialog box
	Close the settings dialog box without saving your settings
	Open online SimMechanics Link help

### Solve Subassemblies Using

Select how to export CAD subassemblies.

- **Component properties** — Treat rigid subassemblies as rigid, and flexible subassemblies as flexible.
- **Flexible settings** — Treat *all* subassemblies as flexible. This setting applies does not affect the original CAD assembly.

## Make Subassemblies Rigid or Flexible in SolidWorks

Subassemblies can be rigid or flexible. Rigid subassemblies behave as a single rigid body. Flexible subassemblies behave as a multibody subsystem. To make a subassembly rigid or flexible:

- 1 Right-click the subassembly.
- 2 Click **Component > Properties**.
- 3 Select between **Flexible** and **Rigid** options.

Select **Rigid** only if the motion between subassembly parts is not important in SimMechanics.

### Assembly Tolerances

Enter the export tolerances for a CAD assembly. During the conversion of CAD constraints to SimMechanics joints, SimMechanics Link compares the spacing, alignment, and relative numerical errors with the export tolerances.

Field	Default Value	Purpose	Default	Unit
<b>Linear tolerance</b>	1e - 005	Smallest significant length difference	1e-5	meter (m)
<b>Angular tolerance</b>	1e - 005	Smallest significant angle difference	1e-5	radian (rad)
<b>Relative roundoff tolerance</b>	1e - 012	Smallest significant relative numerical difference	1e-12	—

### Export Coordinate Systems

Specify which reference coordinate systems to export. The reference coordinate systems are independent of mates between parts. Options include:

- **Do not export coordinate systems** — Export no coordinate systems.
- **Export only CSs with this prefix** — Export only coordinate systems with the specified name prefix. If the prefix field is empty, SimMechanics Link exports all reference coordinate systems.

## Export CAD Assembly from SolidWorks Software

### In this section...

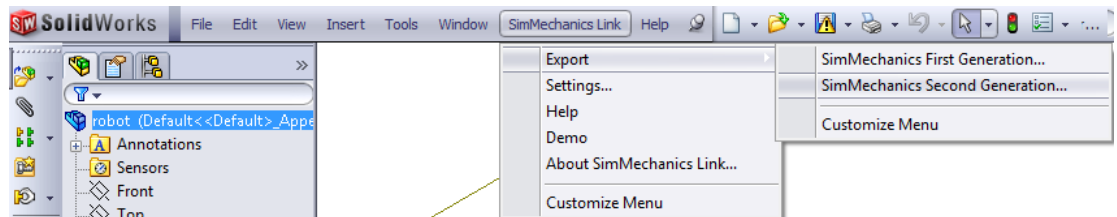
“Export CAD Assembly” on page 3-24

“CAD Assembly Export Errors” on page 3-25

### Export CAD Assembly

To export a CAD assembly:

- 1 In the menu bar of the CAD platform, click **SimMechanics Link**.
- 2 Click **Export > SimMechanics <Generation>**, where <Generation> identifies the desired SimMechanics generation.



- 3 In the dialog box, enter the file name and select a convenient file directory.

SimMechanics Link generates:

- One XML import file.

The file contains the structure and parameters of the CAD assembly. During CAD import, SimMechanics uses the structure and parameters to autogenerate a SimMechanics model.

- A set of STL files.

Each STL file specifies the 3-D surface geometry of one CAD part. The STL files are not required to generate the model, but they are required for visualization. If you import a model without the STL files, during model update and simulation Mechanics Explorer displays a blank screen.

## **CAD Assembly Export Errors**

In the event that a CAD export error occurs:

- A dialog box displays an error message. The message identifies the CAD constraints that SimMechanics Link could not translate into joints.
- SimMechanics Link generates an error log file. Refer to the log for more information about the CAD export error. The error message identifies the name and location of an error log file.
- SimMechanics Link generates the XML file. You can import the file to generate a valid SimMechanics model, but the model may not accurately represent the original CAD assembly.
- If SimMechanics Link cannot export one or more STL files, the error message identifies the CAD parts associated with the STL files.



# Function Reference

---

# smlink\_linkinv

Register and link SimMechanics Link software as Autodesk Inventor add-in

## Syntax

```
smlink_linkinv
```

## Description

smlink\_linkinv registers and links SimMechanics Link software as an add-in to Autodesk Inventor. Execute this function before you attempt to use SimMechanics Link software with Autodesk Inventor.

## Output Arguments

A message indicating that the registration and linking have worked, with the location of the add-in module, if registration and linking succeed.

An error message describing the failure, if registration and linking do not succeed.

## Definitions

*Linking* is associating a version of a CAD platform with a SimMechanics Link installation.

Depending on the CAD platform and if you use the Windows<sup>®</sup> operating system, linking a CAD platform can involve registering one of the executable SimMechanics Link libraries with Windows.

*Registering* is entering an executable module or library in the Windows registry.

## More About

- “Install and Register SimMechanics Link Software”



**See Also**

smlink\_unlinkinv

# smlink\_linksw

Register and link SimMechanics Link software as SolidWorks add-in

## Syntax

```
smlink_linksw
```

## Description

smlink\_linksw registers and links SimMechanics Link software as an add-in to SolidWorks. Execute this function before you attempt to use SimMechanics Link software with SolidWorks.

## Output Arguments

A message indicating that the linking has worked, with the location of the add-in module, if registration and linking succeed.

An error message describing the failure, if registration and linking do not succeed.

## Definitions

*Linking* is associating a version of a CAD platform with a SimMechanics Link installation.

