

Simulink® Requirements™

User's Guide



MATLAB® & SIMULINK®

R2019a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

Simulink® Requirements™ User's Guide

© COPYRIGHT 2017–2019 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 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

September 2017	Online only	New for Version 1.0 (Release 2017b)
March 2018	Online only	Revised for Version 1.1 (Release 2018a)
September 2018	Online only	Revised for Version 1.2 (Release R2018b)
March 2019	Online only	Revised for Version 1.3 (Release R2019a)

1

Requirements Definition

Author Requirements in Simulink	1-2
Author and Edit Requirements Content by Using Microsoft Word	1-4
View or Hide Columns in the Requirements Editor	1-5
Requirement Types	1-6
Import Requirements from Third-Party Tools	1-7
Import Requirements from Microsoft Office Documents	1-7
Import Requirements from ReqIF Files	1-9
Import Modes	1-10
Define Requirements Hierarchy	1-12
Requirement Sets	1-12
Custom Attributes of Requirements Sets	1-12
Create Requirement Set File by Using the Simulink® Requirements™ API	1-14
Update Imported Requirements	1-18
Considerations for Microsoft Word Documents	1-19
Import and Update Requirements from a Microsoft Word Document	1-21
Import Requirements	1-22
Update Requirements	1-22
Export Requirement Sets and Link Sets to Previous Versions of Simulink Requirements	1-24
Export Link Sets	1-24
Export Requirement Sets	1-24

Roundtrip Workflows with ReqIF Files	1-25
Import Requirements from ReqIF Files	1-25
Edit Imported Content	1-26
Export Requirements Content	1-26
Best Practices and Guidelines for ReqIF Roundtrip Workflows	1-28
Managing Requirement Custom IDs	1-28
Guidelines for Updating Referenced Requirements Content .	1-28
Guidelines for Editing Referenced Requirements Content . . .	1-28
Guidelines for Adding Details to Imported Requirements . . .	1-28
Guidelines for Exporting Requirements to ReqIF Files	1-29

Requirements Traceability and Consistency

2

Link Blocks and Requirements	2-2
Work with Simulink Annotations	2-4
Requirement Links	2-6
Link Types	2-6
Review Requirement Links	2-7
Define Custom Requirement and Link Types	2-9
Requirements Consistency Checks	2-11
Check Requirements Consistency in Model Advisor	2-11
Manage Navigation Backlinks in External Requirements Documents	2-16

Requirements-Based Verification

3

Review Requirement Implementation Status Metrics Data . . .	3-2
Review Requirement Verification Status Metrics Data	3-4

Justify Requirements	3-6
-----------------------------------	------------

4 | **Change Tracking and Team-Based Workflows**

Manage Requirements Across a Team	4-2
The Role of Projects in Team-Based Workflows	4-2
Track Changes to Requirements Links	4-4
Enable Change Tracking for Requirements Links	4-4
Resolve Change Issues for Requirement Links	4-6
Add Comments to Links	4-7
Considerations for Using Links Change Tracking	4-7
Compare Requirements Sets	4-8
Compare Two .slreqx Simulink Requirements Sets	4-8
Review Changes in Source-Controlled Files	4-9
Compare Link Sets	4-10
Report Requirements Information	4-11
Report Navigation Links	4-13

5 | **Requirements Management Interface Setup**

Configure RMI for Interaction with Microsoft Office Applications and IBM Rational DOORS Software	5-2
Configure RMI for Microsoft Office Interaction	5-2
Configure RMI for IBM Rational DOORS Interaction	5-2
Configure RMI for IBM Rational DOORS Next Generation Interaction	5-3
Requirements Link Storage	5-5
Save Requirements Links in External Storage	5-5
Load Requirements Links from External Storage	5-6

Move Internally Stored Requirements Links to External Storage	5-7
Move Externally Stored Requirements Links to the Model File	5-7
External Storage	5-8
Guidelines for External Storage of Requirements Links	5-8
Supported Requirements Document Types	5-10
Requirements Settings	5-12
Selection Linking Tab	5-12
Filter Requirements with User Tags	5-13

Microsoft Office Traceability

6

Link to Requirements in Microsoft Word Documents	6-2
Create Bookmarks in a Microsoft Word Requirements Document	6-2
Open the Example Model and Associated Requirements Document	6-4
Create a Link from a Model Object to a Microsoft Word Requirements Document	6-4
Link to Requirements in Microsoft Excel Workbooks	6-8
Navigate from a Model Object to Requirements in a Microsoft Excel Workbook	6-8
Create Requirements Links to the Workbook	6-8
Link Multiple Model Objects to a Microsoft Excel Workbook	6-9
Change Requirements Links	6-10
Navigate to Requirements in Microsoft Office Documents from Simulink	6-12
Enable Linking from Microsoft Office Documents to Simulink Objects	6-12
Insert Navigation Objects in Microsoft Office Requirements Documents	6-13
Customize Microsoft Office Navigation Objects	6-14
Navigate Between Microsoft Word Requirement and Model	6-15

Requirements Traceability with IBM Rational DOORS

7

Configure Requirements Management Interface for IBM Rational DOORS Software	7-2
Before You Begin	7-2
Manually Install Additional Files for DOORS Software	7-2
Diagnose and Fix DXL Errors	7-3
Requirements Traceability with IBM Rational DOORS Next Generation	7-4
Link to Requirements in IBM Rational DOORS Next Generation	7-4
Navigate to Requirements from Simulink	7-5
Work with IBM Rational DOORS Next Generation Projects with Configuration Management Enabled	7-6
Navigate to Requirements in IBM Rational DOORS Databases from Simulink	7-7
Enable Linking from IBM Rational DOORS Databases to Simulink Objects	7-7
Insert Navigation Objects into IBM Rational DOORS Requirements	7-8
Navigate Between IBM Rational DOORS Requirement and Model Object	7-10
Why Add Navigation Objects to IBM Rational DOORS Requirements?	7-11
Customize IBM Rational DOORS Navigation Objects	7-11
Synchronize Simulink Models with IBM Rational DOORS Databases by using Surrogate Modules	7-13
Synchronize a Simulink Model to Create a Surrogate Module	7-13
Create Links Between Surrogate Module and Formal Module in an IBM Rational DOORS Database	7-14
Resynchronize IBM Rational DOORS Surrogate Module to Reflect Model Changes	7-15
Navigate with the Surrogate Module	7-17
Customize IBM Rational DOORS Synchronization	7-19
Synchronization with IBM Rational DOORS Surrogate Modules	7-25
Advantages of Synchronizing Your Model with a Surrogate Module	7-27

Simulink Traceability Between Model Objects

8

Link Model Objects	8-2
Link Objects in the Same Model	8-2
Link Objects in Different Models	8-2
Link Test Cases to Requirements Documents	8-4
Establish Requirements Traceability for Testing	8-4
Link Requirements to Simulink Data Dictionary Entries	8-8
Link Signal Builder Blocks to Requirements and Simulink Model Objects	8-10
Link Signal Builder Blocks to Requirements Documents	8-10
Link Signal Builder Blocks to Model Objects	8-11
Requirements Links for Library Blocks and Reference Blocks	8-14
Introduction to Library Blocks and Reference Blocks	8-14
Library Blocks and Requirements	8-14
Copy Library Blocks with Requirements	8-14
Manage Requirements on Reference Blocks	8-15
Manage Requirements Inside Reference Blocks	8-15
Links from Requirements to Library Blocks	8-18
Navigate to Requirements from Model	8-20
Navigate from Model Object	8-20
Navigate from System Requirements Block	8-20

MATLAB Code Traceability

9

Requirements Traceability for MATLAB Code Lines	9-2
Link Between MATLAB Code Lines and Requirements Information in External Documents	9-2
Enable or Disable Highlighting of Traceability Links for MATLAB Code	9-3
Remove Traceability Links from MATLAB Code Lines	9-3

URL and Custom Traceability

10

Requirement Links and Link Types	10-2
Requirements Traceability Links	10-2
Supported Model Objects for Requirements Linking	10-2
Links and Link Types	10-3
Link Type Properties	10-3
Outgoing Links Editor	10-7
Custom Link Types	10-10
Create a Custom Requirements Link Type	10-10
Implement Custom Link Types	10-17
Why Create a Custom Link Type?	10-18
Custom Link Type Functions	10-18
Custom Link Type Registration	10-19
Custom Link Type Synchronization	10-19

Review and Maintain Requirements Links

11

Highlight Model Objects with Requirements	11-2
Highlight Model Objects with Requirements Using Model Editor	11-2
Highlight Model Objects with Requirements Using Model Explorer	11-3
Navigate to Simulink Objects from External Documents	11-5
Provide Unique Object Identifiers	11-5
Use the rmiobjnavigate Function	11-5
Determine the Navigation Command	11-5
Use the ActiveX Navigation Control	11-6
Typical Code Sequence for Establishing Navigation Controls	11-6

View Requirements Details for a Selected Block	11-8
Requirements Details Workflow	11-8
Requirements Details Limitations	11-8
Generate Code for Models with Requirements Links	11-10
How Requirements Information Is Included in Generated Code	11-11
Create and Customize Requirements Traceability Reports	11-13
Create Requirements Traceability Report for Model	11-13
Customize Requirements Traceability Report for Model ...	11-15
Create Requirements Traceability Report for A Project	11-33
Validate Requirements Links	11-34
Validate Requirements Links in a Model	11-34
Validate Requirements Links in a Requirements Document	11-40
Validation of Requirements Links	11-43
Delete Requirements Links from Simulink Objects	11-46
Delete a Single Link from a Simulink Object	11-46
Delete All Links from a Simulink Object	11-46
Delete All Links from Multiple Simulink Objects	11-46
Document Path Storage	11-48
Relative (Partial) Path Example	11-48
Relative (No) Path Example	11-49
Absolute Path Example	11-49

Test Model Against Requirements and Report Results	13-2
Requirements - Test Traceability Overview	13-2
Display the Requirements and Test Case	13-3
Link Requirements to Tests	13-4
Run the Test	13-5
Report the Results	13-6
Analyze a Model for Standards Compliance and Design Errors	13-8
Standards and Analysis Overview	13-8
Check Model for Style Guideline Violations and Design Errors	13-8
Perform Functional Testing and Analyze Test Coverage . . .	13-11
Incrementally Increase Test Coverage Using Test Case Generation	13-11
Analyze Code and Test Software-in-the-Loop	13-15
Code Analysis and Testing Software-in-the-Loop Overview .	13-15
Analyze Code for Defects, Metrics, and MISRA C:2012	13-15

Requirements Definition

- “Author Requirements in Simulink” on page 1-2
- “Requirement Types” on page 1-6
- “Import Requirements from Third-Party Tools” on page 1-7
- “Define Requirements Hierarchy” on page 1-12
- “Create Requirement Set File by Using the Simulink® Requirements™ API” on page 1-14
- “Update Imported Requirements” on page 1-18
- “Import and Update Requirements from a Microsoft Word Document” on page 1-21
- “Export Requirement Sets and Link Sets to Previous Versions of Simulink Requirements” on page 1-24
- “Roundtrip Workflows with ReqIF Files” on page 1-25
- “Best Practices and Guidelines for ReqIF Roundtrip Workflows” on page 1-28

Author Requirements in Simulink

In this section...

“Author and Edit Requirements Content by Using Microsoft Word” on page 1-4
--

“View or Hide Columns in the Requirements Editor” on page 1-5

In Simulink® Requirements™, you organize your requirements in groups called requirement sets. In each requirement set, you can create additional levels of hierarchy for when you need more requirements to describe the details of a requirement.

In this tutorial, you use the Requirements Editor to create a requirement set to organize related requirements and add requirements to the set.

Suppose that you are writing requirements for a controller model of an automobile cruise control system. You develop these requirements, using your company’s numbering standard for requirements (R1, R2, and so on).

ID and Description	Rationale
R1: The maximum input throttle is 100%	The maximum value of the throttle from the acceleration pedal can be no greater than 100%.
R2: Cruise control has a speed operation range	Cruise control has a minimum and maximum operating speed.
R2.1: The vehicle speed must be at least 40 km/h	The speed of the vehicle must be at least 40 km/h for the cruise control system to engage.
R2.2: The vehicle speed cannot be greater than 100 km/h	The maximum operational speed of the cruise control system for the vehicle is 100 km/h.

Add these requirements to a model called `crs_controller`.

- 1 The model and supporting files are in a project. Open the project. At the MATLAB® command prompt, enter:

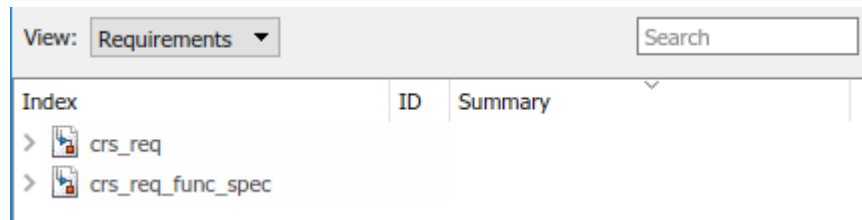
```
slreqCCProjectStart
```



- 2 Open the model. At the command prompt, enter:

```
open_system('models/crs_controller')
```

- 3 Open the Requirements Editor. In the Simulink Editor, select **Analysis > Requirements > Requirements Editor**.

The Requirements Editor opens with the requirements in view in the requirements browser, arranged by requirement set. The `crs_controller` model has two requirements sets that you can see in the browser: `crs_req_func_spec` and `crs_req`.



- 4 Add a requirement set in the requirements browser. From the Requirements Editor toolbar, click **New Requirement Set** .
- 5 You save requirement sets to external files. Save your requirement set to a writable location and name it `cruise_control_reqset.slreqx`.
- 6 When you save requirement sets to external files, you can share requirements with other models.
- 7 Add a requirement to your requirement set. Select the requirement set and click **Add Requirement** .
- 8 In the **Properties** pane, enter details for the requirement. You can copy and paste or drag requirements from another source to the **Properties** pane.
 - **Custom ID:** R1
 - **Summary:** Max input throttle %
 - **Description:** The maximum input throttle is 100%.

If you do not specify a custom ID, the Requirements Editor numbers requirements in order. Custom IDs let you use your company standards for labeling requirements and set the numeric order. (Custom IDs cannot contain a # character.) You can also use an ID to help locate a requirement by searching. Keywords also aid in searching for a requirement.

- 9 Create the requirement R2. Right-click R1 and select **Add Requirement After**. Enter details for the requirement:
 - **Custom ID:** R2

- **Summary:** Cruise control speed operation range
 - **Description:** Cruise control has a minimum and maximum operating speed.
- 10** Create the child requirement of R2. Right-click R2 and select **Add Child Requirement**. Enter details for the requirement:
- **Custom ID:** R2.1
 - **Summary:** Minimum vehicle speed
 - **Description:** The speed of the vehicle must be at least 40 km/h for the cruise control system to engage.

Index	ID	Summary
> crs_req		
> crs_req_func_spec		
▼ cruise_control_reqset*		
1	R1	Max input throttle %
▼ 2	R2	Cruise control speed operation range
2.1	R2.1	Minimum vehicle speed

Add the other child to requirement R2 if you want.

Tip You can rearrange the hierarchy by using the or by dragging requirements.


Author and Edit Requirements Content by Using Microsoft Word

To author and edit the **Description** and **Rationale** fields of your requirements, open Microsoft® Word from within the Requirements Editor or the Requirements Perspective View.

Note This functionality is available only on Microsoft Windows® platforms.

Using Microsoft Word to edit rich text requirements enables you to:

- Spell-check requirements content.
- Resize images.
- Insert and edit equations.
- Insert and edit tables.

To open Microsoft Word, click  in the toolbar of the Edit field of the Description or Rationale. Save the changes to your requirements content within Microsoft Word to see them reflected in Simulink Requirements.

You cannot edit requirements in the built-in editor when you use Microsoft Word to edit requirements content.

View or Hide Columns in the Requirements Editor

You can change the view configuration of the Requirements Editor. Right-click the header row and click **Select Columns**. Add, remove, and reorder columns through the Column Selector dialog box. The view configuration is saved across sessions. You can export view settings to a MAT-File by using the `slreq.exportViewSettings` function and import them by using the `slreq.importViewSettings` function.

You can reset saved view configurations by using the `slreq.resetViewSettings` function.

Requirement Types

When you create or import requirements in Simulink Requirements, you can specify the requirement type by using the **Type** drop-down list in the **Properties** sidebar of the Requirements Editor or the Requirements Perspective View.

Simulink Requirements provides these built-in requirement types:

- **Functional:** Classify requirements that are meant to be implemented or verified in your Model-Based Design workflow. Functional requirements contribute to the Implementation and Verification status metrics of the requirement set that they are in.
- **Container:** Group requirements. Container requirements do not contribute to the Implementation and Verification status metrics of the requirement set that they are in. However, all the Functional requirements under a Container requirement contribute to the status metrics.
- **Informational:** Provide supplemental information. Informational requirements and all requirements under them do not contribute to the Implementation and Verification status metrics of the requirement set that they are in.

You can also define custom requirement types. For more information, see “Define Custom Requirement and Link Types” on page 2-9.

Import Requirements from Third-Party Tools

Using the Requirements Editor in Simulink Requirements, you can work with requirements from third-party applications such as Microsoft Word, Microsoft Excel®, IBM® Rational® DOORS® and other requirements management applications through Requirements Interchange Format (ReqIF) files.

Import requirements from external documents by using the Document Import dialog box. In the Requirements Editor, select **File > Import** to open the Document Import dialog box. If you want to import requirements from a file that is not present on the MATLAB path, you can choose to:

- Store the relative path for the currently running instance of MATLAB,
- Add the parent folder of the requirements document to the MATLAB path, or
- Update the Simulink Requirements path preference to always use the relative path. For more information on requirements document path preference, see “Document Path Storage” on page 11-48.

The Import operation does not import any requirement link information.

Import Requirements from Microsoft Office Documents

Open the Document Import dialog box and select the document type in the **Source document** pane. You can choose to use the currently open document or to select a different document by clicking **Browse**.

Note If you choose to use the currently open document and have more than one document open, the most recently opened document is selected for import.

Import Options for Microsoft Word Documents

You can import requirements in plain and rich text formats from Microsoft Word documents. Use the rich text format to import requirements content such as graphics and tables.

By default, imported requirements content matches the Microsoft Word document outline of section headings. You can also import requirements selectively by using the following qualifiers from the **Requirement Identification** menu:

- Predefined bookmarks in Microsoft Word to identify items and to serve as custom IDs. It is recommended to use bookmarks as requirement Custom IDs as they are persistently stored in the document and cannot be duplicated.
- Regular expression search patterns to identify items by occurrence. See “Regular Expressions” (MATLAB).
- You can choose to ignore outline numbers in the section headers of your Microsoft Word document. If you import requirements as references, it is recommended to ignore outline numbers to prevent issues with the Update process.

Note If you do not have images in your requirements document, consider importing your requirements as plain text to prevent some issues related to font, style, or whitespace differences.

Import Options for Microsoft Excel Spreadsheets

You can import requirements in plain and rich text formats from Microsoft Excel spreadsheets. The plain text format imports only text and associates each column of your spreadsheet to a requirement property. The rich text format imports graphics, layouts, and captures multicell ranges.

Use the qualifiers from the **Requirement Identification** menu to select a subset of your spreadsheet to import requirements from.

- 1 Choose individual rows and columns by mapping columns to requirement attributes. Select **Specify rows and columns** and click **Configure columns**. If there are no predefined headers in your spreadsheet, Simulink Requirements prompts you to specify the row that contains headers for attribute names.
- 2 In the Configure columns dialog box, select the range of rows and columns to import. Select how each column in your spreadsheet can be mapped to Properties and Custom Attributes by choosing an option from that drop-down list. When you map columns to Properties and Custom Attributes, consider:
 - You can select only one column each for the **Custom ID** and **Summary**. If you cannot map one of the columns in the spreadsheet to a column that holds unique requirement Custom IDs, the Import operation automatically generates unique Custom IDs based on the rows in the spreadsheet. These Custom IDs might not be persistent. If you explicitly select a column that does not have unique Custom IDs, you cannot update the requirements document later.

- You can select one or more continuous columns for the **Description** and the **Rationale**. The contents of these columns are concatenated into one field after the import is completed.
- You must select at least one column for the **Summary** or the **Description**.

To omit columns from the import, select the **Ignore** option.


- 3 You can use regular expression search patterns to selectively identify and import items by occurrence. See “Regular Expressions” (MATLAB).

Import Requirements from ReqIF Files

To import requirements from a ReqIF file,

- 1 Open the Document Import dialog box and select ReqIF file (*.reqif or *.reqifz) as the document type in the **Document type** field.
- 2 In the **Document location** field, select the ReqIF file location.
- 3 Simulink Requirements scans the ReqIF file and detects the source tool of the file. You can, however, select a **Source tool** from the drop-down list. Simulink Requirements readily supports ReqIF files created using Polarion, PREEvision, IBM Rational DOORS, or IBM Rational DOORS Next Generation. If you are importing requirements from an external application that is not listed here, select **Generic**.
- 4 Select the location for the destination requirement set. If you are working with requirements that are maintained outside of Simulink Requirements and want to be able to update the imported requirement set with data from updated versions of the ReqIF file, select **Allow updates from external source**. If you plan to migrate your data into Simulink Requirements, do not check this option in order to freely edit the imported requirements. Complete the import process by clicking **Import**.


