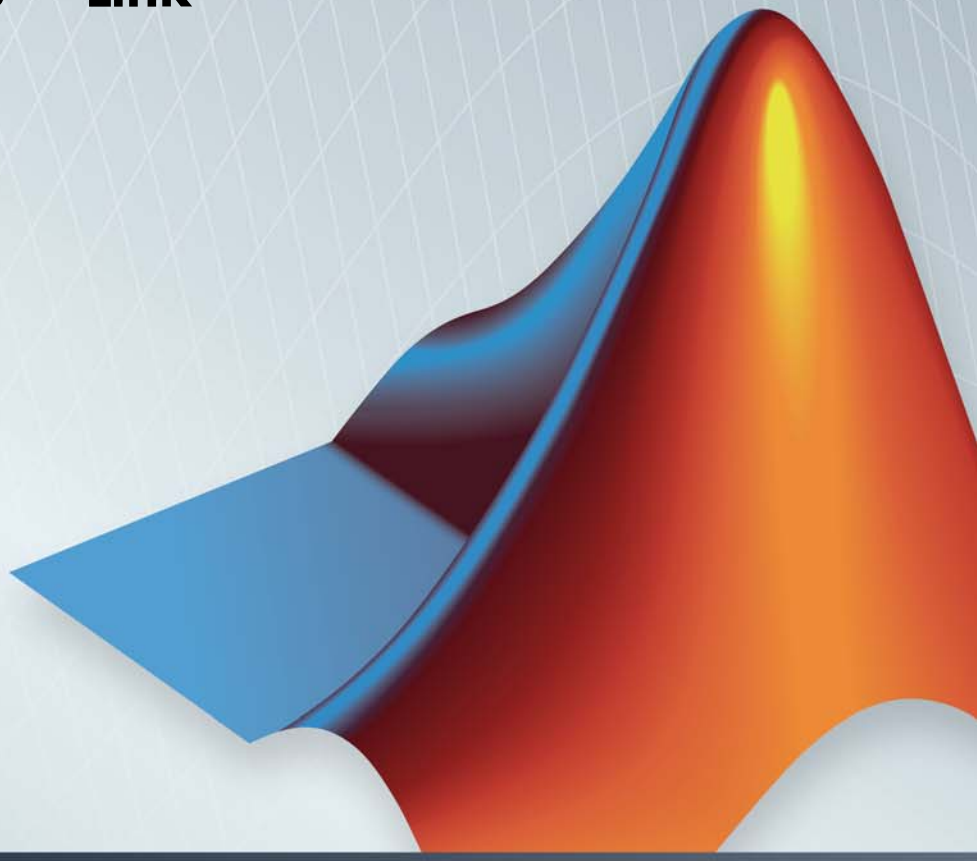
SimMechanics™ Link

Reference

R2014**a**



MATLAB®

MathWorks®

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

SimMechanics™ Link Reference

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `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` 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)

Contents

1 Register and Use the Inventor Add-In

Register and Use the Creo Add-In

2

Register and Use SolidWorks Add-In

3

1

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

In this section...

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

In this section...

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

Entity	Description
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**
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**						
	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...

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**
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**						
	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
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

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:

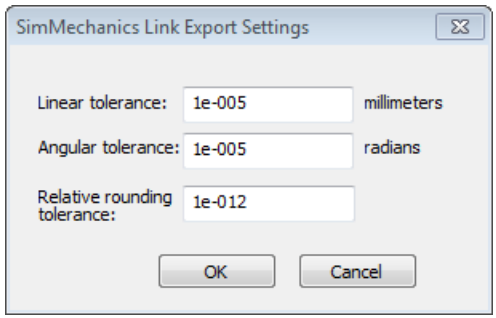
1. Open the assembly to export.
2. In the menu bar, click **Add-Ins > Settings**.

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.