Depending on the CAD platform and if you use the Windows operating system, linking a CAD platform can involve registering one of the executable SimMechanics Link libraries with Windows.

*Registering* is entering an executable module or library in the Windows registry.

## More About

- “Install and Register SimMechanics Link Software”

**See Also**

smlink\_unlinksw

# smlink\_unlinkinv

Unlink SimMechanics Link software as Autodesk Inventor add-in

## Syntax

```
smlink_unlinksw
```

## Description

smlink\_unlinksw unlinks SimMechanics Link software as an add-in to Autodesk Inventor.

## Output Arguments

A message indicating that the unlinking has worked, with the location of the add-in module, if unlinking succeeds.

An error message describing the failure, if unlinking does not succeed.

## Definitions

*Linking* is associating a version of a CAD platform with a SimMechanics Link installation.

Depending on the CAD platform and if you use the Windows operating system, linking a CAD platform can involve registering one of the executable SimMechanics Link libraries with Windows.

*Registering* is entering an executable module or library in the Windows registry.

## More About

- “Install and Register SimMechanics Link Software”

**See Also**

smlink\_linkinv

# smlink\_unlinksw

Unlink SimMechanics Link software as SolidWorks add-in

## Syntax

```
smlink_unlinksw
```

## Description

smlink\_unlinksw unlinks SimMechanics Link software as an add-in to SolidWorks.

## Output Arguments

A message indicating that the unlinking has worked, with the location of the add-in module, if unlinking succeeds.

An error message describing the failure, if unlinking does not succeed.

## Definitions

*Linking* is associating a version of a CAD platform with a SimMechanics Link installation.

Depending on the CAD platform and if you use the Windows operating system, linking a CAD platform can involve registering one of the executable SimMechanics Link libraries with Windows.

*Registering* is entering an executable module or library in the Windows registry.

## More About

- “Install and Register SimMechanics Link Software”

**See Also**

smlink\_linksw





# API — Alphabetical List

---

## **pmit\_add\_cadcs**

Add coordinate system to handle object of PmitCadModelH class

### **Description**

PmitError = pmit\_add\_cadcs(PmitCadModelH pmitCadModelH, PmitCadCSH pmitCadCSH) returns an error status PmitError.

With pmit\_add\_cadcs, you can add a coordinate system to a handle object of PmitCadModelH class that represents an API CAD model.

### **Input Arguments**

#### **pmitCadModelH**

Handle object of PmitCadModelH class representing an API CAD model

**Default:**

#### **pmitCadCSH**

Handle object of PmitCadCSH class representing a coordinate system on an API CAD model

**Default:**

### **See Also**

PmitCadCSH | pmit\_create\_cadcs | PmitError

## pmit\_add\_constrain

Add constraint to handle object of PmitCadModelH class

### Syntax

```
PmitError = pmit_add_constrain(PmitCadModelH pmitCadModelH,  
PmitConstrainH pmitConstrainH)
```

### Description

PmitError = pmit\_add\_constrain(PmitCadModelH pmitCadModelH, PmitConstrainH pmitConstrainH) returns an error status PmitError.

With pmit\_add\_constrain, you can add a constraint to a handle object of PmitCadModelH class that represents an API CAD model.

### Input Arguments

#### **pmitCadModelH**

Handle object of PmitCadModelH class representing an API CAD model

**Default:**

#### **pmitConstrainH**

Handle object of PmitConstrainH class representing an API CAD model constraint

**Default:**

### See Also

PmitCadModelH | PmitConstrainH | PmitError

## **pmit\_add\_refincadmodel**

Add object of PmitCadModelRefH class to object of PmitCadModelH class

### **Syntax**

```
PmitError = pmit_add_refincadmodel(PmitCadModelH pmitCadModelH,  
PmitCadModelRefH pmitCadModelrefH)
```

### **Description**

PmitError = pmit\_add\_refincadmodel(PmitCadModelH pmitCadModelH, PmitCadModelRefH pmitCadModelrefH) returns an error status PmitError.

With pmit\_add\_refincadmodel, you can add an object of PmitCadModelRefH class to an object of PmitCadModelH class, in order to reference a CAD model in an API CAD model hierarchy.

### **Input Arguments**

#### **pmitCadModelH**

Handle object of PmitCadModelH class representing an API CAD model

**Default:**

#### **pmitCadModelrefH**

Handle object of PmitCadModelRefH class referencing a CAD model in an API CAD model hierarchy

**Default:**

### **See Also**

PmitCadModelH | PmitCadModelRefH | PmitError

## pmit\_add\_refincomp

Add object of PmitCadModelRefH class at end of object of PmitAssemCompH class

### Syntax

```
PmitError = pmit_add_refincomp(PmitAssemCompH pmitAssemComp,  
PmitCadModelRefH pmitCadModelrefH)
```

### Description

PmitError = pmit\_add\_refincomp(PmitAssemCompH pmitAssemComp, PmitCadModelRefH pmitCadModelrefH) returns an error status PmitError.

With pmit\_add\_refincomp, you can add an object of PmitCadModelRefH class at the end of an object of PmitAssemCompH class, in order to reference an element in an API CAD hierarchy. You construct the full reference with a chain of objects of PmitCadModelRefH class. Make the chain as long as needed to reach the desired element in the hierarchy.

### Input Arguments

#### **pmitAssemComp**

Handle object of PmitAssemCompH class representing a component in an API CAD model

**Default:**

#### **pmitCadModelrefH**

Handle object of PmitCadModelRefH class referencing a CAD model in an API CAD model hierarchy

**Default:**

### See Also

PmitAssemCompH | PmitCadModelRefH | PmitError

## PmitAssemCompH

Handle object type to represent component in API CAD model

### Description

PmitAssemCompH is a C language opaque type.