In ReqIF, a requirement is represented as a **SpecObject**, which has user-defined attributes. Simulink Requirements enables you to map the attributes of a **SpecObject** to either built-in or custom attributes of a requirement and to save this mapping as an XML file for future use. The mapping allows you to customize how requirements data imported from an external requirements management application is displayed in the Requirements Spreadsheet and in the **Properties** pane of the Requirements Editor or Requirements Perspective View.

To modify the attribute mapping after you import, select the top-level Import node of the requirement set (denoted by ) and expand the **Attribute Mapping** pane. You can also load a previously saved attribute mapping by clicking **Load mapping**.

Import Modes




Simulink Requirements provides two import modes for importing requirements content. Before you complete the Import process, you must specify if you want to allow updates to your imported requirements from the external requirements document by selecting or clearing **Allow updates from external source**.

Import Requirements

If you want to permanently migrate your requirements from the external requirements management application, do not allow updates to imported requirements from the external source document. Requirements are then imported as `slreq.Requirement` objects and are represented by  in the Requirements Spreadsheet. Importing requirements as `slreq.Requirement` objects allows you to freely edit, delete, and rearrange requirements.

Import Referenced Requirements

If you choose to allow updates, requirements are imported as referenced requirements (`slreq.Reference` objects) that you can unlock and edit within Simulink Requirements.

Referenced requirements retain some dependencies to the source document and are locked for editing by default. Locked requirements are represented by  in the Requirements Spreadsheet. Edit an individual requirement by navigating to it and clicking **Unlock** in the **Properties** pane. Unlocked requirements are represented by  in the Requirements Spreadsheet. Unlock all referenced requirements by navigating to the top import node (denoted by ) and clicking **Unlock all** in the **Requirement Interchange** pane. You cannot re-lock requirements after you unlock them, except by updating them. You cannot delete or change the hierarchy of referenced requirements from within Simulink Requirements.

If your requirements are imported from an external source, other users are likely to change them in the external source document. To make your referenced requirements reflect the latest version of the requirements as in the external source document, obtain an updated file from the external source. Updating requirements from the external document overwrites all the local changes that you made to imported requirements content.

The Update operation preserves local custom attributes you create within Simulink Requirements. If you have attributes with the same name in the requirement set and in

the external source document, the Update operation overwrites the local values with the attribute values defined in the external source document.

When working with referenced requirements, you can navigate to the requirement in the external source document by clicking **Show in document** in the **Properties** pane. If there is a change in the source document's file name or location, right-click the top node of the requirement set and select **Update source document name or location**.

See Also

slreq.import


More About

- “Update Imported Requirements” on page 1-18
- “Roundtrip Workflows with ReqIF Files” on page 1-25

Define Requirements Hierarchy

Using Simulink Requirements, you can derive lower-level requirements from higher-level requirements to establish and manage parent-child relationships.

The requirement set is the top level of hierarchy for all requirements. All requirements in Simulink Requirements are contained in requirement sets. Every top-level parent requirement in a requirement set is the first-level hierarchy for that set. Referenced requirements (`slreq.Reference` objects) and requirements (`slreq.Requirement` objects) cannot share the same parent requirement.

Within a requirement set, you can change the level of individual requirements by using the  icons in the Requirements Editor or on the **Requirements Browser** toolbar. When you promote or demote a requirement with children, the parent-child hierarchical relationship is preserved. You can also move requirements up and down the same level of hierarchy by right-clicking the requirement and selecting **Move up** or **Move down**.

The Implementation and Verification Status metrics for a requirement set are cumulatively aggregated over all the requirements in the set. Each parent requirement in a requirement set derives its metrics from all its child requirements. For more information on the Implementation and Verification Status metrics, see “Review Requirement Implementation Status Metrics Data” on page 3-2 and “Review Requirement Verification Status Metrics Data” on page 3-4.

Requirement Sets

You can create requirement sets from the Requirements Editor and from the **Requirements Browser**. Requirement set files (`.slreqx`) are not inherently associated with your Simulink models.

Requirement sets have built-in properties such as the Filepath and the Revision number associated with them as metadata. Except for the description, properties of the requirement set are read-only and are updated as you work with the requirement set.

Custom Attributes of Requirements Sets

Define custom attributes for your requirement sets that apply to the requirements they contain. Custom attributes extend the set of properties associated with your

requirements. Define custom attributes for a requirement set from the **Custom Attribute Registries** pane of the Requirements Editor.

To define custom attributes for your requirements sets:

- 1** Open the Requirements Editor. In the Simulink Editor, select **Analysis > Requirements > Requirements Editor**.
- 2** Select the requirements set and click **Add** in the **Custom Attribute Registries** pane.
- 3** The Custom Attribute Registration dialog box opens. Select the type of custom attribute you want to set for your requirements by using the **Type** drop-down list. You can specify custom attributes as text fields, check boxes, and combo boxes and date time entries.

To view the custom attributes for your requirements in the spreadsheet, right-click the requirements set and click **Select Columns**.

When you define a custom attribute as a combobox, the first entry is preset to **Unset** and it cannot be renamed or deleted. Custom attributes that are imported as referenced requirements from an external document become read-only custom attributes after they are imported. The custom attributes of a requirements set are associated with every individual requirement in the set and removing the custom attributes for a requirements set removes it from all the requirements in the set.

Create Requirement Set File by Using the Simulink® Requirements™ API

This example shows how to use the Simulink® Requirements™ API to create a requirement set with a custom hierarchy and custom requirement types. You create a requirement set as an `.slreqx` file. You can distribute the `.slreqx` file across your organization.

You can integrate the requirement set file into a change management system that users can use as a parent document to create their own requirement sets.

Requirement Set Hierarchy

The requirement set that you create in this example contains two top-level parent requirements and parent justifications for implementation and verification. The requirement set follows this hierarchical structure.

Index	ID	Summary
▼ my_New_Req_Set		
▼ 1	R1	System Requirements
▼ 1.1	R1.1	
1.1.1	R1.1.1	
1.1.2	R1.1.2	
▼ 1.1.3	R1.1.3	
1.1.3.1	R1.1.3.1	
1.2	R1.2	
▼ 2	R2	Safety Requirements
2.1	R2.1	
▼ 2.2	R2.2	
2.2.1	R2.2.1	
2.2.2	R2.2.2	
2.2.3	R2.2.3	
▼ 3	#17	Justifications
▼ 3.1	J	Requirement Justifications
3.1.1	J1	Implementation Justifications
3.1.2	J2	Verification Justifications

Create Requirement Set

Navigate to the folder where you want to create the requirement set. Create a requirement set `my_New_Req_Set` with handle `myReqSet` by using the `slreq.new()` function.

```
myReqSet = slreq.new('my_New_Req_Set');
```

Add System Requirements to the Requirement Set

Add a top-level Container requirement for System Requirements to the requirement set

```
myParentReq1 = add(myReqSet, 'Id', 'R1', 'Summary', 'System Requirements', 'Type', 'Co
```

Create child requirements for R1.

```
childReqR11 = add(myParentReq1, 'Id', 'R1.1');  
childReqR12 = add(myParentReq1, 'Id', 'R1.2');
```

Create child requirements for R1.1.

```
childReqR111 = add(childReqR11, 'Id', 'R1.1.1');  
childReqR112 = add(childReqR11, 'Id', 'R1.1.2');  
childReqR113 = add(childReqR11, 'Id', 'R1.1.3');
```

Create a child requirement for R1.1.3.

```
childReqR1131 = add(childReqR113, 'Id', 'R1.1.3.1');
```

Add Safety Requirements to the Requirement Set

Add a top-level Safety requirement to the requirement set. Safety requirements are informational and do not contribute to the Implementation and Verification status summaries. In this example, you define a custom requirement type that extends the Informational requirement type by using the `sl_customization.m` file.

Refresh customizations to add the Safety requirement type to the list of requirement types.

```
sl_refresh_customizations;
```

Create the parent safety requirement.

```
myParentReq2 = add(myReqSet, 'Id', 'R2', 'Summary', 'Safety Requirements', 'Type', 'Sa
```

Create child requirements for R2.

```
childReqR21 = add(myParentReq2, 'Id', 'R2.1');  
childReqR22 = add(myParentReq2, 'Id', 'R2.2');
```

Create child requirements for R2.2.

```
childReqR221 = add(childReqR22, 'Id', 'R2.2.1');  
childReqR222 = add(childReqR22, 'Id', 'R2.2.2');  
childReqR223 = add(childReqR22, 'Id', 'R2.2.3');
```

Add Justifications to the Requirement Set

Create the parent justification.

```
myParentJustification = addJustification(myReqSet, 'Id', 'J', 'Summary', 'Requirement J
```

Add child justifications to the parent justification J to justify requirements for Implementation

```
childJust1 = add(myParentJustification, 'Id', 'J1', 'Summary', 'Implementation Justifi
```

Add child justifications to the parent justification J to justify requirements for Verification

```
childJust2 = add(myParentJustification, 'Id', 'J2', 'Summary', 'Verification Justificat
```

Save the Requirement Set

```
save(myReqSet);
```

Update Imported Requirements

If you checked **Allow updates from external source** during the Import operation, you can update your imported requirements with data from the external requirements source file. When you import requirements as referenced requirements from external requirement documents, they retain a reference to the source document. If you have an updated version of the requirements document, select it during the Update operation. Alternatively, you can update the file name and location of the source requirements document. Right-click the top node of the requirement set and select **Update source document name or location**.

Update your requirements in the requirements set by selecting the top node of the requirements set and clicking **Update** in the **Requirements Interchange** pane. The Update operation:

- Matches previously imported requirements and updates them to reflect all the changes in the new version of the document. This includes overwriting any local changes you may have made to unlocked requirements.
- Generates comments about the differences in the document versions and displays them in the **Comments** pane of the top Import node in the requirement set.
- Updates the **modifiedOn** value for updated requirements and the **updatedOn** value for the top Import node of the requirement set.
- Marks the requirements set as dirty even if there are no changes to requirements data because its **updatedOn** value changes.
- Preserves links to updated requirements.
- Preserves requirement SIDs.
- Preserves comments on requirements.
- Preserves local custom attributes you create within Simulink Requirements. If you have attributes with the same name in the requirement set and in the external source document, the Update operation uses the attribute values defined in the external source document.

If you have change tracking enabled, and there are changes to a requirement with links, the Update operation might trigger change issues that you might have to resolve:

- **Match:** No changes detected between document versions. When you import different versions of the same document, the Update operation might detect only whitespace differences such as carriage returns, linefeeds, and nonbreaking spaces. In this

scenario, the update operation does not update the rich text fields such as the **Description** and the **Rationale**.

- **Insertion:** A new requirement was inserted in the requirement set.
- **Deletion:** A previously imported requirement was deleted from the requirement set.
- **Update:** Built-in or custom attribute values of a previously updated requirement were changed.
- **Move:** A requirement was moved in the requirement hierarchy.
- **Reorder:** A requirement was reordered with respect to its sibling requirements.

Before importing requirements into Simulink Requirements, ensure that your requirements in the requirements document have persistent and unique custom IDs that do not change across document versions. The Update operation otherwise matches unrelated requirements and displays more differences in document versions than actually exist.

Considerations for Microsoft Word Documents

- Use bookmarks for requirement custom IDs in Microsoft Word documents. You can then add content to the document while maintaining requirement references. If you use Microsoft Word section heading texts as requirement custom IDs, changing the document can result in unresolved links after the update operation.
- If you import requirements into a requirement set on one computer and update your requirements on a different computer with a different set of fonts or styles installed, you might notice changes in the requirement descriptions. These changes occur because the font or style is embedded in the HTML descriptions of the requirements.
- Before you execute the update operation, convert documents that you created in an older version of Microsoft Word to the current version. This conversion prevents Microsoft Word from inserting spurious whitespaces in your requirements document.
- In Microsoft Word, resolve issues related to the Trust Center or pending updates if you encounter any errors during the import or update operations. These issues might cause Microsoft Word to block incoming connections from MATLAB .

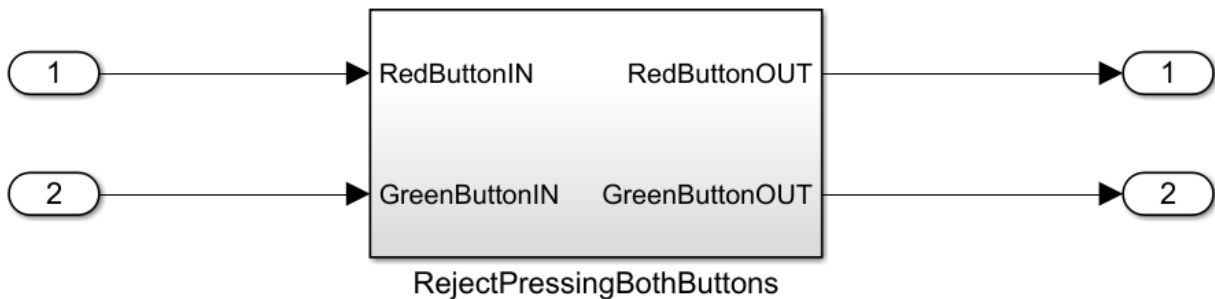
See Also

More About

- “Import Requirements from Third-Party Tools” on page 1-7

Import and Update Requirements from a Microsoft Word Document

This example shows how to import and update requirements from a Microsoft Word requirements document. The model demonstrates a simple two-button switch that passes through outputs when only one switch is pressed at a time.



Simulink® Requirements™ rejectDoublePress.slx
Copyright 2018, The MathWorks, Inc.

Open the model. At the MATLAB command prompt, enter:

```
mdl = 'rejectDoublePress';  
open_system(fullfile(matlabroot, 'examples', 'slrequirements', mdl))
```

The supporting requirements document is located at `matlab/examples/slrequirements`.

This example uses a Microsoft Word document, `Reject_Double_Button_Press_Model_Requirements.docx`. This document contains a set of functional requirements for the `Reject_Double_Button_Press` model. Open the document from `matlab/examples/slrequirements`. The requirements in the document appear in outline format with custom bookmarks for navigation. To get the best results while importing and updating requirements, set up your Microsoft Word documents with document outlines and custom bookmarks.

Save the requirements document to a local folder before importing requirements from it.

Import Requirements

- 1 Open the Requirements Editor. Select **Analysis > Requirements > Requirements Editor**.
- 2 Open the **Document Import dialog box**. In the Requirements Editor, select **File > Import**.
- 3 Select **Microsoft Word document** from the **Document type** drop-down list and select **Reject_Double_Button_Press_Model_Requirements.docx** as the **Document location**.
- 4 Import requirements as plain text, read-only references to a destination requirement set. In the **Requirement Identification** option group, select the options to use bookmarks and to ignore outline numbers. For more information on import options, see “Import Options for Microsoft Word Documents” on page 1-7.

The requirements from the Microsoft Word document are imported into the destination requirements set under a top-level node, **Import1**.

Update Requirements

Requirements that you import as read-only references retain their references to the source requirements document. To change your imported requirements, you must make the changes in the source document first and update your requirements set from within Simulink Requirements. For more information on updating requirements, see “Update Imported Requirements” on page 1-18

- 1 In the document, **Reject_Double_Button_Press_Model_Requirements.docx**, add a new requirement:
 - 2.1.5 The Red and Green Button outputs shall be 0 if no buttons are pressed.
- 2 Create a bookmark called **Red_and_Green_Button_Output_2_1_5** for the new requirement and save the Microsoft Word document.
- 3 In the Requirements Editor, select the top-level node (**Import1**) of the destination requirements set. Update the requirements imported from the document by clicking **Update** in the **Properties** side bar on the right.
- 4 When the update operation is finished, select **Import1** and view the changes to the requirements set in the **Comments** side bar. The Revision number and the **UpdatedOn** values are updated for the requirements set.

See Also

`slreq.ReqSet`

More About

- “Import Requirements from Third-Party Tools” on page 1-7
- “Define Requirements Hierarchy” on page 1-12

Export Requirement Sets and Link Sets to Previous Versions of Simulink Requirements

Export requirement sets and link sets to work with them in previous versions of Simulink Requirements. Starting in R2018b, you can export requirement sets to previous versions of Simulink Requirements (R2017b and beyond).

Export Link Sets

Export link sets to previous versions of Simulink Requirements by exporting the Simulink models that the link sets are associated with to previous versions of Simulink. See “Export a Model to a Previous Simulink Version” (Simulink).

Export Requirement Sets

You can export requirement sets to previous versions from within the Requirements Editor. Navigate to the Requirements View and select the requirement set for export. Before you attempt the Export operation, ensure that the requirement set you want to export is not open in a model. Select **File > Export to Previous**. From the dialog box, select the file name and version you want to export to.

If you export a requirement set with outgoing links to a previous version, Simulink Requirements creates a requirement set and link set files corresponding to that previous version.

Roundtrip Workflows with ReqIF Files


Simulink Requirements supports roundtrip workflows with ReqIF files. ReqIF is an open standard XML format developed for lossless exchange of requirements and their associated metadata between requirements management applications. You can import, edit, and export requirements by using ReqIF files.

Import Requirements from ReqIF Files

To import requirements from a ReqIF file,

- 1 Open the Document Import dialog box and select ReqIF file (*.reqif or *.reqifz) as the document type in the **Document type** field.
- 2 In the **Document location** field, select the ReqIF file location.
- 3 Simulink Requirements scans the ReqIF file and detects the source tool of the file. You can, however, select a **Source tool** from the drop-down list. Simulink Requirements readily supports ReqIF files created using Polarion, PREEvision, IBM Rational DOORS, or IBM Rational DOORS Next Generation. If you are importing requirements from an external application that is not listed here, select **Generic**.
- 4 Select the location for the destination requirement set. If you are working with requirements that are maintained outside of Simulink Requirements and want to be able to update the imported requirement set with data from updated versions of the ReqIF file, select **Allow updates from external source**. If you plan to migrate your data into Simulink Requirements, do not check this option in order to freely edit the imported requirements. Complete the import process by clicking **Import**.

In ReqIF, a requirement is represented as a `SpecObject`, which has user-defined attributes. Simulink Requirements enables you to map the attributes of a `SpecObject` to either built-in or custom attributes of a requirement and to save this mapping as an XML file for future use. The mapping allows you to customize how requirements data imported from an external requirements management application is displayed in the Requirements Spreadsheet and in the **Properties** pane of the Requirements Editor or Requirements Perspective View.

To modify the attribute mapping after you import, select the top-level Import node of the requirement set (denoted by ) and expand the **Attribute Mapping** pane. You can also load a previously saved attribute mapping by clicking **Load mapping**.

Edit Imported Content

Edit imported requirements content by using the editing capabilities of the Requirements Editor. You can unlock and edit a requirement's information such as its **Description** or **Rationale**. You can also define custom attributes on the requirement set and set values for those custom attributes on selected requirements.

Unlock and Edit Imported Requirements

Before you edit an imported requirement, you must unlock it. To unlock all requirements in the requirement set, select the top-level Import node of the requirement set and click **Unlock all** in the **Requirement Interchange** pane. To unlock individual requirements, navigate to the requirement and click **Unlock** in the **Properties** pane.

To add, remove, and edit custom attributes associated with the requirement set, navigate to the top-level node of the requirement set and use the actions available in the **Custom Attribute Registries** pane. Select an individual requirement and unlock it set custom attribute values.

Update Imported Requirements Content

If you selected **Allow updates from external source** during the Import operation, you can update your imported requirement sets with data from the source ReqIF file. Navigate to the top-level Import node of the requirement set and click **Update**. The Update operation overwrites all local modifications such as edits to unlocked requirements with values from the ReqIF source file. The Update operation preserves comments and local attributes.

Export Requirements Content

You can export a requirement set or an individual requirement and its child requirements to ReqIF files from Simulink Requirements. Navigate to the node that you want to export and select **File > Export to ReqIF**.

In the **Export to ReqIF** dialog box, you can select the export mapping and the output ReqIF file name and path. If you are exporting requirement artifacts that you previously imported (roundtrip workflow), it is recommended to use the same import settings for the Export operation.

The Export operation inverts the attribute mapping used by the Import operation. Any local custom attributes that you added to or defined in the Custom Attribute Registry are

also included in the export mapping so that they are visible in external requirements management applications.

See Also

`slreq.import`

More About

- “Import Requirements from Third-Party Tools” on page 1-7
- “Update Imported Requirements” on page 1-18

Best Practices and Guidelines for ReqIF Roundtrip Workflows

Managing Requirement Custom IDs

- When you import requirements as referenced requirements, Simulink Requirements attempts to map a requirement identifier generated by the third-party requirements management application to the Custom ID attribute of the requirement. Verify that the proper attribute mapping between the Custom ID and the requirement identifier is selected.
- Do not modify the requirement custom ID attribute to maintain traceability.

Guidelines for Updating Referenced Requirements Content

- The Update operation overwrites all local modifications such as edits to unlocked referenced requirements with values from the ReqIF source file. Save, check-in, or export your requirement set files before attempting the Update operation.
- The Update operation preserves comments and attributes. Do not delete imported custom attributes as they will be restored when you update the requirement set. For a complete ReqIF roundtrip workflow, include all previously imported attributes.

Guidelines for Editing Referenced Requirements Content

- Rich text attributes like the **Description** and **Rationale** may lose some formatting, particularly tables, during the roundtrip workflow. To preserve formatting, edit these attributes in the same application. Plain text attributes can be edited in multiple applications.
- Rename imported attributes through the Attribute Mapping pane of the Requirements Editor to maintain the connection to the corresponding attribute in the external requirements document during the Export operation.