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

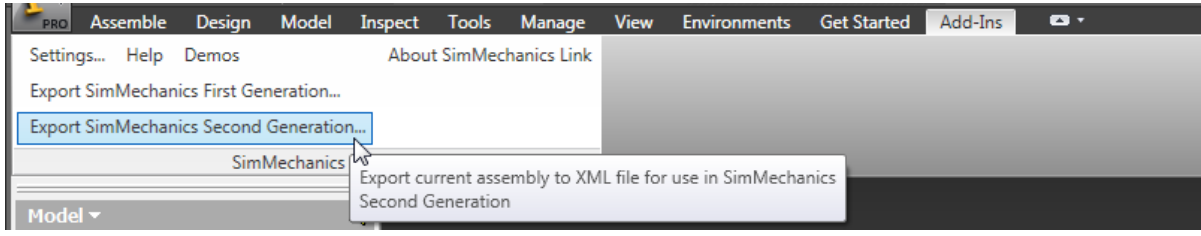
Export CAD Assembly from Autodesk Inventor

In this section...

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.

2

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

In this section...

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

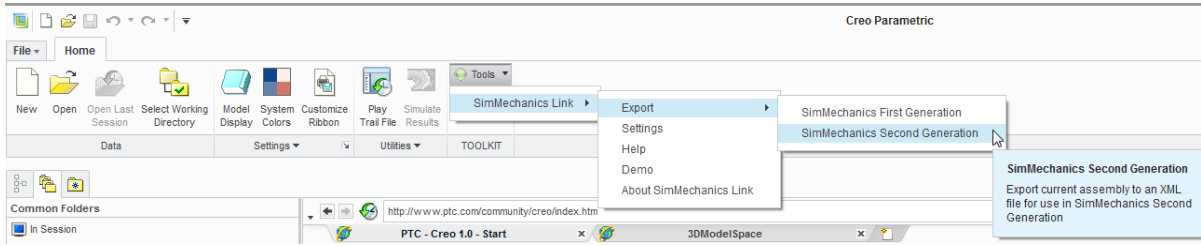
The complete registration procedure contains two steps:

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/cl_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 test to an existing registry file	Delete the registration text from the registry file

Constraint-Joint Mapping

In this section...

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

Entity	Description
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.

Primitive Combination	SimMechanics Block
P	Prismatic Joint
PP	Rectangular Joint
PPP	Cartesian joint

Primitive Combination	SimMechanics Block
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...

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.

Primitive Combination	SimMechanics Block
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

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:

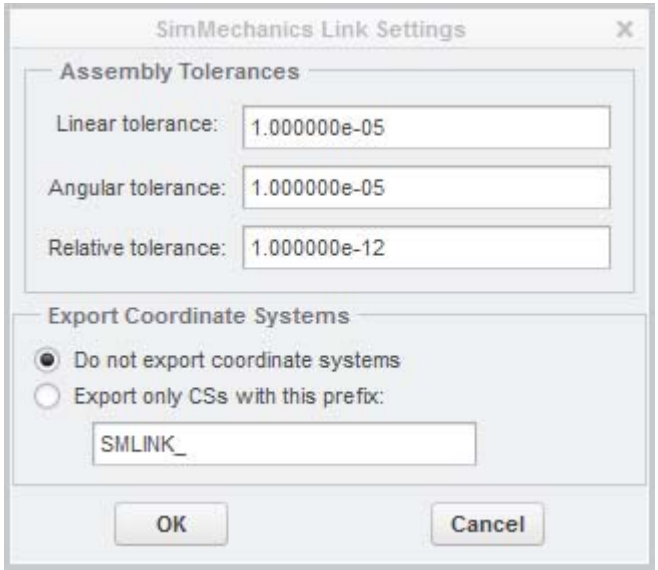
1. Open the assembly to export.
2. In the menu bar, click **Tools > SimMechanics Link**.
3. Click **Settings**.

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
Linear tolerance	1e-005	Smallest significant length difference	1e-5	Units used in assembly
Angular tolerance	1e-005	Smallest significant angle difference	1e-5	Units used in assembly
Relative roundoff tolerance	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.

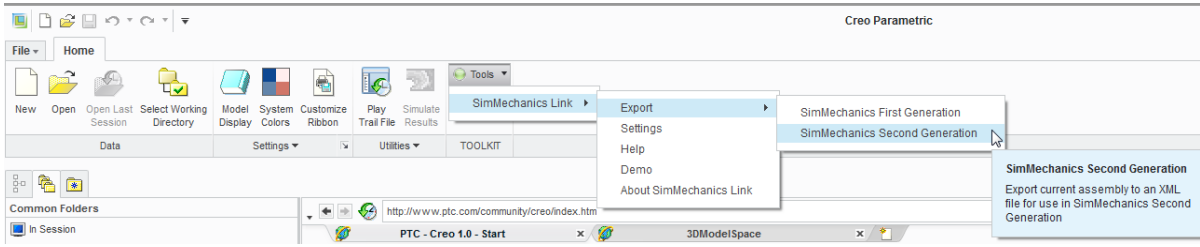
Export CAD Assembly

In this section...