A variable of this type is a handle object created when you instantiate a SimMechanics Link API object representing an API CAD assembly component.

### See Also

`pmit_add_refincomp` | `pmit_create_assemcomp` |  
`pmit_create_assemcomp_fromstr` | `pmit_create_constrain` | `PmitError`

# PmitCad2SMH

Handle object type to represent API-to-XML translator

## Description

PmitCad2SMH is a C language opaque type.

A variable of this type is a handle object created when you instantiate a SimMechanics Link API object that translates an API CAD model into XML.

## See Also

`pmit_create_cad2sm` | `PmitError` | `pmit_set_tolerances` | `pmit_write_xml`

## PmitCadCSH

Handle object type to represent coordinate system

### Description

PmitCadCSH is a C language opaque type.

A variable of this type is a handle object created when you add a coordinate system to a SimMechanics Link API object representing an API CAD model.

### See Also

`pmit_add_cadcs` | `pmit_create_cadcs` | `PmitError`



# PmitCadModelH

Handle object type to represent API CAD model

## Description

PmitCadModelH is a C language opaque type.

A variable of this type is a handle object created when you instantiate a SimMechanics Link API object representing an API CAD model of an assembly or assembly part.

## See Also

`pmit_add_constrain` | `pmit_add_refincadmodel` |  
`pmit_cadmodel_setfilename` | `pmit_cadmodelref_getcadmodel`  
| `pmit_create_assemcomp_fromstr` | `pmit_create_cad2sm` |  
`pmit_create_cadmodel` | `pmit_create_cadmodelref` | `PmitError`

## PmitCadModelRefH

Handle object type to reference a CAD model in API CAD model hierarchy

### Description

PmitCadModelRefH is a C language opaque type.

A variable of this type is a handle object created when you instantiate a SimMechanics Link API object referencing an API CAD model component.

### See Also

`pmit_add_refincadmodel` | `pmit_add_refincomp` |  
`pmit_cadmodelref_getcadmodel` | `pmit_create_cadmodelref` |  
`PmitError` | `pmit_get_reffixedstatus` | `pmit_get_refflexiblestatus` |  
`pmit_set_reffixedstatus` | `pmit_set_refflexiblestatus`

# pmit\_cadmodel\_setfilename

Specify body geometry file name for handle object of PmitCadModelH class

## Syntax

```
PmitError = pmit_cadmodel_setfilename(PmitCadModelH pmitCadModelH,  
const char* fileName)
```

## Description

PmitError = pmit\_cadmodel\_setfilename(PmitCadModelH pmitCadModelH, const char\* fileName) returns an error status PmitError.

With pmit\_cadmodel\_setfilename, specify the STL body geometry file name for a handle object of PmitCadModelH class representing an API CAD model.

The body geometry file carries no units. This body geometry is interpreted with the units defined for the API session.

## Input Arguments

### **pmitCadModelH**

Handle object of PmitCadModelH class representing an API CAD model

### **Default:**

### **fileName**

String specifying STL body geometry file name

### **Default:**

## See Also

PmitCadModelH | PmitError | pmit\_set\_units

## **pmit\_cadmodelref\_getcadmodel**

Get object of PmitCadModelH class from children of object of PmitCadModelRefH class

### **Syntax**

```
PmitError = pmit_cadmodelref_getcadmodel(PmitCadModelH*  
pmitCadModelHOut, PmitCadModelRefH cadModelRefH)
```

### **Description**

PmitError = pmit\_cadmodelref\_getcadmodel(PmitCadModelH\* pmitCadModelHOut, PmitCadModelRefH cadModelRefH) returns an error status PmitError.

With pmit\_cadmodelref\_getcadmodel, you can get an object of PmitCadModelH class that represents an API CAD model from whatever is referenced by an object of PmitCadModelRefH class.

### **Input Arguments**

#### **cadModelRefH**

Handle object of PmitCadModelRefH class referencing a CAD model in an API CAD model hierarchy

**Default:**

### **Output Arguments**

#### **pmitCadModelHOut**

Handle object of PmitCadModelH class representing an API CAD model

### **See Also**

PmitCadModelH | PmitCadModelRefH | PmitError

# **pmit\_connectto\_matlab**

Connect to MATLAB session

## **Syntax**

```
PmitError = pmit_connectto_matlab()
```

## **Description**

PmitError = pmit\_connectto\_matlab() returns an error status PmitError.

## **See Also**

pmit\_disconnectfrom\_matlab | PmitError | pmit\_open\_demo |  
pmit\_open\_help

## PmitConstrainH

Handle object type to represent constraint

### Description

PmitConstrainH is a C language opaque type.

A variable of this type is a handle object created when you add a constraint to a SimMechanics Link API object representing an API CAD model.

### See Also

`pmit_add_constrain` | `PmitConstrainType` | `pmit_create_constrain` | `PmitError`

# PmitConstrainType

Enumerated type for specifying constraint type

## Description

PmitConstrainType is a C language enumerated type.

A variable of this type is defined when you create a constraint in a SimMechanics Link API CAD model.

These are the variable's allowed enumerated values.