Guidelines for Adding Details to Imported Requirements

You can add additional details to imported requirements by:

- Adding additional attributes

- Authoring new requirements and linking imported requirements to them

You cannot insert locally authored requirements as children of imported requirements. To associate newly authored requirements with imported requirements, add them to a separate requirement set and link related requirements.

Guidelines for Exporting Requirements to ReqIF Files

- Do not import requirements from multiple ReqIF files into the same requirement set. Each ReqIF file can contain multiple specifications which get imported under separate top Import nodes in the requirement set. Every Import node has a Custom ID that matches the name of the specification.
- Do not import referenced requirements into a requirement set that contains locally authored requirements. For roundtrip workflows, reuse the previous import settings to requirements that were previously imported.
- You cannot update requirements you author within Simulink Requirements if you export them to ReqIF. Import the exported file as referenced requirements into a new requirement set that you can update in the future. Links created to authored requirements will not be preserved when you re-import them. Export and re-import the locally authored requirements before you create links.

See Also

More About

- “Import Requirements from Third-Party Tools” on page 1-7
- “Roundtrip Workflows with ReqIF Files” on page 1-25
- “Update Imported Requirements” on page 1-18

Requirements Traceability and Consistency

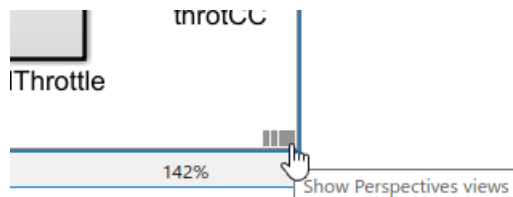
- “Link Blocks and Requirements” on page 2-2
- “Requirement Links” on page 2-6
- “Define Custom Requirement and Link Types” on page 2-9
- “Requirements Consistency Checks” on page 2-11
- “Manage Navigation Backlinks in External Requirements Documents” on page 2-16

Link Blocks and Requirements

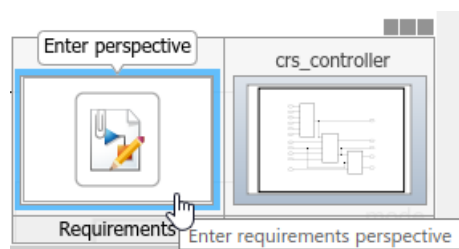
You can create links between your requirements and the Simulink blocks or Stateflow® objects that implement them. By linking between the model elements that implement requirements, you can track whether all your requirements have been implemented. You can also keep requirements and implementation in sync, for example, if a requirement changed or if the implementation causes you to revise your requirements.

In this tutorial, you use the Requirements Perspective View to work with your requirements in the cruise control model. As in the Requirements Editor, you can add requirement sets and requirements in the Requirements Perspective View. In both views, you can link requirements with blocks. In the Requirements Perspective View, visual elements help you to see the links between requirements and blocks. The Requirements Perspective View displays requirement and link information only for the model that is currently open in the Simulink editor.

- 1 In the `crs_controller` model, to preview the perspectives associated with the model, click the perspectives control in the lower-right corner.

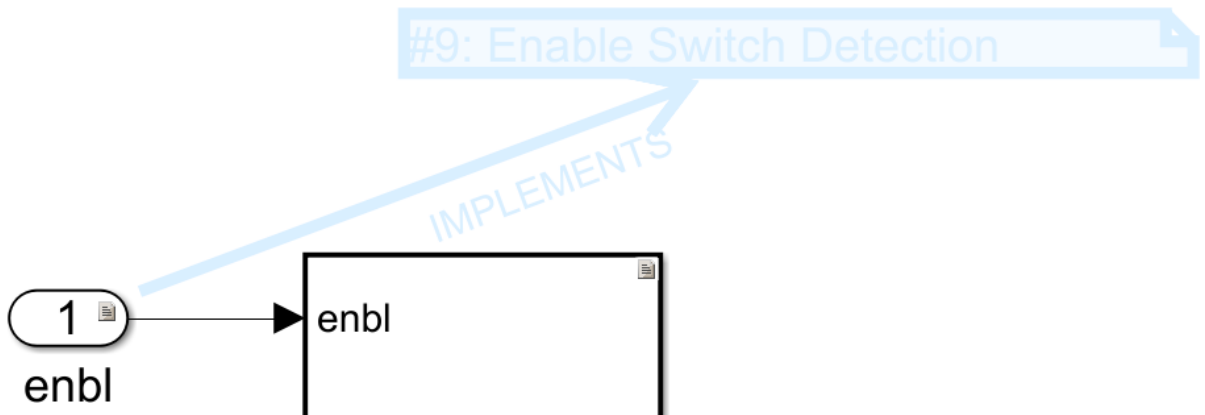


- 2 To open the Requirements Perspective View, click the **Requirements** image.

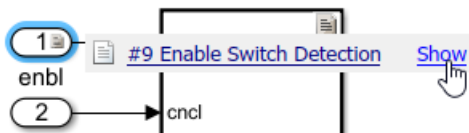


The Requirements Perspective View includes the Requirements Browser, docked at the bottom of the Simulink editor. When you select a requirement, the Property Inspector displays the Simulink Requirements **Properties** pane.

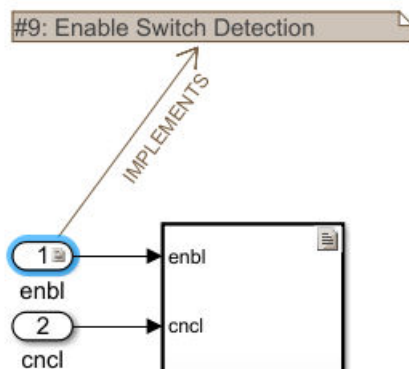
- 3 Link a requirement to the model element that implements it. Search for the requirement that you want to link from. In the Requirements Browser search box, enter `Enable Switch Detection`. As you type, matches are highlighted.
- 4 To create a link, select the matching requirement and drag it onto the `enbl` import block. A markup guide appears on the canvas as a preview of the requirement annotation. Click the markup guide to place the requirement annotation on the canvas. You can also click anywhere else to create the link but not have the annotation display on the canvas.



- 5 A badge appears on the block to show that a link was created. You see badges only in the Requirements Perspective View. To see more information about the requirement, click the badge and select **Show**.

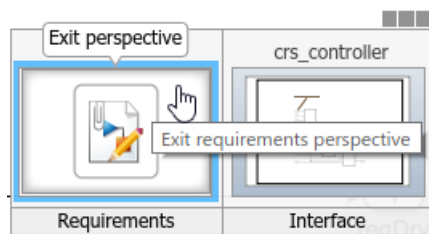


Clicking **Show** displays a requirement annotation on the model canvas.



The annotation displays the ID, the summary, and the relationship between the block and the requirement. In this example, the block implements the requirement. For other possible relationships, see “Requirement Links” on page 2-6.

- To see the requirement description, double-click the annotation.
 - To edit the requirement, right-click the annotation and select **Select in Requirements Browser**. The requirement properties appear in the Property Inspector, where you can edit them.
- 6 Exit the Requirements Perspective View. Click the perspectives control and click the requirements image.



Work with Simulink Annotations

Convert Simulink Annotations to Requirements

You can convert the annotations in your Simulink models to requirements by using the context menu in the Requirements Perspective View and by using the API. See `slreq.convertAnnotation` for more information on converting annotations to requirements by using the API.

To convert annotations to requirements by using the context menu in the Requirements Perspective View:

- 1** Open the Simulink model and enter the Requirements Perspective View.
- 2** Select a requirement set from the Requirements Browser. This is the destination requirement set for the new requirement.
- 3** Right click the annotation you want to convert to a requirement and click **Convert to Requirement**.
- 4** The annotation is converted to a requirement and is linked to the system or subsystem at which the annotation was present.

Link Requirements to Simulink Annotations

Use the Requirements Perspective View to link requirements to text and area annotations on the Simulink Editor. To create a link, select a requirement and drag it onto the annotation. If you link requirements to an area annotation, a badge appears on the annotation to show that the link was created. You see badges only in the Requirements Perspective View. To see more information about the requirement, click the badge and select **Show**.

Requirement Links

Create links between your requirements and various Simulink model elements including blocks, Stateflow objects, Simulink Test™ test objects such as test cases and test suites, Simulink data dictionary entries, MATLAB code, and other requirements by using the linking capabilities available in Simulink Requirements to keep track of how your requirements relate to your model design.

You can create links to blocks and Stateflow objects directly from the Simulink Editor as you work on your model by dragging requirements from the **Requirements Browser** in the Requirements Perspective View. You can create links to Simulink Test test objects from the Simulink Test Test Manager or from the Requirements Editor. For more information on linking Simulink model elements to requirements, see “Link Blocks and Requirements” and “Link to Test Cases from Requirements”.

Requirement links are created to the requirement's SID (Session Independent Identifier) and not to its Custom ID.

Link Types

To keep track of how the various elements in your design are associated with your requirements, you can specify the link type for your requirement links. Link types also specify the nature of requirement-to-requirement links, such as where a requirement is derived from a higher-level requirement.

Simulink Requirements provides these link types.

Link Types in Simulink Requirements

Type	Description
Related to	General relationship between requirement and model element. This link is bidirectional.
Implemented by	The model element implements the requirement and vice versa. These link types contribute to the Implementation Status metric.
Implements	
Verified by	The verification model element or test case verifies that the requirement is satisfied and vice versa. These link types contribute to the Verification Status metric.
Verifies	
Derived from	The destination artifact is derived from the source artifact and vice versa.
Derives	
Refines	The destination artifact adds additional detail for the functionality specified by the source artifact and vice versa.
Refined by	

You can also define custom link types. For more information, see “Define Custom Requirement and Link Types” on page 2-9.

Links between requirements and your Simulink model elements have a source artifact and a destination artifact. Most of the link types are defined relative to the link direction. The Related to link type denotes a general relationship between two entities.

The Implements/Implemented by and Verifies/Verified by link types communicate requirement-model relationships. Specify the source and the destination artifacts correctly for requirements with these link types because the Implementation Status and Verification Status summary metrics are derived from these link types. For more information on the Implementation Status and Verification Status summary metrics, see “Review Requirement Implementation Status Metrics Data” on page 3-2 and “Review Requirement Verification Status Metrics Data” on page 3-4.

Review Requirement Links

Review your requirement links from the Links View. The Links View is available in the Requirements Editor and the **Requirements Browser** in the Requirements Perspective

view. Toggle between the Requirements and Links Views by using the **View** drop-down list in the toolbar. When working in the Simulink Editor, you can review requirement links for individual requirements by using the Property Inspector in the Requirements Perspective view. The Links View of the **Requirements Browser** in the Requirements Perspective view displays only the outgoing links from your source artifacts. Other links associated with your requirements are available in the **Links** pane in the Requirements View. By default, all the outgoing links from a source artifact are stored in a Link Set file (.slmx). See “Requirements Link Storage” on page 5-5 for more information on requirements links storage.

All link information related to a requirement set such as link set files, Simulink models, and test files on the MATLAB or Project path are automatically loaded when you load the requirement set into memory. Link information is not automatically loaded if you save your links with the model as an embedded link set. Load link information programmatically by the using the `slreq.refreshLinkDependencies` command.

Link information is automatically unloaded when you unload the requirement set from memory.

In the Links View, unresolved links are denoted by  .

Define Custom Requirement and Link Types

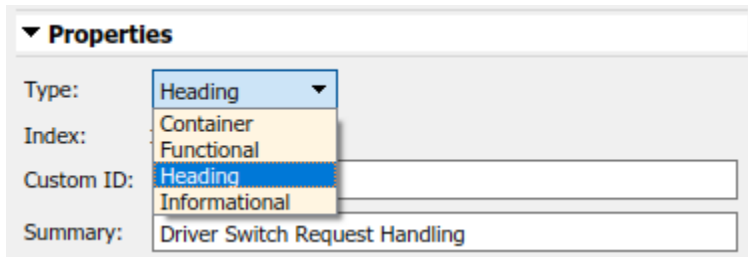
To define custom requirement and link types in addition to the built-in requirement and link types described in “Requirement Types” on page 1-6 and “Link Types” on page 2-6, you customize your Simulinkuser interface by registering a Simulink customization. For more information, “Registering Customizations” (Simulink).

In this example, you define custom requirement and link types by creating an `sl_customization.m` file in the current working folder. The following `sl_customization.m` file creates a custom requirement type called **Heading** and two custom link types called **Satisfy** and **Solve**. You can define custom requirement and link types to exclude requirements from contributing to the Implementation and Verification status metrics as shown in this code example.

```
function sl_customization(cm)
    cObj = cm.SimulinkRequirementsCustomizer;
    cObj.addCustomRequirementType('Heading', slreq.custom.RequirementType.Container, .
    'Headings of functional requirements')
    cObj.addCustomLinkType('Satisfy', slreq.custom.LinkType.Verify, 'Satisfies', ...
    'Satisfied by', 'Links to Verification objects')
    cObj.addCustomLinkType('Solve', slreq.custom.LinkType.Implement, 'Solves', ...
    'Solved by', 'Description')
end
```

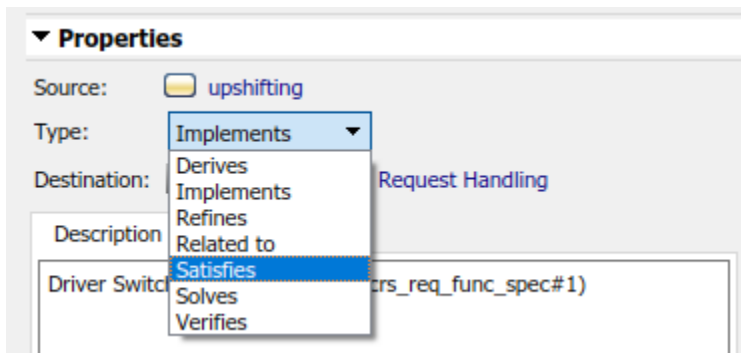
- The **Heading** custom requirement type is defined as a subtype of the built-in **Container** requirement type. **Heading** requirements do not contribute to the Implementation status metric. All **Functional** requirements that are grouped under them do.
- The **Satisfy** custom link type comprises a source and destination artifact: **Satisfies** and **Satisfied by**. It is defined as a subtype of the **Verifies/Verified by** built-in link type. All **Satisfies/Satisfied by** requirement links contribute to the Verification status metric.
- The **Solve** custom link type comprises a source and destination artifact: **Solves** and **Solved by**. It is defined as a subtype of the **Implements/Implemented by** built-in link type. All **Solves/Solved by** requirement links contribute to the Implementation status metric.

You can select the custom requirement or link type from the Requirements Editor or the Requirements Perspective view. To select the custom requirement type, navigate to the **Requirements** view and select a requirement. Select the custom requirement type from the **Type** drop-down list in the **Properties** pane.



The screenshot shows a 'Properties' pane with a dropdown menu open for the 'Type' field. The menu options are: Heading (selected), Container, Functional, Heading, and Informational. Other fields include 'Index', 'Custom ID', and 'Summary' with the value 'Driver Switch Request Handling'.

To select the custom link type, navigate to the **Links** view and select a link. Select the custom link type from the **Type** drop-down list in the **Properties** pane.



The screenshot shows a 'Properties' pane with a dropdown menu open for the 'Type' field. The menu options are: Implements (selected), Derives, Implements, Refines, Related to, Satisfies, Solves, and Verifies. Other fields include 'Source' with the value 'upshifting', 'Destination' with the value 'Request Handling', and 'Description' with the value 'Driver Switch'.

Requirements Consistency Checks

Check Requirements Consistency in Model Advisor

- “Identify requirement links with missing documents” on page 2-11
- “Identify requirement links that specify invalid locations within documents” on page 2-12
- “Identify selection-based links having description fields that do not match their requirements document text” on page 2-13
- “Identify requirement links with path type inconsistent with preferences” on page 2-14
- “Identify IBM Rational DOORS objects linked from Simulink that do not link to Simulink” on page 2-15

You can check requirements consistency using the Model Advisor.

Identify requirement links with missing documents

Check ID: `mathworks.req.Documents`

Verify that requirements link to existing documents.

Description

You used the Requirements Management Interface (RMI) to associate a design requirements document with a part of your model design and the interface cannot find the specified document.

Available with Simulink Requirements.

Results and Recommended Actions

Condition	Recommended Action
The requirements document associated with a part of your model design is not accessible at the specified location.	Open the Requirements dialog box and fix the path name of the requirements document or move the document to the specified location.

Capabilities and Limitations

You can exclude blocks and charts from this check.

Tips

If your model has links to a DOORS requirements document, to run this check, the DOORS software must be open and you must be logged in.

Identify requirement links that specify invalid locations within documents

Check ID: `mathworks.req.Identifiers`

Verify that requirements link to valid locations (e.g., bookmarks, line numbers, anchors) within documents.

Description

You used the Requirements Management Interface (RMI) to associate a location in a design requirements document (a bookmark, line number, or anchor) with a part of your model design and the interface cannot find the specified location in the specified document.

Available with Simulink Requirements.

Results and Recommended Actions

Condition	Recommended Action
The location in the requirements document associated with a part of your model design is not accessible.	Open the Requirements dialog box and fix the location reference within the requirements document.

Capabilities and Limitations

You can exclude blocks and charts from this check.

Tips

If your model has links to a DOORS requirements document, to run this check, the DOORS software must be open and you must be logged in.

If your model has links to a Microsoft Word or Microsoft Excel document, to run this check, those applications must be closed on your computer.

Identify selection-based links having description fields that do not match their requirements document text

Check ID: mathworks.req.Labels

Verify that descriptions of selection-based links use the same text found in their requirements documents.

Description

You used selection-based linking of the Requirements Management Interface (RMI) to label requirements in the model's **Requirements** menu with text that appears in the corresponding requirements document. This check helps you manage traceability by identifying requirement descriptions in the menu that are not synchronized with text in the documents.

Available with Simulink Requirements.

Results and Recommended Actions

Condition	Recommended Action
Selection-based links have descriptions that differ from their corresponding selections in the requirements documents.	If the difference reflects a change in the requirements document, click Update in the Model Advisor results to replace the current description in the selection-based link with the text from the requirements document (the external description). Alternatively, you can right-click the object in the model window, select Edit/Add Links from the Requirements menu, and use the Requirements dialog box that appears to synchronize the text.

Capabilities and Limitations

You can exclude blocks and charts from this check.

Tips

If your model has links to a DOORS requirements document, to run this check, the DOORS software must be open and you must be logged in.

If your model has links to a Microsoft Word or Microsoft Excel document, to run this check, those applications must be closed on your computer.

Identify requirement links with path type inconsistent with preferences

Check ID: mathworks.req.Paths

Check that requirement paths are of the type selected in the preferences.

Description

You are using the Requirements Management Interface (RMI) and the paths specifying the location of your requirements documents differ from the file reference type set as your preference.

Available with Simulink Requirements.

Results and Recommended Actions

Condition	Recommended Action
The paths indicating the location of requirements documents use a file reference type that differs from the preference specified in the Requirements Settings dialog box, on the Selection Linking tab.	Change the preferred document file reference type or the specified paths by doing one of the following: <ul style="list-style-type: none">• Click Fix to change the current path to the valid path.• In the model window, select Analysis > Requirements > Settings, select the Selection Linking tab, and change the value for the Document file reference option.

Linux Check for Absolute Paths

On Linux® systems, this check is named **Identify requirement links with absolute path type**. The check reports warnings for requirements links that use an absolute path.

The recommended action is:

- 1 Right-click the model object and select **Requirements > Edit/Add Links**.
- 2 Modify the path in the Document field to use a path relative to the current working folder or the model location.

Capabilities and Limitations

You can exclude blocks and charts from this check.

Identify IBM Rational DOORS objects linked from Simulink that do not link to Simulink

Identify IBM Rational DOORS objects that are targets of Simulink-to-DOORS requirements traceability links, but that have no corresponding DOORS-to-Simulink requirements traceability links.

Description

You have Simulink-to-DOORS links that do not have a corresponding link from DOORS to Simulink. You must be logged in to the IBM Rational DOORS Client to run this check.

Available with Simulink Requirements.

Results and Recommended Actions

The Requirements Management Interface (RMI) examines Simulink-to-DOORS links to determine the presence of a corresponding return link. The RMI lists DOORS objects that do not have a return link to a Simulink object. For such objects, create corresponding DOORS-to-Simulink links:

- 1** Click the **Fix All** hyperlink in the RMI report to insert required links into the DOORS client for the list of missing requirements links. You can also create individual links by navigating to each DOORS item and creating a link to the Simulink object.
- 2** Re-run the link check.

Manage Navigation Backlinks in External Requirements Documents

Simulink Requirements enables you to insert navigation backlinks in external requirements documents to match requirement links in Simulink. The software also enables you to remove unmatched backlinks from external requirements documents when there is no matching link from Simulink to a requirement in the document.

You can insert navigation backlinks in:

- Microsoft Word documents,
- Microsoft Excel spreadsheets, and
- IBM Rational DOORS modules.

When you insert a navigation backlink in an external requirements document, Simulink Requirements creates two link artifacts - a link (`slreq.Link` object) in Simulink Requirements and a navigation hyperlink (backlink) icon in the external requirements document. Backlink management in Simulink Requirements helps you synchronize both these link artifacts.

To insert or remove backlinks for your requirements, navigate to the **Links** view in the Requirements Editor or Requirements Perspective View. Right-click the link set file corresponding to the currently open model and click **Update Backlinks**. The **Backlinks checked** dialog box displays the total count of added and removed backlinks.