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.

3

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

In this section...

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 Linkadd-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

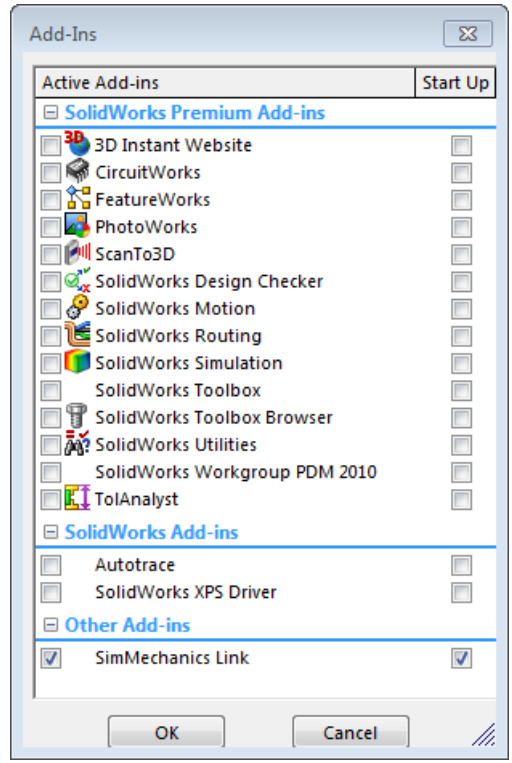
To register SimMechanics Link with the CAD platform:

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

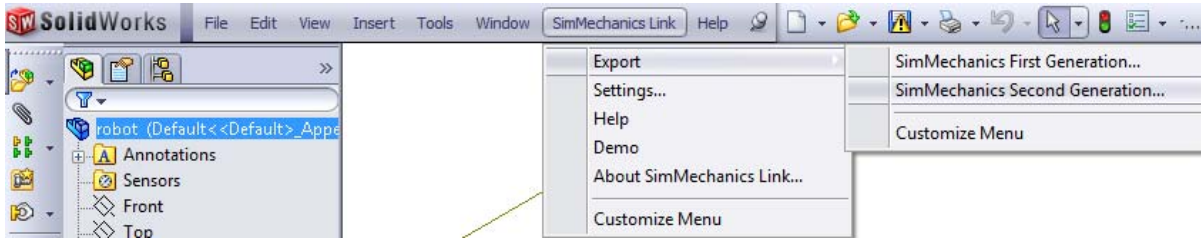
If you have multiple SolidWorks installations on your computer, the command `smlink_linksw` registers SimMechanics Link with *all* installations.

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 Mate Entities

In a SolidWorks assembly, you specify geometric relationships between parts using mates. Each mate applies a kinematic constraint between geometric entities of different parts. Geometric entities include points, lines, and surfaces.

A nut and bolt system provides an example of a mated system. To center the two parts on a common axis, you might add a concentric mate between the cylindrical surfaces of the bolt and the nut hole. This mate restricts motion to translation along the common axis as well as rotation about that same axis.

During CAD Import, SimMechanics converts mates between parts into joints between rigid bodies. In the nut-and-bolt example, SimMechanics might translate the concentric mate between the two cylindrical surfaces into a revolute joint between frames on the nut and bolt rigid bodies.

SimMechanics supports most mates and mate entities in SolidWorks. The supported entities are:

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

The supported mates are:

- 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. For conciseness, the redundant half of the table is omitted.

Mate Entity 1 \ Mate Entity 2	Circle/Arc	Cone	Cylinder	Line	Plane	Point
Circle/Arc	Ct, Cc					
Cone	Ct, Cc	Ct, Cc, Pl				
Cylinder	Ct, Cc	Ct, Cc	Ct, Cc, Pl			
Line	Cc		Cc	A, Ct, Pr, Pl		
Plane	Ct			Ct, D, Pr, Pl	A, Ct, Cc, D, Pr, Pl	
Point					D	Ct, D

A — Angle Cc — Concentric Ct — Coincident
D — Distance Pl — Parallel Pr — Perpendicular

CAD assemblies with other mate-entity combinations may present issues during CAD import into SimMechanics. By default, SimMechanics replaces each unsupported mate with a rigid connection. Parts joined by a rigid connection have zero degrees of freedom with respect to each other. You can replace these rigid connections with the appropriate joints once CAD Import is complete.