Value	Constraint Type
PMIT_CON_UNKNOWN = -1	Unknown
PMIT_CON_COINCIDENT = 0	Coincident points
PMIT_CON_CONCENTRIC	Concentric circles or circular arcs
PMIT_CON_PERPEND	Perpendicular lines or planes
PMIT_CON_PARALLEL	Parallel lines or planes
PMIT_CON_TANGENT	Tangent curves or surfaces
PMIT_CON_DISTANCE	Fixed distance between points
PMIT_CON_ANGLE	Fixed angle between lines
PMIT_CON_FULL	Fully fixing one body's position and orientation with respect to another body. Kinematically equivalent to a rigid weld.

## See Also

[pmit\\_add\\_constrain](#) | [PmitConstrainH](#) | [pmit\\_create\\_constrain](#) | [PmitError](#)

## **pmit\_create\_assemcomp**

Create object of PmitAssemCompH class

### **Syntax**

```
PmitError = pmit_create_assemcomp(PmitAssemCompH* const  
pmitAssemCompHOut)
```

### **Description**

PmitError = pmit\_create\_assemcomp(PmitAssemCompH\* const pmitAssemCompHOut) returns an error status PmitError.

With pmit\_create\_assemcomp, you can create an object of PmitAssemCompH class in order to reference child models in the hierarchy of other API CAD models.

### **Output Arguments**

#### **pmitAssemCompHOut**

Handle object of PmitAssemCompH class representing a component in an API CAD model

### **See Also**

PmitAssemCompH | PmitError



# pmit\_create\_assemcomp\_fromstr

Create object of PmitAssemCompH class

## Syntax

## Description

PmitError = pmit\_create\_assemcomp\_fromstr(PmitAssemCompH\* const pmitAssemCompHOut, const char\* compName, PmitCadModelH parentModelH)  
returns an error status PmitError.

With pmit\_create\_assemcomp\_fromstr, you can create, from its string representation, an object of PmitAssemCompH class that represents an API CAD model component.

## Input Arguments

### compName

String specifying name of component

**Default:**

### parentModelH

Handle object of class PmitCadModelH representing an API CAD model

**Default:**

## Output Arguments

### pmitAssemCompHOut

Handle object of PmitAssemCompH class representing a component in an API CAD model

**See Also**

PmitAssemCompH | PmitCadModelH | PmitError

# pmit\_create\_cad2sm

Create object of PmitCad2SMH class

## Syntax

```
PmitError = pmit_create_cad2sm(PmitCad2SMH* const pmitCad2SMHOut,  
PmitCadModelH const pmitCadModelH, const char* createdUsing, const  
char* createdFrom, const char* createdOn, const char* createdBy,  
const char* name)
```

## Description

```
PmitError = pmit_create_cad2sm(PmitCad2SMH* const pmitCad2SMHOut,  
PmitCadModelH const pmitCadModelH, const char* createdUsing, const  
char* createdFrom, const char* createdOn, const char* createdBy,  
const char* name) returns an error status PmitError.
```

With `pmit_create_cad2sm`, you can create an object of `PmitCad2SMH` class to represent an API-to-XML CAD model translator. The header information that you specify in the inputs is written to the final XML file.

## Input Arguments

### **pmitCadModelH**

Handle object of `PmitCadModelH` class representing an API CAD model

**Default:**

### **createdUsing**

String naming the exporter

**Default:**

**createdFrom**

String naming the source CAD platform or other external application

**Default:**

**createdOn**

String specifying date that the object was created

**Default:**

**createdBy**

String specifying name of user creating the object

**Default:**

**name**

String naming the assembly model

**Default:**

## Output Arguments

**pmitCad2SMHOut**

Handle object of PmitCad2SMH class representing an API-to-XML translator object

### See Also

PmitCad2SMH | PmitCadModelH | PmitError | pmit\_write\_xml

### Tutorials

- “A Custom Exporter Module Example”

## pmit\_create\_cadcs

Create object of PmitCadCSH class

### Syntax

```
PmitError = pmit_create_cadcs(PmitCadCSH* const pmitCadCSHOut, const char* name, const char* nodeID, double rotation[9], double trans[3])
```

### Description

PmitError = pmit\_create\_cadcs(PmitCadCSH\* const pmitCadCSHOut, const char\* name, const char\* nodeID, double rotation[9], double trans[3]) returns an error status PmitError.

With pmit\_create\_cadcs, you can create an object of PmitCadCSH class to represent a coordinate system in an API CAD model.

### Input Arguments

#### **name**

String naming the coordinate system

#### **Default:**

#### **nodeID**

String uniquely identifying the coordinate system for associativity purposes

#### **Default:**

#### **rotation**

Double-type real rotation 9-vector specifying rotational transformation of the origin of this coordinate system with respect to its parent CAD model.

#### **Default:**

**trans**

Double-type real 3-vector specifying translation of the origin of this coordinate system with respect to its parent CAD model.

**Default:**

## Output Arguments

**pmitCadCSHout**

Handle object of PmitCadCSH class representing a coordinate system in an API CAD model

## More About

**Orthogonal Matrix**

A matrix  $R$  is orthogonal if it satisfies the matrix multiplication rule  $R^T * R = R * R^T = I$ , where  $I$  is the identity matrix.

**Rotational Transformation: Rotation Matrix and Rotation Vector**

The rotation vector input is a 9-vector, defined from the 3-by-3 orthogonal rotation matrix  $R$ , that represents the rotational orientation of a CAD model component with respect to its parent CAD model.