See Also

More About

- “Navigate to Requirements in Microsoft Office Documents from Simulink” on page 6-12
- “Link to Requirements in Microsoft Word Documents” on page 6-2
- “Link to Requirements in Microsoft Excel Workbooks” on page 6-8

Requirements-Based Verification

- “Review Requirement Implementation Status Metrics Data” on page 3-2
- “Review Requirement Verification Status Metrics Data” on page 3-4
- “Justify Requirements” on page 3-6



















































Review Requirement Implementation Status Metrics Data

Simulink Requirements provides you with Implementation Status summaries for your requirement sets. You can use these status summaries to identify gaps in requirement implementation in your design. Requirements that have the link type set to **Implemented** by contribute to the Implementation Status summary metric.

The Implementation Status metrics for a requirement set are cumulatively aggregated over all the requirements in the set. Each child requirement belonging to a parent requirement must be implemented for the parent requirement to be considered as implemented.

You can view the Implementation Status metric for your requirement sets from both the Requirements Editor and the Requirements Browser in the Requirements Perspective View. To toggle the metric display, select **Display > Implementation Status** from the Requirements Editor menu. Hover your mouse over the Implemented column in the Requirements Editor or **Requirements Browser** for each requirement or requirement set to view the Implementation Status metric associated with it.

View: Requirements ▾

Index	ID	Summary	Implemented	Verified
▼  crs_req_func_spec				
▼  1	#1	Driver Switch Request Handling		
 1.1	#2	Switch precedence		
 1.2	#3	Avoid repeating commands		
>  1.3	#4	Long Switch recognition		
 1.4	#7	Cancel Switch Detection		
 1.5	#8	Set Switch Detection		
 1.6	#9	Enable Switch Detection		
 1.7	#10	Resume Switch Detection		
>  1.8	#11	Increment Switch Detection		
>  1.9	#15	Decrement Switch Detection		
▼  2	#19	Cruise Control Mode		
>  2.1	#20	Disable Cruise Control system		
>  2.2	#24	Operation mode determination		
▼  3	#37	Calculate Target Speed and Throttle ...		
 3.1	#38	Disabled case		
 3.2	#39	Enabled case		

Review Requirement Verification Status Metrics Data

Simulink Requirements maintains Verification Status summaries for your requirement sets. You can use these status summaries to identify missing or incomplete requirement verification. Verification status metrics are available if you have Simulink Design Verifier™ and use Proof Objective blocks as requirement link sources, or Simulink Test and use test cases in the Test Manager as requirement link sources.

A requirement must satisfy the following conditions to achieve full Verification status:

- The requirement has one or more links where the link type is set to **Verified by**.
- The link sources for the **Verified by** links must be either:
 - Simulink Test test cases and test suites
 - Simulink Design Verifier Proof Objective blocks
 - Blocks from the Simulink Model Verification Library
- The verification associated with the link source must pass. Every child requirement belonging to a parent requirement must be verified for the parent requirement to be considered verified.

A link to a verification item can have five possible Verification Statuses:

- Passed: The test linked with the requirement has passed.
- Failed: The test linked with the requirement has failed.
- Justified: The requirement has been justified for verification.
- Unexecuted: The test linked with the requirement has not been executed.
- None: The test linked with the requirement has no result.

You can generate the Verification Status metric for your requirements linked to Simulink Test test cases by running the tests associated with them from the Test Manager as described in “Link to Test Cases from Requirements”.

If you have links from your requirements to Simulink Design Verifier Proof Objective blocks, you can generate the Verification Status metric by simulating the model.

If you have the Verification Status metric displayed, you can also run the tests associated with a requirement and its child requirements from the Requirements Editor by right-clicking the requirement node and selecting **Run Tests**. In the **Run Tests** dialog box, select the tests you want to run and click **Run Tests**.

Note If you have linked requirements to Simulink Design Verifier Proof Objective blocks in multiple models, the **Run Tests** dialog box runs a Simulink Design Verifier analysis when the corresponding models are open.

You can view the Verification Status metric for your requirements sets from both the Requirements Editor and the Requirements Browser in the Requirements Perspective View. To toggle the metric display, select **Display > Verification Status** from the Requirements Editor menu.

Hover your mouse over the Verified column in the Requirements Editor or **Requirements Browser** for each requirement or requirements set to view the Verification Status metric associated with it.

View: Requirements ▾

Index	ID	Summary	Implemented	Verified
▼ crs_req_func_spec				
▼ 1	#1	Driver Switch Request Handling		
1.1	#2	Switch precedence		
1.2	#3	Avoid repeating commands		
> 1.3	#4	Long Switch recognition		
1.4	#7	Cancel Switch Detection		
1.5	#8	Set Switch Detection		
1.6	#9	Enable Switch Detection		
1.7	#10	Resume Switch Detection		
> 1.8	#11	Increment Switch Detection		
> 1.9	#15	Decrement Switch Detection		
▼ 2	#19	Cruise Control Mode		
> 2.1	#20	Disable Cruise Control system		
> 2.2	#24	Operation mode determination		
▼ 3	#37	Calculate Target Speed and Throttle ...		
3.1	#38	Disabled case		
3.2	#39	Enabled case		

Justify Requirements


Use requirement justification to exclude requirements from the Implementation and Verification Status metrics calculation for your requirement sets. You may have non-functional requirements in your model design specification that cannot be implemented in your design. You may also have requirements that require manual testing, instead of linking to test cases or verification subsystems. You can justify these requirements to override their Implementation and Verification statuses and iterate more effectively on your model design.

A justification is an object associated with a requirement. All justification objects in a requirement set are grouped under a single top-level justification object as its children. Any requirement can be justified for implementation, verification, or both. Justified requirements do not contribute to the overall aggregation of Implementation and Verification Status metrics and appear light blue in the Implemented and Verified columns of the Requirements Editor.

View: Requirements

Index	ID	Summary	Implemented	Verified
crs_req_func_spec				
1	#1	Driver Switch Request Handling		
1.1	#2	Switch precedence		
1.2	#3	Avoid repeating commands		
1.3	#4	Long Switch recognition		
1.4	#7	Cancel Switch Detection		
1.5	#8	Set Switch Detection		
1.6	#9	Enable Switch Detection		
1.7	#10	Resume Switch Detection		
1.8	#11	Increment Switch Detection		
1.9	#15	Decrement Switch Detection		
2	#19	Cruise Control Mode		
2.1	#20	Disable Cruise Control system		
2.2	#24	Operation mode determination		
3	#37	Calculate Target Speed and Throttle ...		
3.1	#38	Disabled case		
3.2	#39	Enabled case		

There are two workflows for justifying requirements in Simulink Requirements. You can either create a justification object and link requirements to it or use an existing justification object and link requirements to it.

- 1** Create a justification object by clicking the  icon in the Requirements Editor or **Requirements Browser** toolbar.
- 2** Right-click the requirement you want to link with the justification object and select **Justification > Link with new Justification for implementation** or **Link with new Justification for verification**.

To justify a parent requirement and all its child requirements, select the Hierarchical Justification option in the **Property Inspector**.

Note You cannot link justification objects to objects that are not requirements.

Change Tracking and Team-Based Workflows

- “Manage Requirements Across a Team” on page 4-2
- “Track Changes to Requirements Links” on page 4-4
- “Compare Requirements Sets” on page 4-8
- “Compare Link Sets” on page 4-10
- “Report Requirements Information” on page 4-11

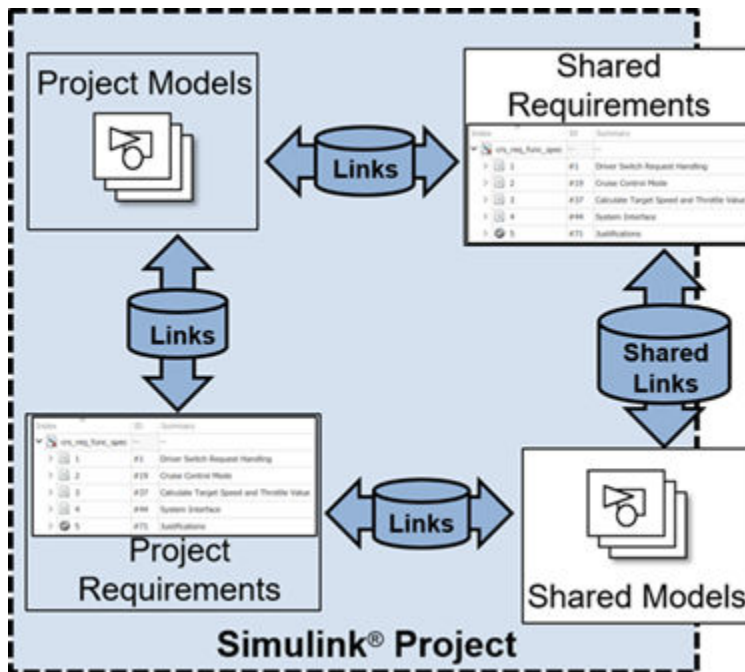
Manage Requirements Across a Team

Use Simulink Requirements with projects for team-based collaboration. You can:

- Create a repository of models, supporting files, documents, requirements, and links by using projects.
- Reuse requirements and links sets for multiple projects.
- Integrate shared library models and requirements sets with local models and requirements sets.

The Role of Projects in Team-Based Workflows

After you “Author Requirements in Simulink” on page 1-2 and create links between design elements and your requirements, you can use projects to share requirements sets, requirements links, and Simulink models and supporting files across your team.



Your project can contain:

- Local, modifiable versions of models, requirements sets, requirements links, and supporting files
- Shared library versions of models, requirements sets, requirements links, and supporting files

If you incorporate the shared library models, requirements sets, requirements links, and supporting files in your requirements analysis, you can create requirements links between your local models and shared requirements. You can also create requirements between your local requirements and shared models and supporting files.

When you include local and shared library versions of models, requirements sets, requirements links, and supporting files in your project, Simulink Requirements loads into the Requirements Editor all the requirement sets and link sets automatically when you open the project. Your shared library requirements data is integrated into your local requirements data. When a link set is no longer used by any requirement sets or other artifacts, they are unloaded automatically.

Generate requirement link information for currently open projects created with older versions of Simulink programmatically by using the `slreq.refreshLinkDependencies` command.

Track Changes to Requirements Links

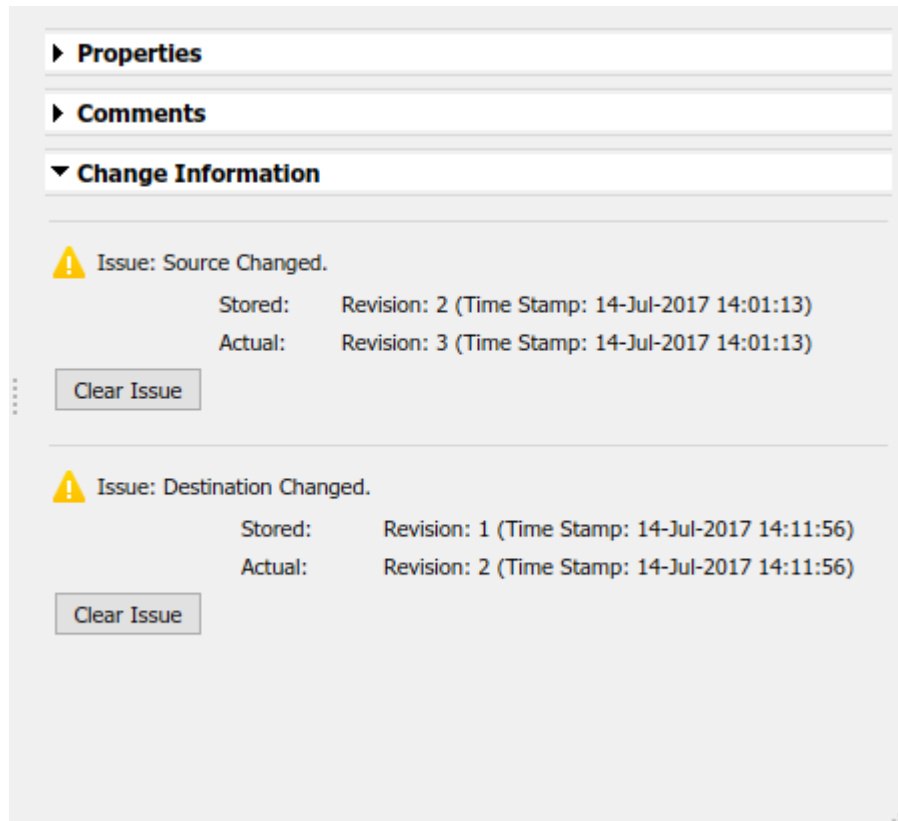
After you “Author Requirements in Simulink” on page 1-2 and create links between design elements and your requirements, Simulink Requirements tracks the requirements links and detects when link artifacts (source or destination) or requirements change. In the Requirements Editor, changes to requirements links are highlighted in red. You can then resolve link issues or clear links changes that have no impact on the requirement status. Track change information from the **Links** view of the Requirements Editor.

Enable Change Tracking for Requirements Links

To view the change information for your requirements links:

- 1 From the Simulink menu, select **Analysis > Requirements > Requirements Editor**.
- 2 Open a requirements set.
- 3 From the **View** drop-down list, select **Links**.
- 4 To enable change tracking from the Requirements Editor, select **Display > Change Information**.

The **Links** view of the Requirements Editor highlights any links with revision mismatch issues in red.

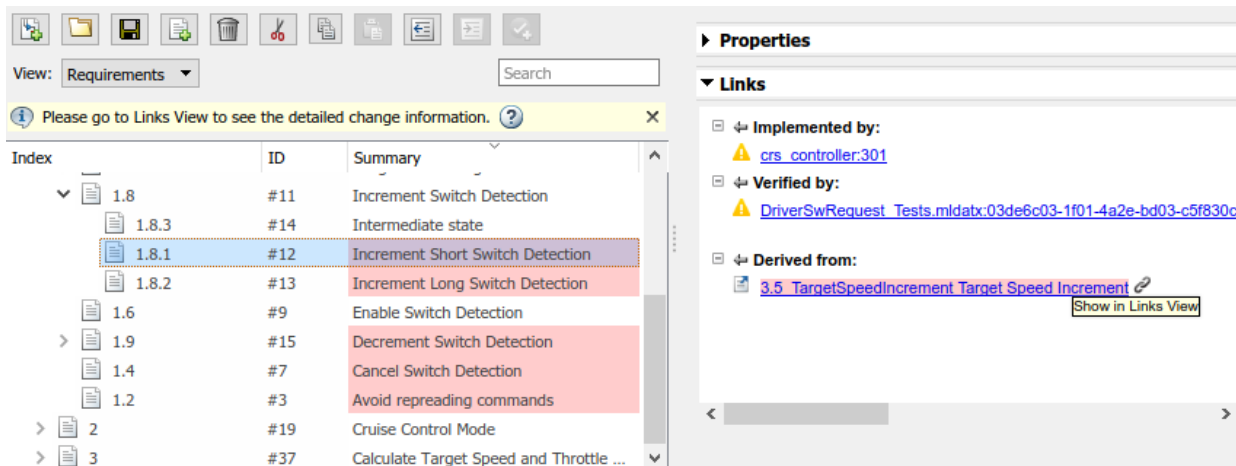


Alternatively, you can enable change tracking for requirement links from the Requirements Perspective. Right-click an item in the Requirements Perspective and select **Change Information**.

The **Change Information** section for each link details any requirement link change issues. Simulink Requirements reports any source or destination requirements that have more recent changes than when you established the requirement link. Simulink Requirements considers the revision information and timestamp for a requirement item when the link was created and the current revision information and timestamp for the requirement item.

Review Requirements Change Issues from the Requirements View

You can view change issues associated with a particular requirement item from the Requirements view. After you “Enable Change Tracking for Requirements Links” on page 4-4, select a requirement item from the **Requirements** view. Any requirements items with requirement link change issues are highlighted in red. The relevant requirement link with a change issue is also highlighted in red in the Links section of the Requirements Editor.



To view the link change issue, and then “Resolve Change Issues for Requirement Links” on page 4-6, click the link icon to the right of requirement link.

Resolve Change Issues for Requirement Links

If you enable change tracking for requirements links, changes to your requirements that cause link issues are highlighted in red in the **Links** view of the Requirements Editor. If a change has no impact, you can clear the issue to update to the latest revision number for the requirement. In the Links view of the Requirements Editor, select the link with a link change issue. In the **Change Information** section, click **Clear Issue**. To clear all change issues for an entire link set, right-click the link set and select **Clear All Change Issues**.

If the link issue affects the status of your requirements, you can change the model, the requirements, the test cases, or the links themselves to resolve the discrepancy between the requirements and your model and testing.


Add Comments to Links

Whenever you resolve link issues, it is good practice to add a comment to the link describing the action that you took. Each link has a **Comments** section. To add a comment:

- 1 In the **Links** view of the Requirements Editor, select the link.
- 2 In the Comments section, select **Add Comment**.

Considerations for Using Links Change Tracking

Change tracking information is updated only when you enable change tracking or when you manually refresh the change tracking information. To refresh the change tracking information manually:

- From the Requirements Editor, select **Analysis > Refresh Change Information**.
- From the Requirements Perspective, or from the Requirements Editor, click the **Refresh** button .

Simulink Requirements provides change tracking information only for link sources that are resolved when you view the change information.

You can resolve some links by using the Requirements Editor.

- 1 In the **Links** view of the Requirements Editor, select the link with an unresolved link item. The Change Information section shows **Unresolved Link Item(s)**.
- 2 To open the link source, click the **Source** link in the **Properties** section.

When the link model opens, the link source is loaded and the Change Information section displays a resolved link. For more information, see “Review Requirement Links” on page 2-7.

See Also

More About

- “Create a Project from a Model” (Simulink)

Compare Requirements Sets

To compare differences between two requirements sets, use the “Compare Revisions” (Simulink) tool.

Compare Two .slreqx Simulink Requirements Sets

If you have two versions of a .slreqx Simulink requirements set file, use the Simulink “Compare Revisions” (Simulink) tool to find any differences between the two files.

Select Two Requirements Set Files to Compare

- 1 In the **Current Folder** pane of MATLAB, or in the **Project Files View** of your project, select the first file for comparison.
- 2 In the **Current Folder** pane of MATLAB, or in the **Project Files View** of your project, press **Ctrl**, and then click the second file for comparison.
- 3 Right-click either file and select **Compare Selected Files/Folders**.

Select One File to Compare

- 1 In the **Current Folder** pane of MATLAB, right-click first file and select **Compare Against > Choose**.
- 2 Select the second file for comparison and select Simulink Requirements Comparison as the **Comparison type**.

The Simulink comparison tool shows the differences between the two .slreqx requirements sets. The comparison shows which specific requirements in a requirement set changed and which fields of each requirement changed.

Note The comparison tool shows only changes in saved .slreqx requirements sets. Changes that have occurred in memory but are not yet saved to file are not shown.

To view a requirements item in the Requirements Editor, highlight the requirements item and click **Highlight Now**. The requirements item from the right comparison pane opens in the Requirements Editor. If you select **Always Highlight**, the Requirements Editor opens to the selected requirements item whenever you click one.

Review Changes in Source-Controlled Files

If you use a separate change management tool to manage changes to your projects, you can use the Simulink comparison tool with your source-controlled Simulink Requirements files. For more information, see “Compare Revisions” (Simulink).

Compare Link Sets

If you have two versions of a .slmx Simulink link set file, use the Simulink “Compare Revisions” (Simulink) tool to find any differences between the two files.

Select Two Link Set Files to Compare

- 1 In the **Current Folder** pane of MATLAB, or in the **Project Files View** of your project, select the first file for comparison.
- 2 In the **Current Folder** pane of MATLAB, or in the **Project Files View** of your project, press **Ctrl**, and then click the second file for comparison.
- 3 Right-click either file and select **Compare Selected Files/Folders**.

Select One File to Compare

- 1 In the **Current Folder** pane of MATLAB, right-click the first file and select **Compare Against > Choose**.
- 2 Select the second file for comparison and select Simulink Requirements Comparison as the **Comparison type**.

The Simulink comparison tool shows the differences between the two .slmx link set files. The comparison shows which specific links in a link set changed and which fields of each link changed.

Note The comparison tool shows only changes in saved .slmx link sets. Changes that have occurred in memory but are not yet saved to file are not shown.

To view a link in the Links View of the Requirements Editor, highlight the link and click **Highlight Now**. The link from the right comparison pane opens in the Links View of the Requirements Editor. If you select **Always Highlight**, the Requirements Editor opens to the selected link item whenever you click one.

Report Requirements Information

To document your requirements for review, you can create a report for one or more requirement sets. You can select the requirements information to contain in the report, including:

- Navigable links to model entities and other requirements
- Requirements change and revision information
- Implementation and Verification status summaries

. You can create reports in .docx (Microsoft Word), PDF and HTML formats. If you select multiple requirement sets for reporting, the information is contained in a single report.

You can create reports using the **Report Generation Options** dialog box or programmatically by using the `slreq.generateReport` function.

Report Generation Options [X]

Title Page Options

Report title:

Report authors:

File

Folder: D:\

File Name: myReport.docx

Included Requirement Sets

	Name	Path
<input checked="" type="checkbox"/>	crs_req	D:\SLRequirementsCruiseControlExample-master\S...
<input checked="" type="checkbox"/>	crs_req_func_spec	D:\SLRequirementsCruiseControlExample-master\S...