SimMechanics Joints

SimMechanics joint and constraint blocks are the functional equivalent of SolidWorks mates. They apply between frames the kinematic relationships that determine how they can move. For example, a revolute joint aligns the Z axes of two frames while keeping their origins coincident. In this configuration, the two frames have one rotational degree of freedom (DoF) about the common Z axis.

Joints are combinations of joint primitives, the building blocks that provide the joint its degrees of freedom. These primitives vary according to the number and type of DoFs that they provide. SimMechanics includes three joint-primitive types:

- Prismatic primitive — Joint primitive with one translational DoF
- Revolute primitive — Joint primitive with one rotational DoF
- Spherical primitive — Joint primitive with three concurrent rotational DoFs

The spherical primitive differs from a set of three revolute primitives in the way it represents 3-D rotation:

- Spherical primitive — Represents rotation in three dimensions as one rotation about an arbitrary 3-D axis.
- Three revolute primitives — Represents rotation in three dimensions as a sequence of three rotations about different axes. During rotation, two axes can become aligned, causing the joint to lose one rotational degree of freedom. This phenomenon, named after the gimbal devices in which they often occur,

The table summarizes the joint primitives present in each SimMechanics joint. In this table, P, R, and S denote prismatic, revolute, and spherical joint primitives. Note that the weld joint, which provides zero degrees of freedom, has no joint primitives.

Joint Block	Joint Primitives
6-DOF Joint	P-P-P-S
Bearing Joint	P-R-R-R

Joint Block	Joint Primitives
Bushing Joint	P-P-P-R-R-R
Cartesian Joint	P-P-P
Cylindrical Joint	P-R
Gimbal Joint	P-P-R
Pin Slot Joint	P-R
Planar Joint	P-P-R
Prismatic Joint	P
Rectangular Joint	P-P
Revolute Joint	R
Spherical Joint	S
Telescoping Joint	P-S
Universal Joint	R-R
Weld Joint	N/A

During CAD Import, SimMechanics translates the mates in a SolidWorks assembly into an equivalent set of joints. For example, a coincident mate between a point-point entity pair maps onto a spherical joint in SimMechanics.

Mate-Joint Mapping in SimMechanics First Generation

In this section...

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**
Mate-Entity 1	**Point**						
	Line		✓	✓			
	Plane		✓	✓			
	Cylinder						
	Cone						
	Circle/Arc						

Parallel Mate

The following table identifies supported constraint-entity combinations for the Parallel 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						

Distance Mate

The following table identifies supported constraint-entity combinations for the Distance 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						

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
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

In this section...

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:

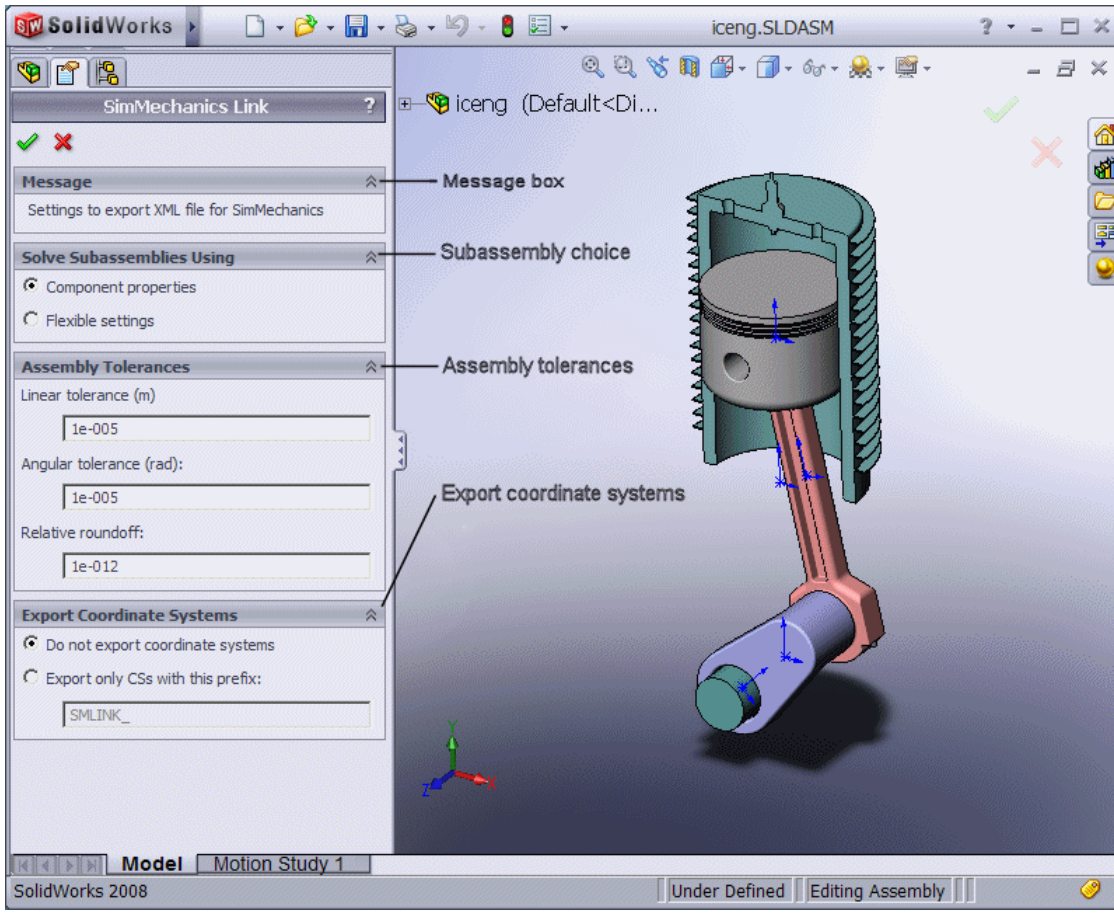
1. Open the assembly to export.
2. In the menu bar, click **SimMechanics Link > Settings**.

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
Linear tolerance	`1e-005`	Smallest significant length difference	1e-5	meter (m)
Angular tolerance	`1e-005`	Smallest significant angle difference	1e-5	radian (rad)
Relative roundoff tolerance	`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.

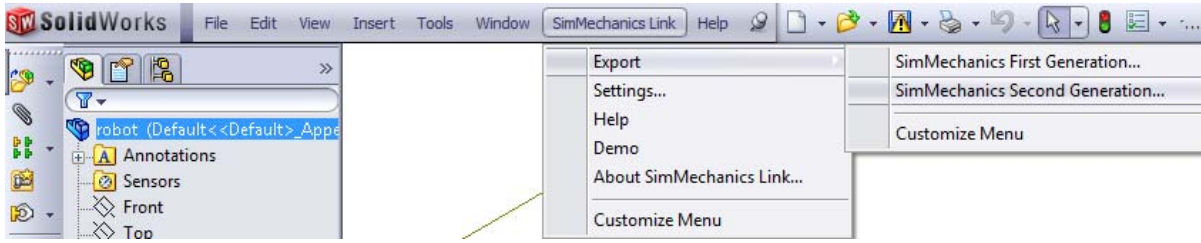
Export CAD Assembly from SolidWorks Software

In this section...

Export CAD Assembly

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.

4

Function Reference

smlink_linkinv

Purpose

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® 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.

See Also

`smlink_unlinkinv`

How To

- "Install and Register SimMechanics Link Software"

Purpose

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.

See Also

`smlink_unlinksw`

How To

- "Install and Register SimMechanics Link Software"

smlink_unlinkinv

Purpose

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.

See Also

`smlink_linkinv`

How To

- "Install and Register SimMechanics Link Software"

Purpose

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.