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}.$$

You define the rotation 9-vector column-wise:

```
rotation = [R(1,1) R(2,1) R(3,1) R(1,2) R(2,2) R(3,2) ...  
           ... R(1,3) R(2,3) R(3,3)]
```

**See Also**

pmit\_add\_cadcs | PmitCadCSH | PmitError

# pmit\_create\_cadmodel

Create object of PmitCadModelH class

## Syntax

```
PmitError = pmit_create_cadmodel(PmitCadModelH* const  
pmitCadModelHOut, const char* name, double mass, const double  
inertia[6], const double cg[3], double volume, double sarea, const  
char* fileName, const PmitVisMatProp* matprops)
```

## Description

PmitError = pmit\_create\_cadmodel(PmitCadModelH\* const pmitCadModelHOut, const char\* name, double mass, const double inertia[6], const double cg[3], double volume, double sarea, const char\* fileName, const PmitVisMatProp\* matprops) returns an error status PmitError.

With pmit\_create\_cadmodel, you can create an object of PmitCadModelH class to represent an API CAD model.

The body geometry file specified by **fileName** carries no units. This body geometry is interpreted with the units defined for the API session.

## Input Arguments

### **name**

String naming the CAD assembly or part model

### **Default:**

### **mass**

Double-type real number specifying the mass of the assembly or part

### **Default:**

**inertia**

Double-type real 6-vector specifying the rotational inertia of the assembly or part. See “Definitions” on page 5- .

**Default:**

**cg**

Double-type real 3-vector specifying the position of the center of gravity of the assembly or part

**Default:**

**volume**

Double-type real number specifying the volume of the assembly or part

**Default:**

**sarea**

Double-type real number specifying the surface area of the assembly or part

**Default:**

**fileName**

String specifying STL body geometry file name

**Default:**

**matprops**

Structure of PmitVisMatProp class specifying the visualizable properties of the assembly or part

**Default:**

## Output Arguments

**pmitCadModelHOut**

Handle object of PmitCadModelH class representing an API CAD model



## More About

### Inertia Tensor and Inertia Vector

The inertia vector input is a 6-vector defined from the 3-by-3 symmetric inertia tensor  $I$  that depends on the part's mass distribution:

$$I = \begin{pmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{pmatrix},$$

where  $I_{21} = I_{12}$ ,  $I_{31} = I_{13}$ , etc.

You define the inertia 6-vector as:

```
inertia = [I(1,1) I(2,2) I(3,3) I(1,2) I(3,1) I(2,3)]
```

### See Also

[PmitCadModelH](#) | [PmitError](#) | [pmit\\_set\\_units](#) | [PmitVisMatProp](#)

## **pmit\_create\_cadmodelref**

Create object of PmitCadModelRefH class

### **Syntax**

```
PmitError = pmit_create_cadmodelref(PmitCadModelRefH* const  
pmitCadModelRefHOut, const char* name, const char* nodeID,  
PmitCadModelH pmitCadModelH, double rotation[9], double trans[3],  
double scale, int isFlexible, int isFixed, const PmitVisMatProp*  
matprops)
```

### **Description**

```
PmitError = pmit_create_cadmodelref(PmitCadModelRefH* const  
pmitCadModelRefHOut, const char* name, const char* nodeID,  
PmitCadModelH pmitCadModelH, double rotation[9], double trans[3],  
double scale, int isFlexible, int isFixed, const PmitVisMatProp*  
matprops) returns an error status PmitError.
```

With `pmit_create_cadmodelref`, you can create an object of `PmitCadModelRefH` class to reference a CAD model in an API CAD model hierarchy.

### **Input Arguments**

#### **name**

String specifying name of component instance

#### **Default:**

#### **nodeID**

String specifying unique identity of model component within parent hierarchy. This identity must be unique within the full model.

**Default:**

**pmitCadModelH**

Handle object of PmitCadModelH class representing an API CAD model. This is the same model referenced by the output object pmitCadModelRefHOut, an object of PmitCadModelRefH class.

**Default:**

**rotation**

Double-type real rotation 9-vector specifying rotational transformation of the origin of this CAD model with respect to its parent CAD model. See “Definitions”.

**Default:**

**trans**

Double-type real 3-vector specifying translation of the origin of this CAD model with respect to its parent CAD model.

**Default:**

**scale**

Double-type real number specifying overall length scaling of this instance of the model. A value of 1 means no overall scaling.

**Default:**

**isFlexible**

Integer-type flag specifying whether component is rigid or nonrigid. A value of 0 means the component is rigid; a value of 1 means the component is nonrigid.

**Default:**

**isFixed**

Integer-type flag specifying whether component is welded or not to its attachment point in the assembly. A value of 0 means the component is not welded; a value of 1 means the component is welded. See “Definitions” on page 5-

**Default:**

### **matprops**

Structure of type `PmitVisMatProp` for defining visualized material properties of the machine

**Default:**

## Output Arguments

### **pmitCadModelRefHOut**

Handle object of `PmitCadModelRefH` class referencing a CAD model in an API CAD model hierarchy

## More About

### **Orthogonal Matrix**

A matrix  $R$  is orthogonal if it satisfies the matrix multiplication rule  $R^T * R = R * R^T = I$ , where  $I$  is the identity matrix.

### **Rotational Transformation: Rotation Matrix and Rotation Vector**

The rotation vector input is a 9-vector, defined from the 3-by-3 orthogonal rotation matrix  $R$ , that represents the rotational orientation of a CAD model component with respect to its parent CAD model.

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}.$$

You define the rotation 9-vector column-wise:

```
rotation = [R(1,1) R(2,1) R(3,1) R(1,2) R(2,2) R(3,2) ...  
           ... R(1,3) R(2,3) R(3,3)]
```

**Flexible Model**

A flexible or nonrigid model is made of components that can move with respect to one another.

An inflexible or rigid model is made of components that cannot move with respect to one another.