Report content

Table of Contents
 Implementation Status

Rationale
 Verification Status

Keywords
 Links

Custom Attributes

Revision information
 Artifact Link Type

Comments
 Change Information

Empty Sections

To create a report by using the Report Generation Options dialog box:

- 1 Right-click a requirement set in the Requirements Editor or Requirements Browser, and select **Generate Report**.

To create a report with multiple requirements sets, select **Report > Generate Report** in the Requirements Editor menu.

The Report Generation Options dialog box opens.

- 2 Set the report file name and location by clicking the **Select** button next to the file name.
- 3 Select report content options.
- 4 Select requirement sets to include in the report. The dialog box displays requirement sets that are loaded in memory. To include a requirement set that does not appear in the list, first open the requirement set using the Requirements Editor.
- 5 Click **Generate Report**.

The Report Appendix provides summaries of all the change issues and requirement set artifacts that you create the report for.

Report Navigation Links

The requirements report contains links you can use to navigate to model items and other requirements. For example, this requirement is implemented by two model entities, and is derived from two requirements. **Ctrl**+click a link to open the linked item.

↔Implemented by

- ☐ Switch2
- ☐ Enumerated Constant2

↔Derived from

- 📄 3.3_Activatingcruisecon Activating cruise control
- 📄 3.4_Deactivatingcruiseec Deactivating cruise control

If you use `slreq.generateReport` to generate a report as a Microsoft Word document, you will need to manually update the Table of Contents. Open the report, select all the contents, and press **F9**.

See Also

`slreq.generateReport` | `slreq.getReportOptions`

Requirements Management Interface Setup

- “Configure RMI for Interaction with Microsoft Office Applications and IBM Rational DOORS Software” on page 5-2
- “Requirements Link Storage” on page 5-5
- “Supported Requirements Document Types” on page 5-10
- “Requirements Settings” on page 5-12

Configure RMI for Interaction with Microsoft Office Applications and IBM Rational DOORS Software

The Requirements Management Interface (RMI) must communicate with external software products such as Microsoft Office and IBM Rational DOORS software to establish links between requirements and Simulink model elements.

Perform initial configuration steps to set up the RMI if you want to:

- Use ActiveX® controls for navigation from Microsoft Office documents to Simulink models (PC only).
- Use the RMI with IBM Rational DOORS software (PC only).
- Use the RMI with IBM Rational DOORS Next Generation web server.

Configure RMI for Microsoft Office Interaction

When you work with older requirements documents that include ActiveX controls inserted by previous versions of Simulink, registerActiveX controls. More recent versions of Simulink use HTTP hyperlinks to navigate from Microsoft Office to Simulink.

- 1 Run MATLAB as an administrator.
- 2 At the command prompt, enter:

```
rmi setup
```
- 3 To register the current MATLAB installation as an ActiveX Automation Server, press Y.

Configure RMI for IBM Rational DOORS Interaction

If you have IBM Rational DOORS installed, you must configure it to communicate with MATLAB.

- 1 Run MATLAB as an administrator.
- 2 At the command prompt, enter:

```
rmi setup
```
- 3 Skip the ActiveX Automation Server setup by pressing N.

The IBM Rational DOORS-MATLAB Interface setup utility opens.

- 4 If the detected path to your IBM Rational DOORS installation is correct, configure the software to communicate with MATLAB by pressing 1.
- 5 If MATLAB did not detect your IBM Rational DOORS installation in the previous step, enter the installation folder by pressing 2.

Note You can directly access the IBM Rational DOORS-MATLAB Interface setup utility. At the command prompt, enter:

```
rmi setup doors
```

Configure RMI for IBM Rational DOORS Next Generation Interaction

- 1 At the command prompt, enter:

```
rmi setup
```
- 2 You do not need to set up the ActiveX Automation Server because it is not required for requirements linking with IBM Rational DOORS Next Generation. Skip this step by pressing N.
- 3 If you have IBM Rational DOORS installed, you are prompted to select the installation path to configure for communication with MATLAB. If you are using RMI with IBM Rational DOORS Next Generation only, you do not need to configure IBM Rational DOORS. Skip this step by pressing 3.
- 4 Configure RMI for integration with IBM Rational DOORS Next Generation server by pressing Y.
- 5 Complete the setup by entering the IBM Rational DOORS Next Generation **Server Address** and the **Port Number** in the next prompt.

Install the Simulink Requirements Widget in IBM Rational DOORS Next Generation

Installing the **Simulink Requirements** widget enables you to propagate selection information from IBM Rational DOORS Next Generation to MATLAB.

In the Windows File Explorer, navigate to the folder `toolbox\slrequirements\slrequirements\resources` in your MATLAB installation folder. Copy the `dnsslk` folder into the `extensions` subfolder of your IBM Rational DOORS Next Generation installation. The location of this folder depends on your server version.

After copying the `dnsslk` folder to your server, add the **Simulink Requirements** widget to the **Mini Dashboard** in IBM Rational DOORS Next Generation. In the **Mini Dashboard**, select **Add Widget > Add OpenSocial Gadget**. Specify the URL to `dnsslk.xml` that corresponds to the `extensions\dnsslk` subfolder in your server installation folder.

For example, if you have Liberty server version 6.0.2 installed, the `extensions` subfolder is located in: `JazzTeamServer_6.0.02\server\liberty\servers\clm\dropins\war\extensions`. The corresponding URL for adding the widget is: `https://JAZZSERVERNAME:9443/extensions/dnsslk/dnsslk.xml`.

Requirements Link Storage

The Requirements Management Interface (RMI) stores the requirements links associated with your Simulink models in two modes - internal and external. When you create links from a model to requirements, by default, the Requirements Management Interface (RMI) stores the link information in an external `.slmx` file in the same folder as the model. External storage does not modify your model when creating or modifying requirements links.

To specify the requirements link storage setting:

- 1** In the model window, select **Analysis > Requirements > Settings**.
- 2** In the Requirements Settings dialog box, select the **Storage** tab.
- 3** Under **Default storage location for requirements links data**:
 - To enable internal storage, select **Store internally (embedded in Simulink diagram file)**.
 - To enable external storage, select **Store externally (in a separate *.slmx file)**.

This setting goes into effect immediately and applies to all new models and to existing models with no links to requirements.

If you open a model that already has requirements links, the RMI uses the storage mechanism you used previously with that model, regardless of what your default storage setting is.

When links are stored with the model (internal storage), the time stamp and version number of the model changes every time you modify your requirements links.

Save Requirements Links in External Storage

The Requirements Management Interface (RMI) stores externally stored requirements links in a file whose name is based on the model file. Because of this, before you create requirements links to be stored in an external file, you must save the model with a value file name.

You add, modify, and, delete requirements links in external storage the same way you do when the requirements links are stored in the model file. The main difference is when you change externally stored links, the model file does not change. The asterisk in the title bar of the model window that indicates a model has unsaved changes does not appear

when you change requirements links. However, when you close the model, the RMI asks if you want to save the requirements links modifications.

There are several ways to save requirements links that are stored in an external file, as listed in the following table.

Select...	To...
Analysis > Requirements > Links File > Save Links As	Save the requirements links in an external file using a file name that you specify. The model itself is not saved.
Analysis > Requirements Traceability > Links File > Save Links	Save the requirements links in an external file using the default file name, <i>model_name.slmx</i> , or to the previously specified file. The model itself is not saved.
File > Save	Save the current requirements links to an external file named <i>model_name.slmx</i> , or to the previously specified file. Any changes you have made to the model are also saved.
File > Save As	Rename and save the model and the external requirements links. The external file is saved as <i>new_model_name.slmx</i> .

Load Requirements Links from External Storage

When you open a Simulink model that does not have internally stored requirements links, the RMI tries to load requirements links from a *.slmx* file — either the default file or a previously specified file. If that file does not exist, the RMI assumes that this model has no requirements links.

To explicitly load requirements links from an external file:

- 1 Select **Analysis > Requirements > Links File > Load Links**.

The Select a file to load RMI data dialog box opens, with the default file name or the previously used file name loaded into the **File name** field.

- 2 Select the file from which to load the requirements links.
- 3 Click **Open** to load the links from the selected file.

Caution If your model has unsaved changes to requirements links and you try to load another file, a warning appears.

Move Internally Stored Requirements Links to External Storage

If you have a model with requirements links that are stored with the model, you can move those links to an external file. When you move internally stored links to a file, the RMI deleted the internally stored links from the model file and saves the model. From this point on, the data exists only in the external file.

- 1 Open the model that contains internally stored requirements links.
- 2 Select **Analysis > Requirements > Links File > Save Links**.

The Select a file to store RMI data dialog box prompts you to save the file with the default name *model_name.slmx*.

- 3 Accept the default name, or enter a different file name if required.
- 4 Click **Save**.

Note Use the default name for externally stored requirements. For more information about this recommendation, see “Guidelines for External Storage of Requirements Links” on page 5-8.

Move Externally Stored Requirements Links to the Model File

If you have a model with requirements links that are stored in an external file, you can move those links to the model file.

- 1 Open the model that has only externally stored requirements links.
- 2 Make sure the right set of requirements links are loaded from the external file.
- 3 Select **Analysis > Requirements > Link File > Copy to Model**.

An asterisk appears next to the model name in the title bar of the model window indicating that your model now has unsaved changes.

- 4 Save the model with the requirements links.

From this point on, the RMI stores all requirements links internally, in the model file. When you add, modify, or delete links, the changes are stored with the model, even if the

Default storage location for requirements links data option is set to **Store externally (in a separate *.slmx file)**.

External Storage

The first time you create links to requirements in a Simulink model, the RMI uses your designated storage preference. When you reopen the model, the RMI loads the internally stored links, or the links from the external file, as long as the file exists with the same name and location as when you last saved the links.

The RMI allows you to save your links file as a different name or in a different folder. However, when you start with the links file in a nondefault location, you must manually load those links into the model. After you load those links, the RMI associates that model with that file and loads the links automatically.

As you work with your model, the RMI stores links using the same storage as the existing links. For example, if you open a model that has internally stored requirements links, any new links you create are also stored internally. This is true even if your preference is set to external storage.

All requirements links must be stored either with the model or in an external file. You cannot mix internal and external storage within a given model.

To see an example of the external storage capability using a Simulink model, at the MATLAB command line, enter:

```
slvnvdemo_powerwindow_external
```

Guidelines for External Storage of Requirements Links

Follow these guidelines when storing requirements links in an external file.

- **When sharing models, use the default name and location.**

By default, external requirements are stored in a file named *model_name.slmx* in the same folder as the model. If you give your model to others to review the requirements traceability, give the reviewer both the model and *.slmx* files. That way, when you load the model, the RMI automatically loads the links file.

- **Do not rename the model outside of Simulink.**

If you need to resave the model with a new name or in a different location, in the model window, use **File > Save As**. Selecting this option causes the RMI to resave the corresponding `.slmx` file using the model name and in the same location as the model.

- **Be aware of unsaved requirements changes.**

When you change a Simulink model, an asterisk appears next to the model name in the title bar, indicating that the model has unsaved changes. If you are creating new requirements links and storing them externally, this asterisk does not appear because the model file itself has not changed. You can explicitly save the links, or, when you close the model, the RMI prompts you to save the requirements links. When you save the model, the RMI saves the links in the external file.

Supported Requirements Document Types

The Requirements Management Interface (RMI) supports linking with external documents of the types listed in the table below. For each supported requirements document type, the table lists the options for requirements locations within the document.

If you would like to implement linking with a requirements document of a type that is not listed in the table below, you can register a custom requirements document type with the RMI. For more information, see “Create a Custom Requirements Link Type” on page 10-10.

Requirements Document Type	Location Options
Microsoft Word 2003 or later	<ul style="list-style-type: none"> • Named item — A bookmark name. The RMI links to the location of that bookmark in the document. The most stable location identifier because the link is maintained when the target content is modified or moved. • Search text — A search string. The RMI links to the first occurrence of that string in the document. This search is not case sensitive. • Page/item number — A page number. The RMI links to the top of the specified page.
Microsoft Excel 2003 or later	<ul style="list-style-type: none"> • Named item — A named range of cells. The RMI links to that named item in the workbook. The most stable location identifier because the link is maintained when the target content is modified or moved. • Search text — A search string. The RMI links to the first occurrence of that string in the workbook. This search is not case sensitive. • Sheet range — A cell location in a workbook: <ul style="list-style-type: none"> • Cell number (A1, C13) • Range of cells (C5:D7) • Range of cells on another worksheet (Sheet1!A1:B4) <p>The RMI links to that cell or cells.</p>

Requirements Document Type	Location Options
IBM Rational DOORS	Page/item number — The unique numeric ID of the target DOORS object. The RMI links to that object.
Text	<ul style="list-style-type: none"> • Search text — A search string. The RMI links to the first occurrence of that string within the document. This search is not case sensitive. • Line number — A line number. The RMI links to the beginning of that line.
HTML	<p>You can link only to a named anchor.</p> <p>For example, in your HTML requirements document, if you define the anchor</p> <pre style="text-align: center;"> ...contents... </pre> <p>in the Location field, enter <code>valve_timing</code> or, from the document index, choose the anchor name.</p> <p>Select the Document Index tab in the “Outgoing Links Editor” on page 10-7 to see available anchors in an HTML file.</p>
Web browser URL	<p>The RMI can link to a URL location. In the Document field, type the URL string. When you click the link, the document opens in a Web browser:</p> <ul style="list-style-type: none"> • Named item — An anchor name. The RMI links to that location on the Web page at that URL.
PDF	<p>Navigation will open a PDF document but will not scroll to a specific page or bookmark.</p> <p>The RMI cannot create a document index of bookmarks in PDF files.</p>

Requirements Settings

You can manage your RMI preferences in the Requirements Settings dialog box. These settings are global and not associated with any particular model. To open the Requirements Settings dialog box, from the Simulink Editor, select **Analysis > Requirements > Settings**. In this dialog box, you can select the:

- **Storage** tab to set the default way in which the RMI stores requirements links in a model. For storage information, see “Requirements Link Storage” on page 5-5.
- **Selection Linking** tab to set the options for linking to the active selection in a supported document. For setting information, see “Selection Linking Tab” on page 5-12.
- **Filters** tab to set the options for filtering requirements in a model. For filtering information, see “Configure Requirements Filtering” on page 5-19.
- **Report** tab to customize the requirements report without using the Report Generator. For setting information, see “Customize Requirements Report Using the RMI Settings” on page 11-25.

Selection Linking Tab

In the Requirements Settings dialog box, on the **Selection Linking** tab, use the following options for linking to the active selection in a supported document. To open the Requirements Settings dialog box, select **Analysis > Requirements > Settings**.

Options	Description
For linking to the active selection within an external document:	
Enabled applications	Enable selection-based linking shortcuts to Microsoft Word, Microsoft Excel, or DOORS applications.
Document file reference	Select type of file reference. For information on what settings to use, see “Document Path Storage” on page 11-48.
Apply this keyword to new links	Enter text to attach to the links you create. For more information about user tags, see “Filter Requirements with User Tags” on page 5-13.

Options	Description
When creating selection-based links:	
Modify destination for bidirectional linking	Creates links both to and from selected link destination.
Store absolute path to model file	Select to store the absolute path to the Simulink model file.
Use custom bitmap for navigation controls in documents	Select and browse for your bitmap. You can use your own bitmap file to control the appearance of navigation links in your document.
Use ActiveX buttons in Word and Excel (backward compatibility)	Select to use legacy ActiveX controls to create links in Microsoft Word and Microsoft Excel applications. By default, if not selected, you create URL-based links.

Filter Requirements with User Tags

- “User Tags and Requirements Filtering” on page 5-13
- “Apply a User Tag to a Requirement” on page 5-13
- “Filter, Highlight, and Report with User Tags” on page 5-15
- “Apply User Tags During Selection-Based Linking” on page 5-17
- “Configure Requirements Filtering” on page 5-19

User Tags and Requirements Filtering

User tags are user-defined keywords that you associate with specific requirements. With user tags, you can highlight a model or generate a requirements report for a model in the following ways:

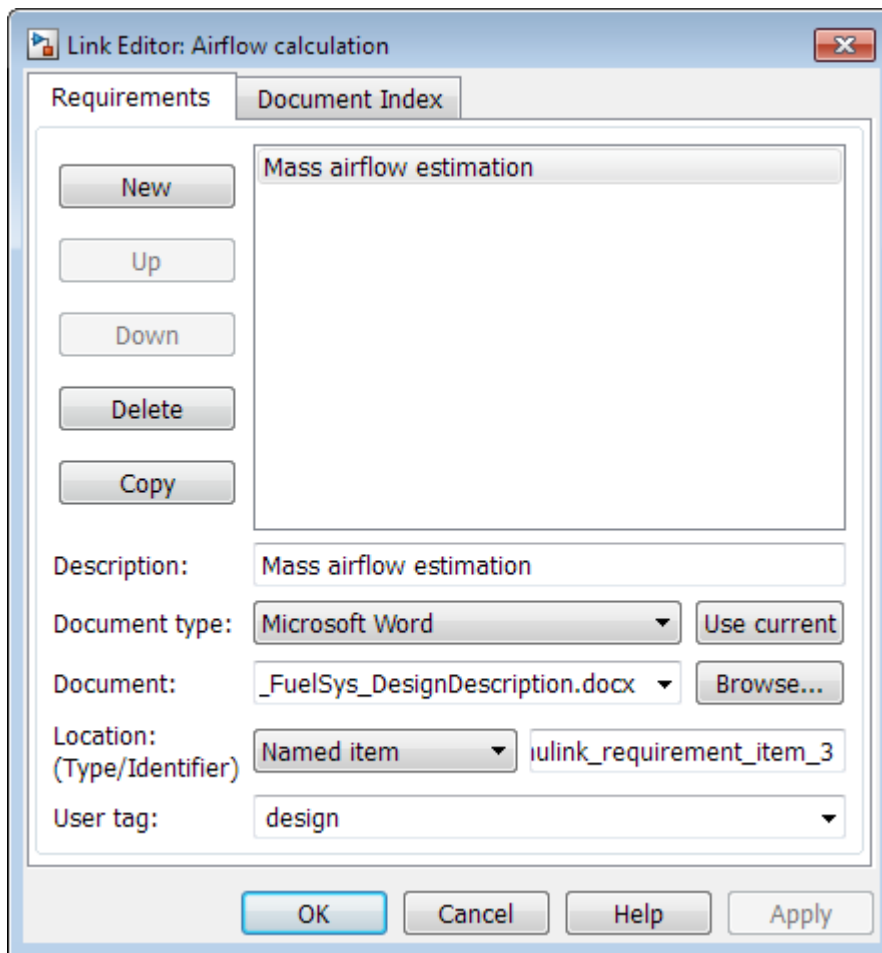
- Highlight or report only those requirements that have a specific user tag.
- Highlight or report only those requirements that have one of several user tags.
- Do not highlight and report requirements that have a specific user tag.

Apply a User Tag to a Requirement

To apply one or more user tags to a newly created requirement:

- 1 Open the example model:
slvndemo_fuelsys_officereq
- 2 Open the fuel rate controller subsystem.
- 3 To open the requirements document, right-click the Airflow calculation subsystem and select **Requirements > Open Link Editor**.

The Requirements Traceability Link Editor opens with the details about the requirement that you created.



- 4 In the **User tag** field, enter one or more keywords, separated by commas, that the RMI can use to filter requirements. In this example, after `design`, enter a comma, followed by the user tag `test` to specify a second user tag for this requirement.

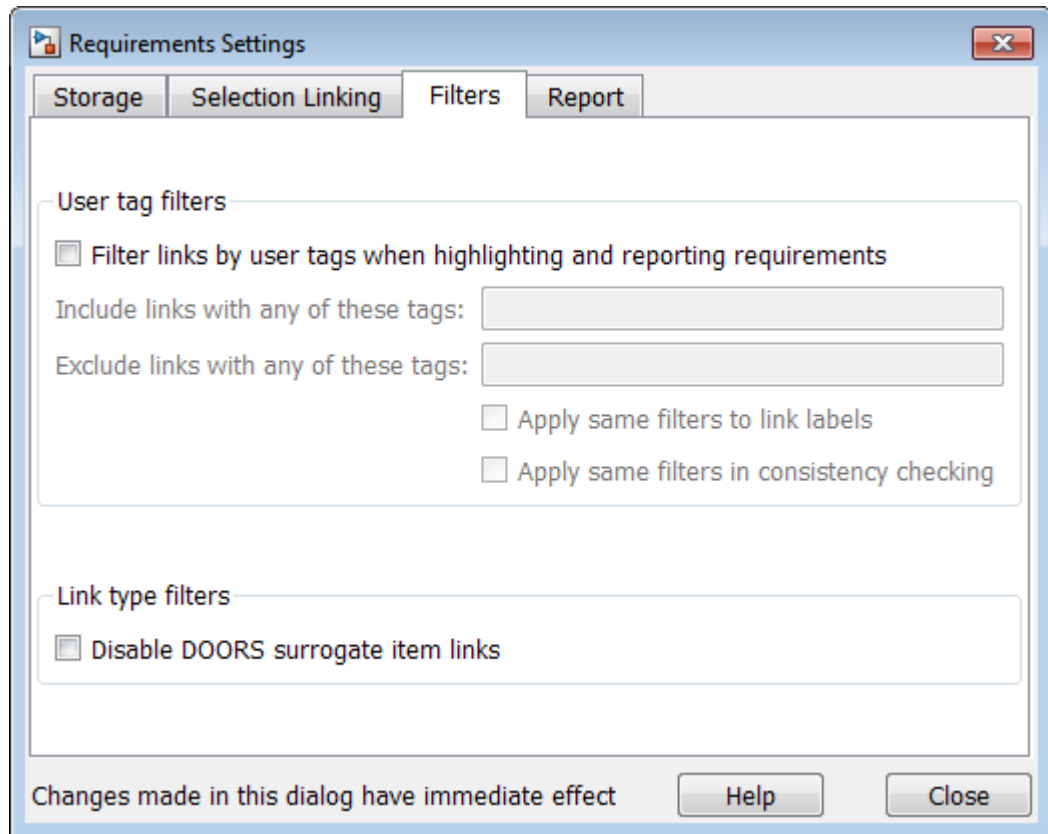
User tags:

- Are not case sensitive.
 - Can consist of multiple words. For example, if you enter `design requirement`, the entire phrase constitutes the user tag. Separate user tags with commas.
- 5 Click **Apply** or **OK** to save the changes.