See Also

`smlink_linksw`

How To

- "Install and Register SimMechanics Link Software"

5

API — Alphabetical List

pmit_add_cadcs

Purpose

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

pmitCadCSH

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

See Also

`PmitCadCSH` | `pmit_create_cadcs` | `PmitError`

Purpose

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

pmitConstrainH

Handle object of `PmitConstrainH` class representing an API CAD model constraint

See Also

`PmitCadModelH` | `PmitConstrainH` | `PmitError`

pmit_add_refincadmodel

Purpose

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

pmitCadModelrefH

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

See Also

`PmitCadModelH` | `PmitCadModelRefH` | `PmitError`

Purpose

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

pmitCadModelrefH

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

See Also

`PmitAssemCompH` | `PmitCadModelRefH` | `PmitError`

PmitAssemCompH

Purpose

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`

Purpose

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

Purpose

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`

Purpose

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

Purpose

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`

Purpose

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

fileName

String specifying STL body geometry file name

See Also

`PmitCadModelH` | `PmitError` | `pmit_set_units`

pmit_cadmodelref_getcadmodel

Purpose

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

Syntax

```
PmitError = pmit_cadmodelref_getcadmodel(PmitCadModelH* pmitCadModelHO
   ut, PmitCadModelRefH cadModelRefH)
```

Description

`PmitError = pmit_cadmodelref_getcadmodel(PmitCadModelH* pmitCadModelHO ut, 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

Output Arguments

pmitCadModelHOut

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

See Also

`PmitCadModelH` | `PmitCadModelRefH` | `PmitError`

Purpose

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

Purpose

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`

Purpose

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

Purpose

Create object of `PmitAssemCompH` class

Syntax

```
PmitError = pmit_create_assemcomp(PmitAssemCompH* const pmitAssemCompH
   Out)
```

Description

`PmitError = pmit_create_assemcomp(PmitAssemCompH* const pmitAssemCompH Out)` 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`

Purpose

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

parentModelH

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

Output Arguments

pmitAssemCompHOut

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

See Also

`PmitAssemCompH` | `PmitCadModelH` | `PmitError`

pmit_create_cad2sm

Purpose

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

createdUsing

String naming the exporter

createdFrom

String naming the source CAD platform or other external application

createdOn

String specifying date that the object was created

createdBy

String specifying name of user creating the object

name

String naming the assembly model

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

Purpose

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

nodeID

String uniquely identifying the coordinate system for associativity purposes

rotation

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

trans

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

Output Arguments

pmitCadCSHOut

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

Definitions

Orthogonal Matrix

A matrix R is orthogonal if it satisfies the matrix multiplication rule $R^T*R = R*R^T = 1$, where 1 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

Purpose

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

mass

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

inertia

Double-type real 6-vector specifying the rotational inertia of the assembly or part. See “Definitions” on page 5-23.

cg

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

volume

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

sarea

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

fileName

String specifying STL body geometry file name

matprops

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

Output Arguments

pmitCadModelHOut

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

Definitions

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

Purpose

Create object of `PmitCadModelRefH` class

Syntax

```
PmitError = pmit_create_cadmodelref(PmitCadModelRefH* const pmitCadMod
   elRefHOut, 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 pmitCadMod elRefHOut, 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

nodeID

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

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.

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” on page 5-25.

trans

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

scale

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

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.

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 5-25.

matprops

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

Output Arguments

pmitCadModelRefHOut

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

Definitions

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`

Purpose

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 5-28.

name

String naming the constraint

type

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

body1Comp

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

body2Comp

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

body1Type

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

body2Type

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

body1Loc

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

body1Axis

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

body2Loc

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

body2Axis

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

Output Arguments

pmitConstrainhOut

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

Definitions

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

Purpose

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`

Purpose

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

Purpose

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`

Purpose

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

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 5-33.

Definitions

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

Purpose

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

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 5-34.

Definitions

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`

Purpose

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

Purpose

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`

Purpose

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

Purpose

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`

Purpose

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

See Also

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

pmit_release_buffer

Purpose

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

See Also

`PmitError` | `pmit_release_object`

Purpose

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

See Also

`PmitError` | `PmitObjectH` | `pmit_release_buffer`

pmit_set_reffixedstatus

Purpose

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

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 5-42.

Definitions

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`

Purpose

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

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 5-43.

Definitions

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

Purpose

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`

angularTol

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

relativeTol

Smallest significant relative numerical difference

See Also

`PmitCad2SMH` | `PmitError` | `PmitLengthUnit` | `pmit_set_units`

Purpose

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

lenUnit

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

See Also

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

PmitVisMatProp

Purpose

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

Purpose

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

filename

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

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"