**Fixed Model**

A fixed model cannot move relative to the ground of the assembly model.

A nonfixed model can move relative to the ground of the assembly hierarchy.

**See Also**

`PmitCadModelH` | `PmitCadModelRefH` | `PmitError` | `PmitVisMatProp`

## **pmit\_create\_constrain**

Create object of PmitConstrainH class

### **Syntax**

### **Description**

```
PmitError = pmit_create_constrain(PmitConstrainH* const  
pmitConstrainhOut, const char* name, PmitConstrainType type,  
PmitAssemCompH body1Comp, PmitAssemCompH body2Comp, PmitGeomType  
body1Type, PmitGeomType body2Type, const double body1Loc, const  
double body1Axis, const double body2Loc, const double body2Axis)  
returns an error status PmitError.
```

With `pmit_create_constrain`, you can create an object of `PmitConstrainH` class to represent a constraint in an API CAD model.

### **Input Arguments**

For a complete specification of these inputs, see “Definitions” on page 5- .

**name**

String naming the constraint

**Default:**

**type**

Handle object of `PmitConstrainType` class to represent constraint type in an API CAD model

**Default:**

**body1Comp**

Handle object of `PmitAssemCompH` class to represent first constrained body in an API CAD model

**Default:**

**body2Comp**

Handle object of `PmitAssemCompH` class to represent second constrained body in an API CAD model

**Default:**

**body1Type**

Handle object of `PmitGeomType` class to represent the geometry of first constrained body in an API CAD model

**Default:**

**body2Type**

Handle object of `PmitGeomType` class to represent the geometry of second constrained body in an API CAD model

**Default:**

**body1Loc**

Double-type 3-vector specifying the spatial location of body 1

**Default:**

**body1Axis**

Double-type 3-vector specifying the spatial orientation of the axis of body 1

**Default:**

**body2Loc**

Double-type 3-vector specifying the spatial location of body 2

**Default:**

**body2Axis**

Double-type 3-vector specifying the spatial orientation of the axis of body 2

**Default:**

## Output Arguments

### **pmitConstrainhOut**

Handle object of PmitConstrainH class to represent a constraint in an API CAD model

## More About

### **Constraint**

A constraint imposes a restriction on how two component bodies can move relative to one another.

You define a constraint by an axis through a point oriented and located, respectively, with respect to body 1.

### **Body Specification**

To impose a constraint, specify the two bodies by their:

- Component handles
- Component body geometry type handles
- Locations in space. The location of body 2 is a translation with respect to the coordinate origin of the CAD model representing body 1.
- Axis directions in space . The axis of body 2 is a direction with respect to the coordinate axes of the CAD model representing body 1.

### **See Also**

PmitAssemCompH | PmitConstrainH | PmitConstrainType | PmitError | PmitGeomType



# **pmit\_disconnectfrom\_matlab**

Disconnect from MATLAB session

## **Syntax**

```
PmitError = pmit_disconnectfrom_matlab()
```

## **Description**

PmitError = pmit\_disconnectfrom\_matlab() returns an error status PmitError.

## **See Also**

pmit\_connectto\_matlab | PmitError

## PmitError

Enumerated type for error status

### Description

PmitError is a C language enumerated type.

A variable of this type is defined whenever you call any SimMechanics Link API function.

These are the variable's allowed enumerated values.

Value	Error Type
PMIT_NO_ERROR = 0	No error
PMIT_GENERIC_FAIL	Function call failure not otherwise specified
PMIT_CAD_MODEL_NOTSET	API representation of machine not defined
PMIT_XML_DOM_ERROR	XML error
PMIT_UNHANDLED_CONSTRAIN	Constraint translation error
PMIT_INVALID_CON_COMPS	
PMIT_UNSUPPORTED_INERTIA_UNIT	Mass or inertia unit specified that is not supported by API
PMIT_COULDNOT_CONNECTTO_MATLAB	Failure to connect to MATLAB

# PmitGeomType

Enumerated type for specifying geometry of component

## Description

PmitGeomType is a C language enumerated type.

A variable of this type is defined when you create a component in a SimMechanics Link API CAD model.

These are the variable's allowed enumerated values.

Value	Geometry Type
PMIT_GEO_UNKNOWN = -1	Unknown
PMIT_GEO_POINT = 0	Point
PMIT_GEO_LINE	Line
PMIT_GEO_PLANE	Plane
PMIT_GEO_CYL	Cylinder
PMIT_GEO_CONE	Cone
PMIT_GEO_CIRCLE	Circle

## See Also

`pmit_create_constrain` | `PmitError`

## **pmit\_get\_reffixedstatus**

Get fixed status of CAD model

### **Syntax**

```
PmitError = pmit_get_reffixedstatus(int* fixedstatusOut, const PmitCadModelRefH cadModelRefH)
```

### **Description**

`PmitError = pmit_get_reffixedstatus(int* fixedstatusOut, const PmitCadModelRefH cadModelRefH)` returns an error status `PmitError`.

With `pmit_get_reffixedstatus`, you can get the fixed status of a CAD model referenced by an object of `PmitCadModelRefH` class.

### **Input Arguments**

#### **cadModelRefH**

Handle object of `PmitCadModelRefH` class referencing a CAD model in an API CAD model hierarchy

**Default:**

### **Output Arguments**

#### **fixedstatusOut**

Integer flag indicating if the model is fixed or not. A value of 0 means the model is not fixed. A value of 1 means the model is fixed. See “Definitions” on page 5- .

## More About

### Fixed Model

A fixed model cannot move relative to the ground of the assembly model.