Filter, Highlight, and Report with User Tags

The `slvndemo_fuelsys_officereq` model includes several requirements with the user tag `design`. This section describes how to highlight only those model objects that have the user tag, `test`.

- 1 In the Simulink Editor, remove highlighting from the `slvndemo_fuelsys_officereq` model by selecting **Analysis > Requirements > Unhighlight model**.
- 2 Select **Analysis > Requirements > Settings**.
- 3 In the Requirements Settings dialog box, click the **Filters** tab.



- 4 To enable filtering with user tags, click the **Filter links by user tags when highlighting and reporting requirements** option.
- 5 To include only those requirements that have the user tag, test, enter test in the **Include links with any of these tags** field.
- 6 Click **Close**.
- 7 In the Simulink Editor, select **Analysis > Requirements > Highlight model**.

The RMI highlights only those model objects whose requirements have the user tag test, for example, the MAP sensor.

- 8 Reopen the Requirements Settings dialog box to the **Filters** tab.
- 9 In the **Include links with any of these tags** field, delete test. In the **Exclude links with any of these tags** field, add test.

In the model, the highlighting changes to exclude objects whose requirements have the `test` user tag. The MAP sensor and Test inputs blocks are no longer highlighted.

- 10** In the Simulink Editor, select **Analysis > Requirements > Generate Report**.

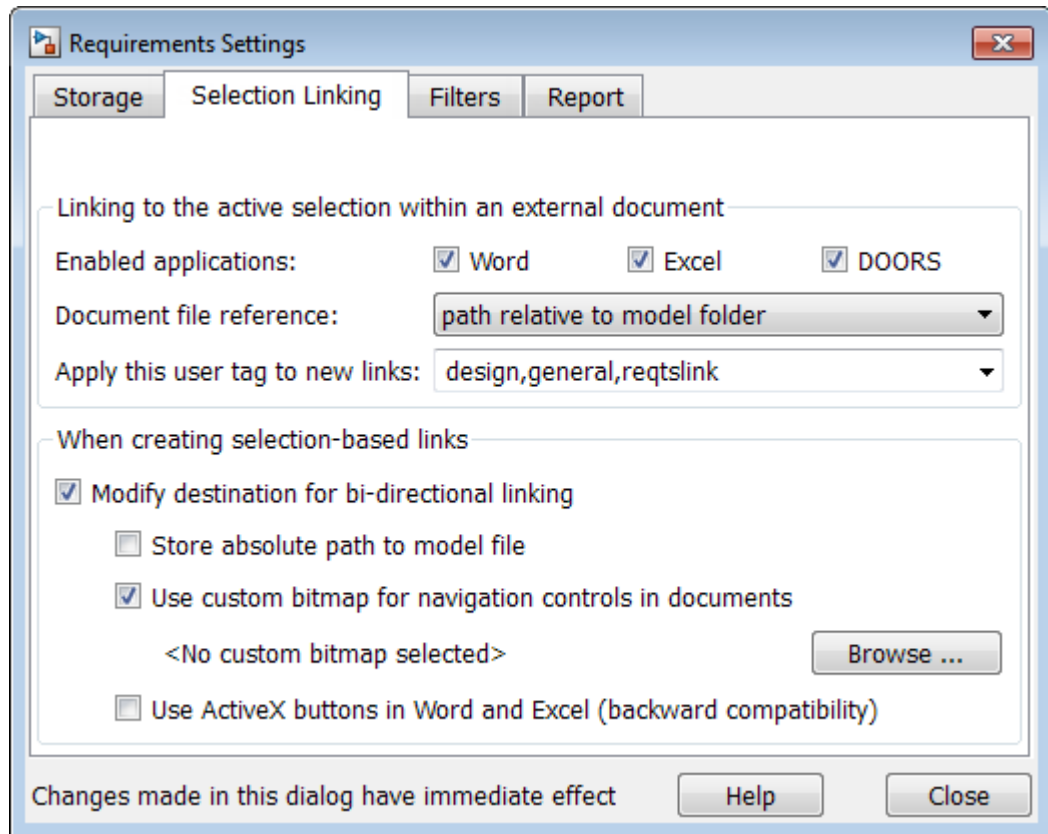
The report does not include information about objects whose requirements have the `test` user tag.

Apply User Tags During Selection-Based Linking

When creating a succession of requirements links, you can apply the same user tags to all links automatically. This capability, also known as selection-based linking, is available only when you are creating links to selected objects in the requirements documents.

When creating selection-based links, specify one or more user tags to apply to requirements:

- 1** In the Simulink Editor, select **Analysis > Requirements > Settings**.
- 2** Select the **Selection Linking** tab.

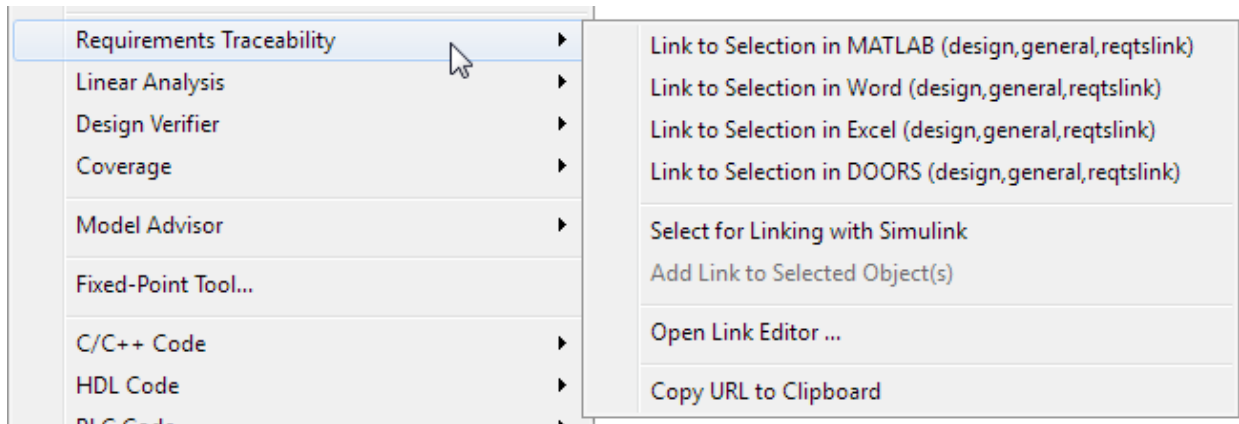


- 3 In the **Apply this user tag to new links** field, enter one or more user tags, separated by commas.

The RMI applies these user tags to all new selection-based requirements links that you create.

- 4 Click **Close** to close the Requirements Settings dialog box.
- 5 In a requirements document, select the specific requirement text.
- 6 Right-click a model object and select **Requirements**.

The selection-based linking options specify which user tags the RMI applies to the link that you create. In the following example, you can apply the user tags `design`, `general`, and `reqtslink` to the link that you create to your selected text.



Configure Requirements Filtering

In the Requirements Settings dialog box, in the **Filters** tab, use the following options for filtering requirements in a model.

Option	Description
Filter links by user tags when highlighting and reporting requirements	Enables filtering for highlighting and reporting, based on specified user tags.
Include links with any of these tags	Includes information about all requirements that have any of the specified user tags. Separate multiple user tags with commas.
Exclude links with any of these tags	Excludes information about all requirements that have any of the specified user tags. Separate multiple user tags with commas or spaces.
Apply same filters to link labels	Disables link labels in context menus if any of the specified filters are satisfied, for example, if a requirement has a designated user tag.
Apply same filters in consistency checking	Includes or excludes requirements with specified user tags when running a consistency check between a model and its associated requirements documents.

Option	Description
Under Link type filters , Disable synchronization item links in context menus	Disables links to IBM Rational DOORS surrogate items from the context menus when you right-click a model object. This option does not depend on current user tag filters.

Microsoft Office Traceability

- “Link to Requirements in Microsoft Word Documents” on page 6-2
- “Link to Requirements in Microsoft Excel Workbooks” on page 6-8
- “Navigate to Requirements in Microsoft Office Documents from Simulink” on page 6-12

Link to Requirements in Microsoft Word Documents

Create Bookmarks in a Microsoft Word Requirements Document

You can create bookmarks in your Microsoft Word requirements documents to identify the requirements that you want to link to. When you create the links, you specify that the RMI must link to an existing bookmark, rather than create a new bookmark.

This approach offers advantages to creating new bookmarks:

- You can give the bookmarks meaningful names that represent the content of the requirement.
- When the RMI creates the links, it does not modify your requirements document.

Note When you link to an existing bookmark, navigating the link highlights the entire range of the existing bookmark. Therefore, when you create a bookmark for requirement linking, make sure to select only the document information relevant to your requirement.

If you have a requirements document containing bookmarks, follow these steps to create requirements links from your Simulink model to the bookmarks:

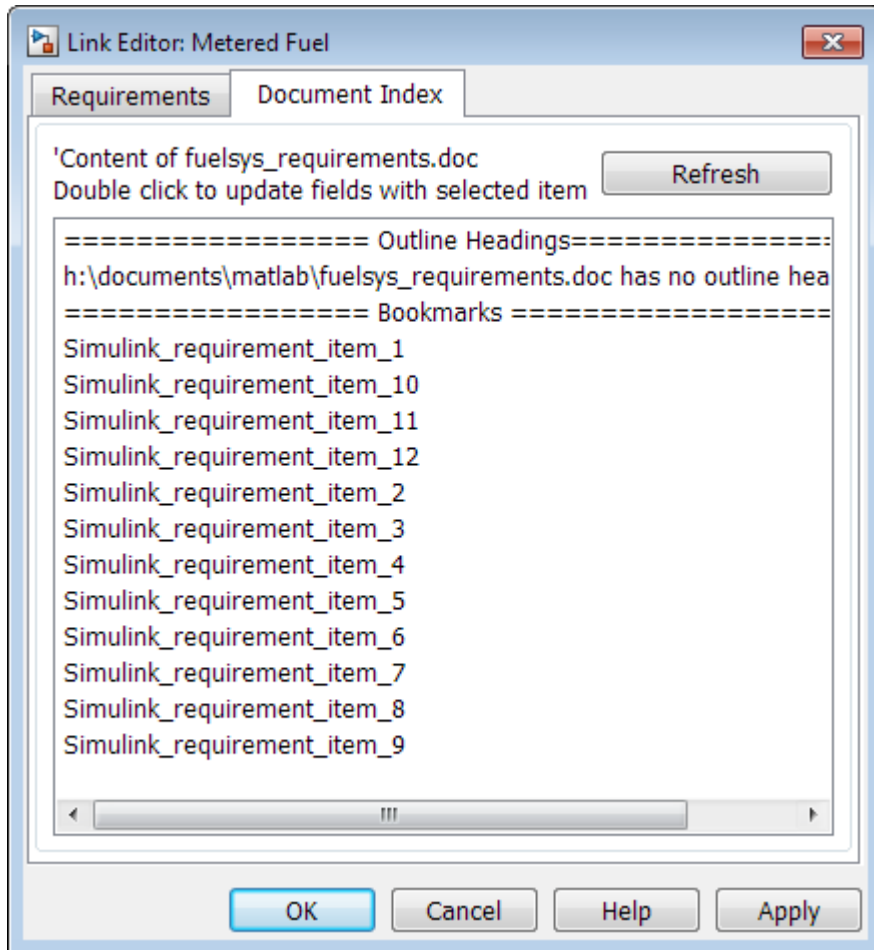
- 1 Open your model.
- 2 Open your Microsoft Word requirements document that contains bookmarks that identify requirements.
- 3 Right-click a block in the model that you want to link to a requirement and select **Requirements > Open Link Editor**

The Requirements Traceability Link Editor opens.

- 4 Click **New**.
- 5 Click **Browse** and navigate to the Microsoft Word requirements document that has bookmarks.
- 6 Open the document. The RMI populates the **Document** and **Document type** fields.
- 7 Click the **Document Index** tab of the Link Editor.

The **Document Index** tab lists all bookmarks in the requirements document, as well as all section headings (text that you have styled as **Heading 1**, **Heading 2**, and so on).

The document index lists the bookmarks in alphabetical order, not in order of location within the document.



- 8 Select the bookmark that you want to link the block to and click **OK**.

The RMI creates a link from the block to the location of the bookmark in the requirements document without modifying the document itself.

Open the Example Model and Associated Requirements Document

This example describes how to create links from objects in a Simulink model to selected requirements text in a Microsoft Word document.

Navigate from the model to the requirements document:

- 1 Open the example model:

```
slvndemo_fuelsys_officereq
```
- 2 Open a requirements document associated with that model:

```
rmi('view','slvndemo_fuelsys_officereq',1);
```

Keep the example model and the requirements document open.

Create a Link from a Model Object to a Microsoft Word Requirements Document

Create a link from the Airflow calculation subsystem in the `slvndemo_fuelsys_officereq` model to selected text in the requirements document:

- 1 In `slvndemo_FuelSys_DesignDescription.docx`, find the section titled **2.2 Determination of pumping efficiency**.
- 2 Select the header text.
- 3 Open the example model:

```
slvndemo_fuelsys_officereq
```
- 4 Select **Analysis > Requirements > Settings**. The Requirements Settings dialog box opens.
- 5 On the **Selection Linking** tab of the Requirements Settings dialog box:
 - Set the **Document file reference** option to `path relative to model folder`.

- Enable **Modify destination for bidirectional linking**.

When you select this option, every time you create a selection-based link from a model object to a requirement, the RMI inserts navigation objects at the designated location.

For more information about the settings, see “Requirements Settings” on page 5-12.

- 6 Double-click the fuel rate controller subsystem to open it.
- 7 Open the Airflow calculation subsystem.
- 8 Right-click the Pumping Constant block and select **Requirements > Link to Selection in Word**.

The RMI inserts a bookmark at that location in the requirements document and assigns it a generic name, in this case, Simulink_requirement_item_7.

Note You cannot link to a Microsoft Word document from a model object if you run either one of MATLAB or Microsoft Word as an Administrator. Run both software with the same privilege level to link to Microsoft Word requirements documents.

- 9 To verify that the link was created, select **Analysis > Requirements > Highlight Model**.

The Pumping Constant block, and other blocks with requirements links, are highlighted.

- 10 To navigate to the link, right-click the Pumping Constant block and select **Requirements > 1. “Determination of pumping efficiency”**.

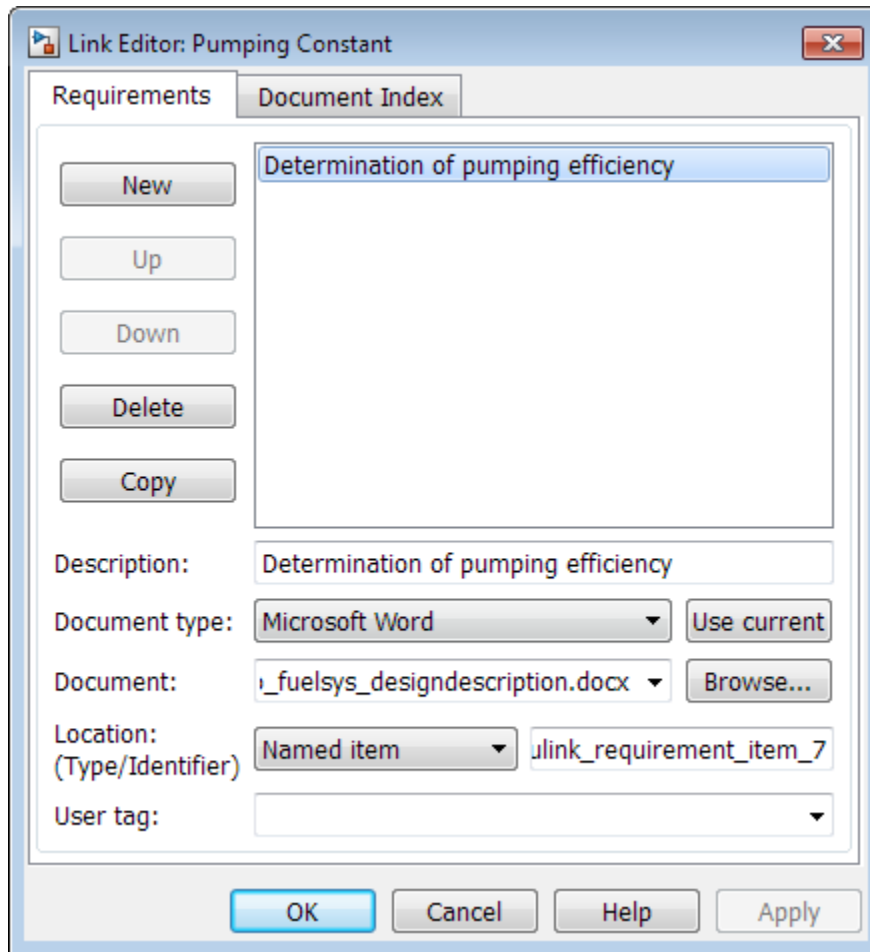
The section **2.2 Determination of pumping efficiency** is displayed, selected in the requirements document.

Keep the example model and the requirements document open.

View Link Details

To view the details of the link that you just created, right-click the Pumping Constant block and select **Requirements > Open Link Editor**.

The Requirements Traceability Link Editor opens.



This dialog box contains the following information for the link you just created:

- Description of the link, which for selection-based links, matches the text of the selected requirements document, in this case **Determination of pumping efficiency**.
- Name of the requirements document, in this case **slvnvdemo_FuelSys_DesignDescription.docx**.
- Document type, in this case, **Microsoft Word**.

- The type and identifier of the location in the requirements document. With selection-based linking for Microsoft Word requirements documents, the RMI creates a bookmark in the requirements document. For this link, the RMI created a bookmark named Simulink_requirement_item_7.

If you do not want the RMI to modify the Microsoft Word requirements document when it creates links, create bookmarks in your Microsoft Word file, as described in “Create Bookmarks in a Microsoft Word Requirements Document” on page 6-2.

- User tag, a user-defined keyword. This link does not have a user tag.

Note For more information about user tags, see “Filter Requirements with User Tags” on page 5-13.

Link to Requirements in Microsoft Excel Workbooks

Navigate from a Model Object to Requirements in a Microsoft Excel Workbook

- 1 Open the example model:
`slvndemo_fuelsys_officereq`
- 2 Select **Analysis > Requirements > Highlight Model** to highlight the model objects with requirements.
- 3 Right-click the Test inputs Signal Builder block and select **Requirements > 1. "Normal mode of operation"**.

The `slvndemo_FuelSys_TestScenarios.xlsx` file opens, with the associated cell highlighted.

Keep the example model and Microsoft Excel requirements document open.

For information about creating requirements links in Signal Builder blocks, see “Link Signal Builder Blocks to Requirements and Simulink Model Objects” on page 8-10.

Create Requirements Links to the Workbook

- 1 At the top level of the `slvndemo_fuelsys_officereq` model, right-click the speed sensor block and select **Requirements > Open Link Editor**.

The Requirements Traceability Link Editor opens.

- 2 To create a requirements link, click **New**.
- 3 In the **Description** field, enter:

Speed sensor failure

You will link the speed sensor block to the **Speed Sensor Failure** information in the Microsoft Excel requirements document.

- 4 When you browse and select a requirements document, the RMI stores the document path as specified by the **Document file reference** option on the Requirements Settings dialog box, **Selection Linking** tab.

For information about which setting to use for your working environment, see “Document Path Storage” on page 11-48.

- 5 At the **Document** field, click **Browse** to locate and open the `slvndemo_FuelSys_TestScenarios.xlsx` file.

The **Document Type** field information changes to Microsoft Excel.

- 6 In the workbook, the **Speed sensor failure** information is in cells B22:E22. For the **Location (Type/Identifier)** field, select Sheet range and in the second field, enter B22:E22. (The cell range letters are not case sensitive.)
- 7 Click **Apply** or **OK** to create the link.
- 8 To confirm that you created the link, right-click the speed sensor block and select **Requirements > 1. “Speed sensor failure”**.

The workbook opens, with cells B22:E22 highlighted.

Keep the example model and Microsoft Excel requirements document open.

Link Multiple Model Objects to a Microsoft Excel Workbook

You can use the same technique to link multiple Simulink and Stateflow objects to a requirement in a Microsoft Excel workbook. The workflow is:

- 1 In the model window, select the objects to link to a requirement.
- 2 Right-click one of the selected objects and select **Requirements > Open Link Editor**.
- 3 When you browse and select a requirements document, the RMI stores the document path as specified by the **Document file reference** option on the Requirements Settings dialog box, **Selection Linking** tab.