A nonfixed model can move relative to the ground of the assembly hierarchy.

### See Also

[PmitCadModelRefH](#) | [PmitError](#)

## **pmit\_get\_refflexiblestatus**

Get flexible status of CAD model

### **Syntax**

```
PmitError = pmit_get_refflexiblestatus(int* flexstatusOut, const  
PmitCadModelRefH cadModelRefH)
```

### **Description**

PmitError = pmit\_get\_refflexiblestatus(int\* flexstatusOut, const PmitCadModelRefH cadModelRefH) returns an error status PmitError.

With pmit\_get\_refflexiblestatus, you can get the flexible status of a CAD model referenced by an object of PmitCadModelRefH class.

### **Input Arguments**

#### **cadModelRefH**

Handle object of PmitCadModelRefH class referencing a CAD model in an API CAD model hierarchy

**Default:**

### **Output Arguments**

#### **flexstatusOut**

Integer flag indicating if the model is flexible or not. A value of 0 means the model is inflexible, or rigid. A value of 1 means the model is flexible, or nonrigid. “Definitions” on page 5-

## More About

### Flexible Model

A flexible or nonrigid model is made of components that can move with respect to one another.

An inflexible or rigid model is made of components that cannot move with respect to one another.

### See Also

[PmitCadModelRefH](#) | [PmitError](#)

## PmitLengthUnit

Enumerated type for specifying length unit in API session

### Description

PmitLengthUnit is a C language enumerated type.

You can define a variable of this type globally when you start a SimMechanics Link API session.

These are the variable's allowed enumerated values.

Value	Length Unit Type
PMIT_LU_UNKNOWN = -1	Unknown
PMIT_LU_M = 0	Meter
PMIT_LU_CM	Centimeter
PMIT_LU_MM	Millimeter
PMIT_LU_KM	Kilometer
PMIT_LU_IN	Inch
PMIT_LU_FT	Foot
PMIT_LU_MI	Mile
PMIT_LU_YD	Yard

### See Also

PmitError | PmitMassUnit | pmit\_set\_units



# PmitMassUnit

Enumerated type for specifying mass unit in API session

## Description

PmitMassUnit is a C language enumerated type.

You can define a variable of this type globally when you start a SimMechanics Link API session.

These are the variable's allowed enumerated values.

Value	Mass Unit Type
PMIT_MU_UNKNOWN = -1	Unknown
PMIT_MU_KG = 0	Kilogram
PMIT_MU_G	Gram
PMIT_MU_MG	Milligram
PMIT_MU_LBM	Pound (mass)
PMIT_MU_OZ	Ounce
PMIT_MU_SLUG	Slug

## See Also

PmitError | PmitLengthUnit | pmit\_set\_units

## PmitObjectH

Handle object type to represent any API object

### Description

PmitObjectH is a C language opaque type.

You can define a variable of this type for any object created by the SimMechanics Link API.

### See Also

PmitError | pmit\_release\_object

## **pmit\_open\_demo**

Open SimMechanics Link examples in MATLAB Help browser

### **Syntax**

```
PmitError = pmit_open_demo()
```

### **Description**

`PmitError = pmit_open_demo()` returns an error status `PmitError`.

This function is equivalent to the MATLAB command:

```
demo('matlab','simmechanics link')
```

### **See Also**

`demo` | `pmit_connectto_matlab` | `PmitError` | `pmit_open_help`

## **pmit\_open\_help**

Open product documentation in MATLAB Help browser

### **Syntax**

```
PmitError = pmit_open_help(const char* helpItem)
```

### **Description**

`PmitError = pmit_open_help(const char* helpItem)` returns an error status `PmitError`.

This function causes MATLAB to issue the command:

```
doc StringValue
```

*StringValue* is the value of the string `helpItem`.

### **Input Arguments**

#### **helpItem**

String specifying the product documentation item to display in the MATLAB Help browser

**Default:**

#### **See Also**

`doc` | `pmit_connectto_matlab` | `PmitError` | `pmit_open_demo`

## **pmit\_release\_buffer**

Release character buffer returned by API function

### **Syntax**

```
PmitError = pmit_release_buffer(char** buffer)
```

### **Description**

PmitError = pmit\_release\_buffer(char\*\* buffer) returns an error status PmitError.

### **Input Arguments**

#### **buffer**

String specifying the name of the buffer that you want to release

**Default:**

#### **See Also**

PmitError | pmit\_release\_object

## **pmit\_release\_object**

Release object used by API session

### **Syntax**

```
PmitError = pmit_release_object(PmitObjectH objectH)
```

### **Description**

PmitError = pmit\_release\_object(PmitObjectH objectH) returns an error status PmitError.

### **Input Arguments**

#### **objectH**

Handle object of PmitObjectH class representing the API CAD object that you want to release

**Default:**

#### **See Also**

PmitError | PmitObjectH | pmit\_release\_buffer

# pmit\_set\_reffixedstatus

Set fixed status of CAD model

## Syntax

```
PmitError = pmit_set_reffixedstatus(PmitCadModelRefH cadModelRefH,  
int status)
```

## Description

PmitError = pmit\_set\_reffixedstatus(PmitCadModelRefH cadModelRefH, int status) returns an error status PmitError.

With pmit\_set\_reffixedstatus, you can set the fixed status of a CAD model referenced by an object of PmitCadModelRefH class.

## Input Arguments

### cadModelRefH

Handle object of PmitCadModelRefH class referencing a CAD model in an API CAD model hierarchy

**Default:**

### status

Integer flag indicating if the model is fixed or not. A value of 0 means the model is not fixed. A value of 1 means the model is fixed. “Definitions” on page 5-

**Default:**

## More About

### Fixed Model

A fixed model cannot move relative to the ground of the assembly hierarchy.

A nonfixed model can move relative to the ground of the assembly hierarchy.

**See Also**

`PmitCadModelRefH` | `PmitError`



# pmit\_set\_refflexiblestatus

Set flexible status of CAD model

## Syntax

```
PmitError = pmit_set_refflexiblestatus(PmitCadModelRefH  
cadModelRefH, int status)
```

## Description

PmitError = pmit\_set\_refflexiblestatus(PmitCadModelRefH  
cadModelRefH, int status) returns an error status PmitError.

With pmit\_set\_refflexiblestatus, you can set the flexible status of a CAD model referenced by an object of PmitCadModelRefH class.

## Input Arguments

### cadModelRefH

Handle object of PmitCadModelRefH class referencing a CAD model within an API CAD model hierarchy

### Default:

### status

Integer flag indicating if the model is flexible or not. A value of 0 means the model is inflexible, or rigid. A value of 1 means the model is flexible, or nonrigid. “Definitions” on page 5-

### Default:

## **More About**

### **Flexible Model**

A flexible or nonrigid model is made of components that can move with respect to one another.

An inflexible or rigid model is made of components that cannot move with respect to one another.

### **See Also**

`PmitCadModelRefH` | `PmitError`

## pmit\_set\_tolerances

Set linear, angular, and relative tolerances of object of PmitCad2SMH class

### Syntax

```
PmitError = pmit_set_tolerances(PmitCad2SMH pmitCad2SMH, double  
linearTol, double angularTol, double relativeTol)
```

### Description

PmitError = pmit\_set\_tolerances(PmitCad2SMH pmitCad2SMH, double linearTol, double angularTol, double relativeTol) returns an error status PmitError.

With pmit\_set\_tolerances, you can set the linear, angular, and relative tolerances of an object of PmitCad2SMH class representing an API-to-XML translator.

### Input Arguments

#### **pmitCad2SMH**

Handle object of PmitCad2SMH class representing an API-to-XML translator

#### **linearTol**

Error tolerance when comparing linear alignments and spacings, measured in length unit specified by PmitLengthUnit

**Default:**

#### **angularTol**

Error tolerance when comparing angular alignments and spacings, measured in radians

**Default:**

**relativeTol**

Smallest significant relative numerical difference

**Default:**

**See Also**

PmitCad2SMH | PmitError | PmitLengthUnit | pmit\_set\_units

## pmit\_set\_units

Set units for API session

### Syntax

```
PmitError = pmit_set_units(PmitMassUnit massUnit, PmitLengthUnit lenUnit)
```

### Description

`PmitError = pmit_set_units(PmitMassUnit massUnit, PmitLengthUnit lenUnit)` returns an error status `PmitError`.

With `pmit_set_units`, you can set the units for an API session.

### Input Arguments

#### **massUnit**

Input of enumerated type `PmitMassUnit` specifying the mass unit system

**Default:**

#### **lenUnit**

Input of enumerated type `PmitLengthUnit` specifying the length unit system

**Default:**

### See Also

`PmitError` | `PmitLengthUnit` | `PmitMassUnit` | `pmit_set_tolerances`

## PmitVisMatProp

Structure type for defining visualized material properties of API CAD object

### Description

PmitVisMatProp is a C language structure type.

You can define a variable of this type for any object in a SimMechanics Link API CAD model. This variable specifies the visualized material properties of the object, usually a part component of a CAD assembly.

You refer to the fields of the structure as `PmitVisMatProp.field`. These are the structure fields and their possible values, which all range from 0 to 1.

Field	Values
<code>rgb</code>	3-vector [ <code>r g b</code> ] specifying red, green, and blue color intensities <code>r</code> , <code>g</code> , and <code>b</code>
<code>ambient</code>	Intensity of the ambient component of light falling on the component
<code>diffuse</code>	Intensity of the diffuse component of light falling on the component
<code>specular</code>	Intensity of the specular component of light falling on the component
<code>shininess</code>	Shininess coefficient of the component's material
<code>transparency</code>	Transparency factor of the component's material. 0 means the material is not transparent. 1 means it is fully transparent.
<code>emission</code>	Intensity of emission from the component's material

### See Also

`pmit_create_cadmodel` | `pmit_create_cadmodelref` | `PmitError`

### Related Links

- [OpenGL resources on visualized lighting and material properties](#)

# pmit\_write\_xml

Write output of object of PmitCad2SMH class

## Syntax

```
PmitError = mit_write_xml(char** const pconstrainErrorOut,  
PmitCad2SMH pmitCad2SMH, const char* filename)
```

## Description

```
PmitError = mit_write_xml(char** const pconstrainErrorOut,  
PmitCad2SMH pmitCad2SMH, const char* filename) returns an error status  
PmitError.
```

With `pmit_write_xml`, you can write the output of an object of `PmitCad2SMH` class to a Physical Modeling XML file.

## Input Arguments

### **pmitCad2SMH**

Handle object of `PmitCad2SMH` class representing an API-to-XML translator object

**Default:**

### **filename**

String specifying the name of the XML file to which the API representation is written

**Default:**

## Output Arguments

### **pconstrainErrorOut**

String indicating constraint errors, if any, encountered while writing the XML file

### **See Also**

PmitCad2SMH | pmit\_create\_cad2sm | PmitError

### **Tutorials**

- “A Custom Exporter Module Example”