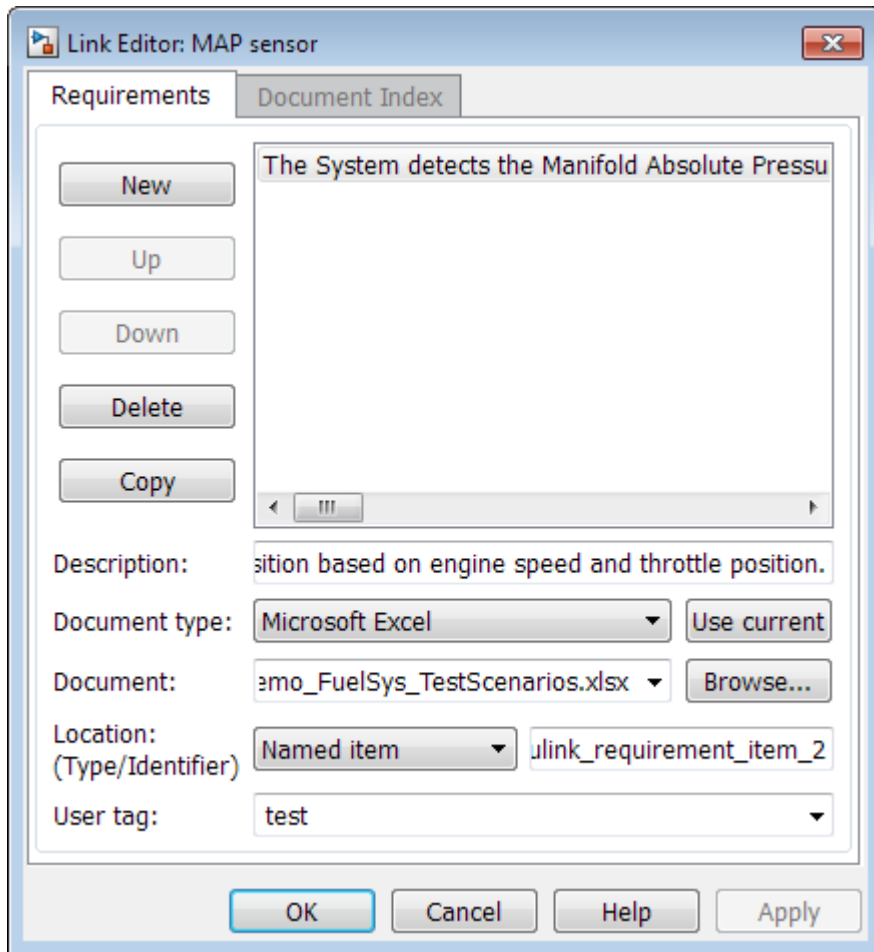
For information about which setting to use for your working environment, see “Document Path Storage” on page 11-48.

- 4 Use the Link Editor to specify information about the Microsoft Excel requirements document, the requirement, and the link.
- 5 Click **Apply** or **OK** to create the link.

Change Requirements Links

- 1 In the `slvndemo_fuelsys_officereq` model, right-click the MAP sensor block and select **Requirements > Open Link Editor**.

The Requirements Traceability Link Editor opens displaying the information about the requirements link.



- 2 In the **Description** field, enter:
MAP sensor test scenario

The **User tag** field contains the tag `test`. User tags filter requirements for highlighting and reporting.

Note For more information about user tags, see “Filter Requirements with User Tags” on page 5-13.

- 3** Click **Apply** or **OK** to save the change.

Keep the example model open.

Navigate to Requirements in Microsoft Office Documents from Simulink

Enable Linking from Microsoft Office Documents to Simulink Objects

You can use the Microsoft Word and Microsoft Excel applications to capture, track, and manage requirements. The Requirements Management Interface (RMI) allows you to link Simulink objects to requirements managed by external applications.

When you create a link from a Simulink object to a requirement in a Microsoft Office document, the RMI stores the link data in Simulink. Using this link, you can navigate from the Simulink object to its associated requirement.

You can also configure the RMI to insert a navigation object in a Microsoft Office requirements document. This navigation object serves as a link from the requirement to its associated Simulink object.

By default, the RMI does not insert navigation objects into requirements documents. If you want to insert a navigation object into the requirements document when you create a link from a Simulink object to a requirement, you must change the RMI's settings. The following tutorial uses the `slvndemo_fuelsys_officereq` example model to illustrate how to do this.

The RMI can insert navigation objects into the following Microsoft Office applications:

- Microsoft Excel
- Microsoft Word

To enable linking from a Microsoft Office document to the example model:

- 1 Open the model:

```
slvndemo_fuelsys_officereq
```

Note You can modify requirements settings in the Requirements Settings dialog box. These settings are global and not specific to open models. Changes you make apply not only to open models, but also persist for models you subsequently open. For more information about these settings, see “Requirements Settings” on page 5-12.

- 2 Select **Analysis > Requirements > Settings**.

The Requirements Settings dialog box opens.

- 3 On the **Selection Linking** tab of the Requirements Settings dialog box:

- Enable **Modify destination for bidirectional linking**.

When you select this option, every time you create a selection-based link from a Simulink object to a requirement, the RMI inserts a navigation object at the designated location in the requirements document.

- To specify one or more user tags to apply to the links that you create, in the **Apply this keyword to new links** field, enter the tag names.

For more information about user tags, see “User Tags and Requirements Filtering” on page 5-13.

- 4 Click **Close** to close the Requirements Settings dialog box. Keep the `slvndemo_fuelsys_officereq` model open.

Insert Navigation Objects in Microsoft Office Requirements Documents

Use selection-based linking to create a link from the `slvndemo_fuelsys_officereq` model to a requirements document. If you have configured the RMI as described in “Enable Linking from Microsoft Office Documents to Simulink Objects” on page 6-12, the RMI can insert a navigation object into the requirements document.

- 1 Open the Microsoft Word requirements document:

```
matlabroot/toolbox/slvnv/rmidemos/fuelsys_req_docs/
slvndemo_FuelSys_RequirementsSpecification.docx
```

- 2 Select the **Throttle Sensor** header.
- 3 In the `slvndemo_fuelsys_officereq` model, open the engine gas dynamics subsystem.
- 4 Right-click the Throttle & Manifold subsystem and select **Requirements > Link to Selection in Word**.
- 5 The RMI inserts an URL-based link into the requirements document.

1.1.6. Throttle Sensor

Insert Navigation Object That Links to Multiple Simulink Objects

If you have several Simulink objects that correspond to one requirement, you can link them all to that requirement with a single navigation object. This eliminates the need to insert multiple navigation objects for a single requirement. The Simulink objects must be available in the same model diagram or Stateflow chart.


The workflow for linking multiple Simulink objects to one Microsoft Word requirement is as follows:

- 1 Make sure that the RMI is configured to insert navigation objects into requirements documents, as described in “Enable Linking from Microsoft Office Documents to Simulink Objects” on page 6-12.
- 2 Select the Microsoft Word requirement to link to.
- 3 Select the Simulink objects that need to link to that requirement.
- 4 Right-click one of the Simulink objects and select **Requirements > Link to Selection in Word**.

A single navigation object is inserted at the selected requirement.

- 5 Follow the navigation object link in Microsoft Word to highlight the Simulink objects that are linked to that requirement.

Customize Microsoft Office Navigation Objects

If the RMI is configured to modify destination for bidirectional linking, the RMI inserts a navigation object into your requirements document. This object looks like the icon for the Simulink software: 

Note In Microsoft Office requirements documents, following a navigation object link highlights the Simulink object that contains a bidirectional link to the associated requirement.

To use an icon of your own choosing for the navigation object:

- 1 Select **Analysis > Requirements > Settings**.
- 2 Select the **Selection Linking** tab.
- 3 Select **Modify destination for bidirectional linking**.

Selecting this option enables the **Use custom bitmap for navigation controls in documents** option.

- 4 Select **Use custom bitmap for navigation controls in documents**.
- 5 Click **Browse** to locate the file you want to use for the navigation objects.

For best results, use an icon file (.ico) or a small (16×16 or 32×32) bitmap image (.bmp) file for the navigation object. Other types of image files might give unpredictable results.

- 6 Select the desired file to use for navigation objects and click **Open**.
- 7 Close the Requirements Settings dialog box.

The next time you insert a navigation object into a requirements document, the RMI uses the file you selected.

Navigate Between Microsoft Word Requirement and Model

In “Insert Navigation Objects in Microsoft Office Requirements Documents” on page 6-13, you created a link between a Microsoft Word requirement and the Throttle & Manifold subsystem in the slvndemo_fuelsys_officereq example model. Navigate these links in both directions:

- 1 In the slvndemo_fuelsys_officereq model, right-click the Throttle & Manifold subsystem and select **Requirements > 1. “Throttle Sensor”**.

The requirements document opens, and the header in the requirements document is highlighted.

1.1.6. Throttle Sensor


- 2 In the requirements document, next to **Throttle Sensor**, follow the navigation object link.

The engine gas dynamics subsystem opens, with the Throttle & Manifold subsystem highlighted.



Navigation from Microsoft Office requirements documents is not automatically enabled upon MATLAB startup. Navigation is enabled when you create a new requirements link or when you have enabled bidirectional linking as described in “Insert Navigation Objects in Microsoft Office Requirements Documents” on page 6-13.

Note You cannot navigate to requirements from Microsoft Word 2013 onwards when the document is open in read-only mode. Alternately, consider disabling the “Open e-mail attachments and other uneditable files in reading view” option in the Microsoft Word options or using editable documents.

When attempting navigation from requirements links with the  icon, if you get a “Server Not Found” or similar message, enter the command `rmi('httpLink')` to activate the internal MATLAB HTTP server.

Requirements Traceability with IBM Rational DOORS

- “Configure Requirements Management Interface for IBM Rational DOORS Software” on page 7-2
- “Requirements Traceability with IBM Rational DOORS Next Generation” on page 7-4
- “Navigate to Requirements in IBM Rational DOORS Databases from Simulink” on page 7-7
- “Synchronize Simulink Models with IBM Rational DOORS Databases by using Surrogate Modules” on page 7-13

Configure Requirements Management Interface for IBM Rational DOORS Software

Before You Begin

If you plan to use DOORS software with the RMI, make sure to install additional files to establish communication between the DOORS application and the Simulink software. Follow the instructions in “Configure RMI for Interaction with Microsoft Office Applications and IBM Rational DOORS Software” on page 5-2.

Manually Install Additional Files for DOORS Software

The setup script automatically copies the required DOORS files to the installation folders. However, the script might fail because of file permissions in your DOORS installation. If the script fails, change the file permissions on the DOORS installation folders and rerun the script.

You can also manually install the required files into the specified folders, as described in the following steps:

- 1 If the DOORS software is running, close the application.
- 2 Copy the following files from *matlabroot*\toolbox\shared\reqmgt\dxl to the *<doors_install_dir>*\lib\dxl\addins folder.

```
addins.idx  
addins.hlp
```

If you have not modified the files, replace any existing versions of the files; otherwise, merge the contents of both files into a single file.

- 3 Copy the following files from *matlabroot*\toolbox\shared\reqmgt\dxl to the *<doors_install_dir>*\lib\dxl\addins\dmi folder.

```
dmi.hlp  
dmi.idx  
dmi.inc  
runsim.dxl  
selblk.dxl
```

Replace any existing versions of these files.

- 4 Open the `<doors_install_dir>\lib\dxl\startup.dxl` file. In the user-defined files section, add the following `include` statement:

```
#include <addins/dmi/dmi.inc>
```

If you upgrade from Version 7.1 to a later version of the DOORS software, perform these additional steps:

- a In your DOORS installation folder, navigate to the `...\lib\dxl\startupFiles` subfolder.
- b In a text editor, open the `copiedFromDoors7.dxl` file.
- c Add `//` before this line to comment it out:

```
#include <addins/dmi/dmi.inc>
```

- d Save and close the file.

- 5 Start the DOORS and MATLAB software.
- 6 Run the setup script using the following MATLAB command.

```
rmi setup
```

Diagnose and Fix DXL Errors

If you try to synchronize your Simulink model to a DOORS project, you might see the following errors:

```
-E- DXL: <Line:2> incorrectly concatenated tokens  
-E- DXL: <Line:2> undeclared variable (dmiRefreshModule)  
-I- DXL: all done with 2 errors and 0 warnings
```

If you see these errors, exit the DOORS software, rerun the steps in “Configure RMI for Interaction with Microsoft Office Applications and IBM Rational DOORS Software” on page 5-2, and restart the DOORS software.

Requirements Traceability with IBM Rational DOORS Next Generation

You can link and trace Simulink model elements to requirements in IBM Rational DOORS Next Generation. Before you begin, configure IBM Rational DOORS Next Generation for communication with MATLAB by following the instructions in “Install the Simulink Requirements Widget in IBM Rational DOORS Next Generation” on page 5-3. Enable bidirectional requirements traceability with IBM Rational DOORS Next Generation:

- 1 Select **Analysis > Requirements > Settings** in the Simulink Editor.
- 2 Switch to the **Selection Linking** tab and select **DOORS** in the **Enabled applications** field.
- 3 Select **Modify destination for bidirectional linking**.

Link to Requirements in IBM Rational DOORS Next Generation

To link and trace your Simulink model elements to requirements in IBM Rational DOORS Next Generation, use any of these workflows:

- “Link to Requirements by Using the Outgoing Links Editor Dialog Box” on page 7-4
- “Link to Selected Requirements in a Project by Using the Simulink Context Menu” on page 7-5 if you have installed the **Simulink Requirements** widget in IBM Rational DOORS Next Generation.
- “Link to the Requirements in a Project by Using the Numeric ID” on page 7-5 if you cannot use either of the two options.

Link to Requirements by Using the Outgoing Links Editor Dialog Box

- 1 Right-click the Simulink model element to which you want to link IBM Rational DOORS Next Generation requirements.
- 2 Select **Requirements > Open Outgoing Links dialog**.
- 3 In the **Outgoing Links** dialog box, click **New** and select **OSLC Resource** as the **Document type**.
- 4 Click **Browse**.
- 5 Enter your IBM Rational DOORS Next Generation login credentials. From the drop-down list, select the active project name in IBM Rational DOORS Next Generation.
- 6 Switch to the **Document Index** tab and select the requirements that you want to link to from the list of requirements. To create the link, click **OK**.

Link to Selected Requirements in a Project by Using the Simulink Context Menu

Install the **Simulink Requirements** widget in IBM Rational DOORS Next Generation. For more information, see “Install the Simulink Requirements Widget in IBM Rational DOORS Next Generation” on page 5-3

- 1 In IBM Rational DOORS Next Generation, open the **Mini Dashboard** and pin it to the screen.
- 2 Switch to the **Browse Artifacts** view.
- 3 Select the requirements that you want to link to by selecting the check box next to the requirement.

The requirements that you select for linking are displayed in the **Simulink Requirements** widget in the **Mini Dashboard**.

- 4 Right-click the Simulink model element to which you want to link IBM Rational DOORS Next Generation requirements.
- 5 Establish links to the requirements by selecting **Requirements > Link to Current Item in DNG**.

Click **List Projects** in the dialog box that appears and select the requirements from within IBM Rational DOORS Next Generation.

Link to the Requirements in a Project by Using the Numeric ID

Use this option if you are unable to link to requirements by using the Outgoing Links dialog box or by using the Simulink context menu.

- 1 Right-click the Simulink model to which you want to link IBM Rational DOORS Next Generation requirements.
- 2 Select **Requirements > Link to Current Item in DNG**.
- 3 Click **Manual entry** in the dialog box that appears and enter the numeric ID for the link target. Establish links to the requirements by clicking **OK**.

Navigate to Requirements from Simulink

Right-click the Simulink model element that requirements have been linked to. Select **Requirements** and navigate to the corresponding requirement in IBM Rational DOORS Next Generation by clicking the navigation shortcut at the top of the menu.

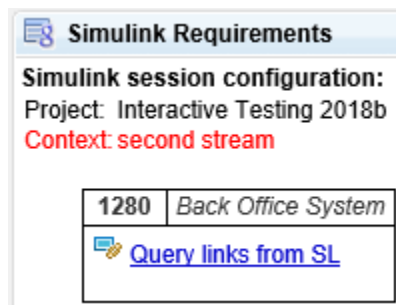
Work with IBM Rational DOORS Next Generation Projects with Configuration Management Enabled

Projects with configuration management enabled in IBM Rational DOORS Next Generation support multiple branches called *streams* and *changesets*. Changesets are akin to shared development branches that can later be merged with the parent main stream. Simulink Requirements enables you to update the outgoing link destination for an existing link in Simulink to the same requirement in a different stream or changeset.

You can select the IBM Rational DOORS Next Generation Project and the configuration stream or changeset you want to work with. At the MATLAB command prompt, enter:

```
oslc.configure
```

The **Simulink Requirements** widget displays information about the current configuration stream context you work with in Simulink Requirements. The widget indicates if there is a mismatch between the active configuration stream contexts in Simulink Requirements and in IBM Rational DOORS Next Generation by highlighting the active configuration stream context in Simulink Requirements.



To resolve the mismatch, click the highlighted text in the widget. Click **Update** in the **DNG Configuration Context Mismatch** dialog box to update the configuration stream context in Simulink Requirements to be consistent with the current configuration stream context in IBM Rational DOORS Next Generation. Alternatively, you can change the active configuration stream in IBM Rational DOORS Next Generation.

Navigate to Requirements in IBM Rational DOORS Databases from Simulink

Enable Linking from IBM Rational DOORS Databases to Simulink Objects

By default, the RMI does not insert navigation objects into requirements documents. If you want to insert a navigation object into the requirements document when you create a link from a Simulink object to a requirement, you must change the RMI's settings. The following tutorial uses the `sldemo_fuelsys` example model to illustrate how to do this.

To enable linking from the DOORS database to the example model:

- 1 Open the model:

```
sldemo_fuelsys
```

Note You can modify requirements settings in the Requirements Settings dialog box. These settings are global and not specific to open models. Changes you make apply not only to open models, but also persist for models you subsequently open. For more information about these settings, see “Requirements Settings” on page 5-12.

- 2 Select **Analysis > Requirements > Settings**.

The Requirements Settings dialog box opens.

- 3 Click the **Selection Linking** tab.
- 4 Select **Modify destination for bidirectional linking**.

When you enable this option, every time you create a selection-based link from a Simulink object to a requirement, the RMI inserts navigation objects at the designated location. Using this option requires write access to the requirements document.

- 5 Select **Store absolute path to model file**.

For this exercise, you save a copy of the example model on the MATLAB path.

If you add requirements to a model that is not on the MATLAB path, you must select this option to enable linking from your requirements document to your model.

- 6 In the **Apply this keyword to new links** field, enter one or more user tags to apply to the links that you create.

For more information about user tags, see “User Tags and Requirements Filtering” on page 5-13.

- 7 Click **Close** to close the Requirements Settings dialog box. Keep the `sldemo_fuelsys` model open.


Insert Navigation Objects into IBM Rational DOORS Requirements


When you enable **Modify destination for bidirectional linking** as described in “Enable Linking from IBM Rational DOORS Databases to Simulink Objects” on page 7-7, the RMI can insert a navigation object into both the Simulink object and its associated DOORS requirement. This tutorial uses the `sldemo_fuelsys` example model to illustrate how to do this. For this tutorial, you also need a DOORS formal module that contains requirements.

- 1 Rename the `sldemo_fuelsys` model and save it in a writable folder on the MATLAB path.
- 2 Start the DOORS software and open a formal module that contains requirements.
- 3 Select the requirement that you want to link to by left-clicking that requirement in the DOORS database.
- 4 In the `sldemo_fuelsys` model, select an object in the model.

This example creates a requirement from the `fuel_rate_control` subsystem.

- 5 Right-click the Simulink object (in this case, the `fuel_rate_control` subsystem) and select **Requirements > Link to Selection in DOORS**.

The RMI creates the link for the `fuel_rate_control` subsystem. It also inserts a navigation object into the DOORS formal module—a Simulink reference object () that enables you to navigate from the requirement to the model.

ID	
1	<p>1 Fuel rate controller requirements</p> <p>The controller will use engine speed, throttle position and manifold pressure to airflow through the engine.</p>
2	<p>[Simulink reference: sldemo_fuelsys/fuel_rate_control (SubSystem)]</p> 

- 6 Close the model.

Note When you navigate to a DOORS requirement from outside the software, the DOORS module opens in read-only mode. If you want to modify the DOORS module, open the module using DOORS software.

Insert Navigation Objects to Multiple Simulink Objects

If you have several Simulink objects that correspond to one requirement, you can link them all to that requirement with a single navigation object. This eliminates the need to insert multiple navigation objects for a single requirement. The Simulink objects must be available in the same model diagram or Stateflow chart.

The workflow for linking multiple Simulink objects to one DOORS requirement is as follows:

- 1 Make sure that you have enabled **Modify destination for bidirectional linking**.
- 2 Select the DOORS requirement to link to.
- 3 Select the Simulink objects that need to link to that requirement.
- 4 Right-click one of the objects and select **Requirements Traceability > Link to Selection in DOORS**.

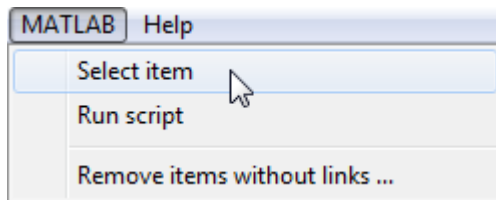
A single navigation object is inserted at the selected requirement.

- 5 Double-click the navigation object in DOORS to highlight the Simulink objects that are linked to that requirement.

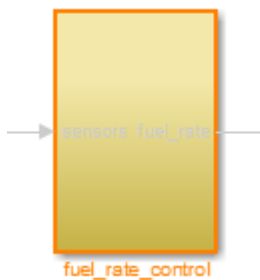
Navigate Between IBM Rational DOORS Requirement and Model Object

In “Insert Navigation Objects into IBM Rational DOORS Requirements” on page 7-8, you created a link between a DOORS requirement and the `fuel_rate_control` subsystem in the `sldemo_fuelsys` model. Navigate the links in both directions:

- 1 With the `sldemo_fuelsys` model closed, go to the DOORS requirement in the formal module.
- 2 Left-click the Simulink reference object that you inserted to select it.
- 3 Select **MATLAB > Select item**.



Your version of the `sldemo_fuelsys` model opens, with the `fuel_rate_control` subsystem highlighted.



- 4 Log in to the DOORS software.
- 5 Navigate from the model to the DOORS requirement. In the Model Editor, right-click the `fuel_rate_control` subsystem and select **Requirements > 1. “<requirement name>”** where **<requirement name>** is the name of the DOORS requirement that you created.

The DOORS formal module opens with the requirement object and its child objects highlighted in red.

1 Fuel rate controller requirements

The controller will use engine speed, throttle position and manifold pressure airflow through the engine.

[Simulink reference: sldemo_fuelsys_test/fuel_rate_control (SubSystem)]



Why Add Navigation Objects to IBM Rational DOORS Requirements?

IBM Rational DOORS software is a requirements management application that you use to capture, track, and manage requirements. The Requirements Management Interface (RMI) allows you to link Simulink objects to requirements managed by external applications, including the DOORS software.

When you create a link from a Simulink object to a DOORS requirement, the RMI stores the link data in Simulink. Using this link, you can navigate from the Simulink object to its associated requirement.

You can also configure the RMI to insert a navigation object in the DOORS database. This navigation object serves as a link from the DOORS requirement to its associated Simulink object.

To insert navigation objects into a DOORS database, you must have write access to the DOORS database.

Customize IBM Rational DOORS Navigation Objects

If the RMI is configured to modify the destination for bidirectional linking as described in “Enable Linking from IBM Rational DOORS Databases to Simulink Objects” on page 7-7, the RMI can insert a navigation object into your requirements document. This object

looks like the icon for the Simulink software: 

Note In IBM Rational DOORS requirements documents, clicking a navigation object does not navigate back to your Simulink object. Select **MATLAB > Select object** to find the Simulink object that contains the requirements link.

To use an icon of your choosing for the navigation object:

- 1 Select **Analysis > Requirements > Settings**.
- 2 Select the **Selection Linking** tab.
- 3 Select **Modify destination for bidirectional linking**.

Selecting this option enables the **Use custom bitmap for navigation controls in documents** option.

- 4 Select **Use custom bitmap for navigation controls in documents**.
- 5 Click **Browse** to locate the file you want to use for the navigation objects.

For best results, use an icon file (.ico) or a small (16×16 or 32×32) bitmap image (.bmp) file for the navigation object. Other types of image files might give unpredictable results.

- 6 Select the desired file to use for navigation objects and click **Open**.
- 7 Close the Requirements Settings dialog box.

The next time you insert a navigation object into a requirements document, the RMI uses the file you selected.

Tip You can specify a custom template for labels of requirements links to DOORS objects. For more information, see the `rmi` command.

Synchronize Simulink Models with IBM Rational DOORS Databases by using Surrogate Modules

Synchronize a Simulink Model to Create a Surrogate Module

The first time that you synchronize your model with the DOORS software, the DOORS software creates a surrogate module.

In this tutorial, you synchronize the `sf_car` model with the DOORS software.

Note Before you begin, make sure you know how to create links from a Simulink model object to a requirement in a DOORS database.

- 1 To create a surrogate module, start the DOORS software and open a project. If the DOORS software is not already running, start the DOORS software and open a project.
- 2 Open the `sf_car` model.
- 3 Rename the model to `sf_car_doors`, and save the model in a writable folder.
- 4 Create links to a DOORS formal module from two objects in `sf_car_doors`:
 - The transmission subsystem
 - The engine torque block inside the Engine subsystem
- 5 Save the changes to the model.
- 6 In the Simulink Editor, select **Analysis > Requirements > Synchronize with DOORS**.

The DOORS synchronization settings dialog box opens.

- 7 For this tutorial, accept the default synchronization options.

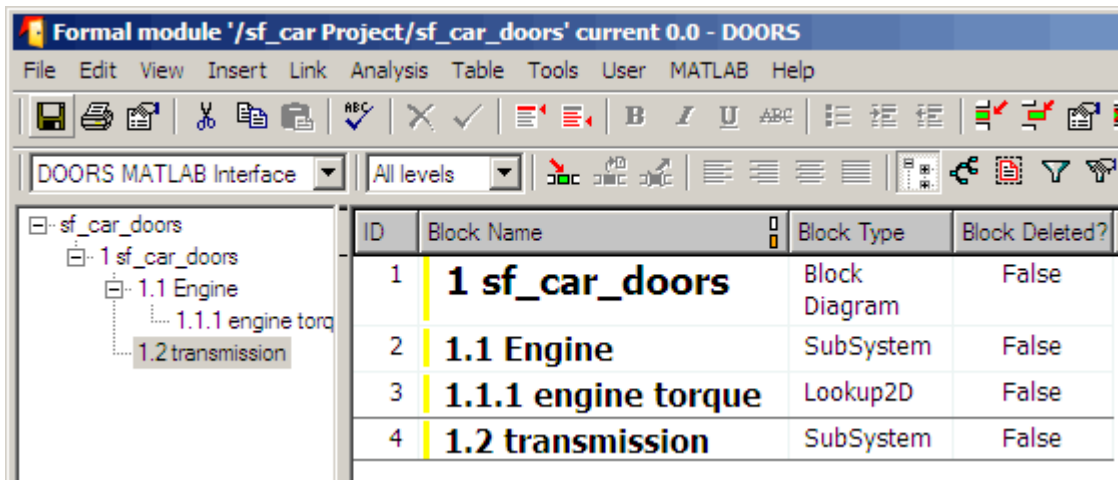
The default option under **Extra mapping additionally to objects with links**, `None`, creates objects in the surrogate module only for the model and any model objects with links to DOORS requirements.

Note For more information about the synchronization options, see “Customize IBM Rational DOORS Synchronization” on page 7-19.

- 8 Click **Synchronize** to create and open a surrogate module for all DOORS requirements that have links to objects in the `sf_car_doors` model.

After synchronization with the None option, the surrogate module, a formal module named `sf_car_doors`, contains:

- A top-level object for the model (`sf_car_doors`)
- Objects that represent model objects with links to DOORS requirements (transmission, engine torque), and their parent objects (Engine).



- 9 Save the surrogate module and the model.

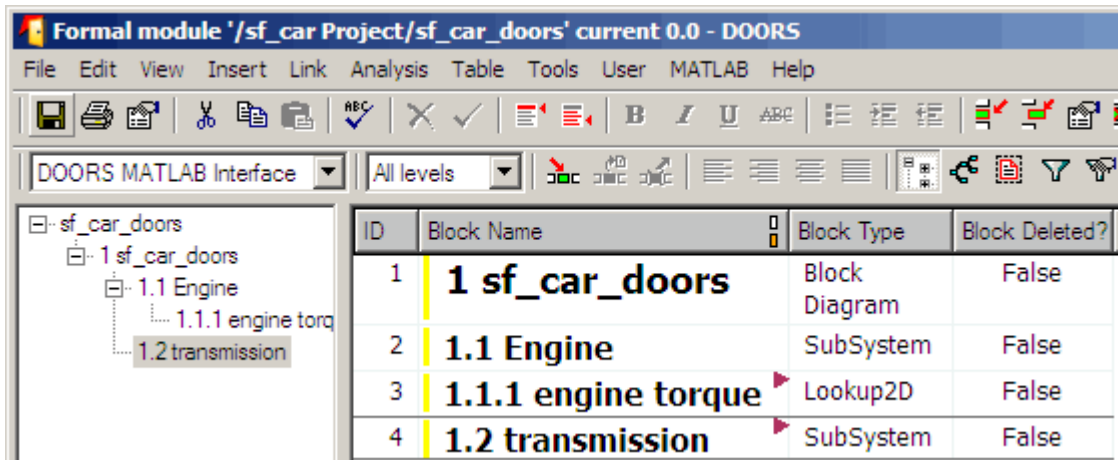
Create Links Between Surrogate Module and Formal Module in an IBM Rational DOORS Database

The surrogate module is the interface between the DOORS formal module that contains your requirements and the Simulink model. To establish links between the surrogate module and the requirements module, copy the link information from the model to the surrogate module:

- 1 Open the `sf_car_doors` model.
- 2 In the Simulink Editor, select **Analysis > Requirements > Synchronize with DOORS**.

- 3 In the DOORS synchronization settings dialog box, select two options:
 - **Update links during synchronization**
 - **from Simulink to DOORS.**
- 4 Click **Synchronize**.

The RMI creates links from the DOORS surrogate module to the formal module. These links correspond to links from the Simulink model to the formal module. In this example, the DOORS software copies the links from the engine torque block and transmission subsystems to the formal module, as indicated by the red triangles.



Resynchronize IBM Rational DOORS Surrogate Module to Reflect Model Changes

If you change your model after synchronization, the RMI does not display a warning message. If you want the surrogate module to reflect changes to the Simulink model, resynchronize your model.

In this tutorial, you add a new block to the `sf_car_doors` model, and later delete it, resynchronizing after each step:

- 1 In the `sf_car_doors` model, make a copy of the vehicle mph (yellow) & throttle % Scope block and paste it into the model. The name of the new Scope block is vehicle mph (yellow) & throttle %1.

- 2 Select **Analysis > Requirements > Synchronize with DOORS**.
- 3 In the DOORS synchronization settings dialog box, leave the **Extra mapping additionally to objects with links** option set to Complete - All blocks, subsystems, states, and transitions. Click **Synchronize**.

After the synchronization, the surrogate module includes the new block.

89	1.10.6 Ti	Outport	False
90	1.10.7 Tout	Outport	False
91	1.11 vehicle mph (yellow) & throttle %0	Scope	False
92	1.12 vehicle mph (yellow) & throttle %01	Scope	True

- 4 In the sf_car_doors model, delete the newly added Scope block and resynchronize.

The block that you delete appears at the bottom of the list of objects in the surrogate module. Its entry in the **Block Deleted** column reads True.

89	1.10.6 Ti	Outport	False
90	1.10.7 Tout	Outport	False
91	1.11 vehicle mph (yellow) & throttle %0	Scope	False
92	1.12 vehicle mph (yellow) & throttle %01	Scope	False

- 5 Delete the copied object (vehicle mph (yellow) & throttle %1) and resynchronize the model.
- 6 Save the surrogate module.
- 7 Save the sf_car_doors model.

Navigate with the Surrogate Module

Navigate Between Requirements and the Surrogate Module in the DOORS Database

The surrogate module and the requirements in the formal module are both in the DOORS database. When you synchronize your model, the DOORS software creates links between the surrogate module objects and the requirements in the DOORS database.

Navigating between the requirements and the surrogate module allows you to review the requirements that have links to the model without starting the Simulink software.

To navigate from the surrogate module transmission object to the requirement in the formal module:

- 1 In the surrogate module object for the transmission subsystem, right-click the right-facing red arrow.

65	1.9.3.7 up_th	Output	False
4	1.10 transmission	/sf_car Project/sf_car Requirements ▶ 1: Transmission Requirements: Shoul	
66	1.10.1 Ne	Inport	False

- 2 Select the requirement name.

The formal module opens, at the Transmission Requirements object.

To navigate from the requirement in the formal module to the surrogate module:

- 1 In the Transmission Requirements object in the formal module, right-click the left-facing orange arrow.

1	1 Transmission Requirements	Should have five gears: Park, Reverse, D1, D2, and Lo. User cannot start engine unless the Park gear is engaged. Reverse cannot be entered from D1 unless the vehicle is not moving. Lo cannot be entered from D2 unless the speed of the vehicle is less than 15 mph.	
		/sf_car Project/sf_car_doors ▶ 4: transmission	

- 2 Select the object name.

The surrogate module for `sf_car_doors` opens, at the object associated with the transmission subsystem.

Navigate Between DOORS Requirements and the Simulink Module via the Surrogate Module

You can create links that allow you to navigate from Simulink objects to DOORS requirements and from DOORS requirements to the model. If you synchronize your model, the surrogate module serves as an intermediary for the navigation in both directions. The surrogate module allows you to navigate in both directions even if you remove the direct link from the model object to the DOORS formal module.

Navigate from a Simulink Object to a Requirement via the Surrogate Module

To navigate from the transmission subsystem in the `sf_car_doors` model to a requirement in the DOORS formal module:

- 1 In the `sf_car_doors` model, right-click the transmission subsystem and select **Requirements > 1. "DOORS Surrogate Item"**. (The direct link to the DOORS formal module is also available.)

The surrogate module opens, at the object associated with the transmission subsystem.

- 2 To display the individual requirement, in the surrogate module, right-click the right-facing red arrow and select the requirement.

The formal module opens, at `Transmission Requirements`.

Navigate from a Requirement to the Model via the Surrogate Module

To navigate from the `Transmission Requirements` requirement in the formal module to the transmission subsystem in the `sf_car_doors` model:

- 1 In the formal module, in the `Transmission Requirements` object, right-click the left-facing orange arrow.
- 2 Select the path to the linked surrogate object: **/sf_car Project/sf_car_doors > 4. transmission**.

The surrogate module opens, at the transmission object.

- 3 In the surrogate module, select **MATLAB > Select item**.

The linked object is highlighted in `sf_car_doors`.

Customize IBM Rational DOORS Synchronization

DOORS Synchronization Settings

When you synchronize your Simulink model with a DOORS database, you can:

- Customize the level of detail for your surrogate module.
- Update links in the surrogate module or in the model to verify the consistency of requirements links among the model, and the surrogate and formal modules.

The DOORS synchronization settings dialog box provides the following options during synchronization.

DOORS Settings Option	Description
DOORS surrogate module path and name	Specifies a unique DOORS path to a new or an existing surrogate module. For information about how the RMI resolves the path to the requirements document, see “Document Path Storage” on page 11-48.
Extra mapping additionally to objects with links	Determines the completeness of the Simulink model representation in the DOORS surrogate module. None specifies synchronizing only those Simulink objects that have linked requirements, and their parent objects. For more information about these synchronization options, see “Customize the Level of Detail in Synchronization” on page 7-21.
Update links during synchronization	Specifies updating any unmatched links the RMI encounters during synchronization, as designated in the Copy unmatched links and Delete unmatched links options.

DOORS Settings Option	Description
Copy unmatched links	<p>During synchronization, selecting the following options has the following results:</p> <ul style="list-style-type: none"> • from Simulink to DOORS: For links between the model and the formal module, the RMI creates matching links between the DOORS surrogate and formal modules. • from DOORS to Simulink: For links between the DOORS surrogate and formal modules, the RMI creates matching links between the model and the DOORS modules.
Delete unmatched links	<p>During synchronization, selecting the following options has the following results:</p> <ul style="list-style-type: none"> • Remove unmatched in DOORS: For links between the formal and surrogate modules, when there is not a corresponding link between the model and the DOORS modules, the RMI deletes the link in DOORS. <p>This option is available only if you select the from Simulink to DOORS option.</p> <ul style="list-style-type: none"> • Remove unmatched in Simulink: For links between the model and the DOORS modules, when there is not a corresponding link between the formal and surrogate modules, the RMI deletes the link from the model. <p>This option is available only if you select the from DOORS to Simulink option.</p>
Save DOORS surrogate module	<p>After the synchronization, saves changes to the surrogate module and updates the version of the surrogate module in the DOORS database.</p>
Save Simulink model (recommended)	<p>After the synchronization, saves changes to the model. If you use a version control system, selecting this option changes the version of the model.</p>

Resynchronize a Model with a Different Surrogate Module

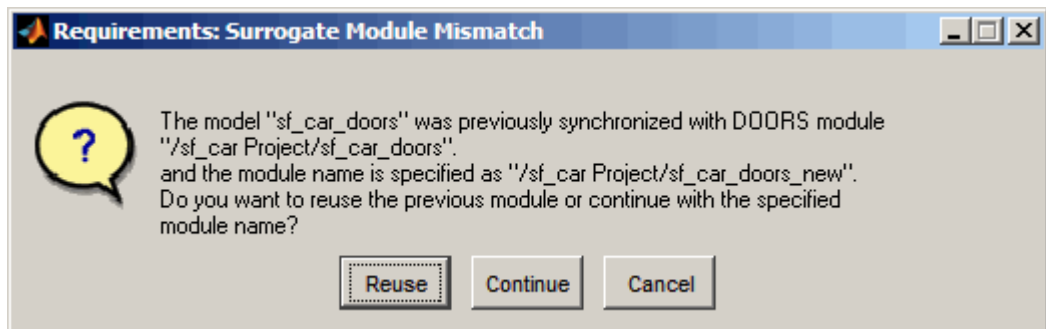
You can synchronize the same Simulink model with a new DOORS surrogate module. For example, you might want the surrogate module to contain only objects that have requirements to DOORS, rather than all objects in the model. In this case, you can change the synchronization options to reduce the level of detail in the surrogate module:

- 1 In the DOORS synchronization settings dialog box, change the **DOORS surrogate module path and name** to the path and name of the new surrogate module in the DOORS database.
- 2 Specify a module with either a relative path (starting with ./) or a full path (starting with /).

The software appends relative paths to the current DOORS project. Absolute paths must specify a project and a module name.

When you synchronize a model, the RMI automatically updates the **DOORS surrogate module path and name** with the actual full path. The RMI saves the unique module ID with the module.

- 3 If you select a new module path or if you have renamed the surrogate module, and you click **Synchronize**, the Requirements: Surrogate Module Mismatch dialog box opens.



- 4 Click **Continue** to create a new surrogate module with the new path or name.

Customize the Level of Detail in Synchronization

You can customize the level of detail in a surrogate module so that the module reflects the full or partial Simulink model hierarchy.

In “Synchronize a Simulink Model to Create a Surrogate Module” on page 7-13, you synchronized the model with the **Extra mapping additionally to objects with links** option set to **None**. As a result, the surrogate module contains only Simulink objects that have requirement links, and their parent objects. Additional synchronization options, described in this section, can increase the level of surrogate detail. Increasing the level of surrogate detail can slow down synchronization.

The **Extra mapping additionally to objects with links** option can have one of the following values. Each subsequent option adds additional Simulink objects to the surrogate module. You choose **None** to minimize the surrogate size or **Complete** to create a full representation of your model. The **Complete** option adds all Simulink objects to the surrogate module, creating a one-to-one mapping of the Simulink model in the surrogate module. The intermediate options provide more levels of detail.

Drop-Down List Option	Description
None (Recommended for better performance)	Maps only Simulink objects that have requirements links and their parent objects to the surrogate module.
Minimal - Non-empty unmasked subsystems and Stateflow charts	Adds all nonempty Stateflow charts and unmasked Simulink subsystems to the surrogate module.
Moderate - Unmasked subsystems, Stateflow charts, and superstates	Adds Stateflow superstates to the surrogate module.
Average - Nontrivial Simulink blocks, Stateflow charts and states	Adds all Stateflow charts and states and Simulink blocks, except for trivial blocks such as ports, bus objects, and data-type converters, to the surrogate module.
Extensive - All unmasked blocks, subsystems, states and transitions	Adds all unmasked blocks, subsystems, states, and transitions to the surrogate module.
Complete - All blocks, subsystems, states and transitions	Copies <i>all</i> blocks, subsystems, states, and transitions to the surrogate module.

Resynchronize to Include All Simulink Objects

This tutorial shows how you can include *all* Simulink objects in the DOORS surrogate module. Before you start these steps, make sure you have completed the tutorials “Synchronize a Simulink Model to Create a Surrogate Module” on page 7-13 and “Create

Links Between Surrogate Module and Formal Module in an IBM Rational DOORS Database” on page 7-14.

- 1 Open the `sf_car_doors` model that you synchronized in “Synchronize a Simulink Model to Create a Surrogate Module” on page 7-13 and again in “Create Links Between Surrogate Module and Formal Module in an IBM Rational DOORS Database” on page 7-14.
- 2 In the Simulink Editor, select **Analysis > Requirements > Synchronize with DOORS**.


The DOORS synchronization settings dialog box opens.

- 3 Resynchronize with the same surrogate module, making sure that the **DOORS surrogate module path and name** specifies the surrogate module path and name that you used in “Synchronize a Simulink Model to Create a Surrogate Module” on page 7-13.

For information about how the RMI resolves the path to the requirements document, see “Document Path Storage” on page 11-48.

- 4 Update the surrogate module to include *all* objects in your model. To do this, under **Extra mapping additionally to objects with links**, from the drop-down list, select **Complete - All blocks, subsystems, states and transitions**.
- 5 Click **Synchronize**.

After synchronization, the DOORS surrogate module for the `sf_car_doors` model opens with the updates. All Simulink objects and all Stateflow objects in the `sf_car_doors` model are now mapped in the surrogate module.

ID	Block Name	Block Type	Block Deleted?
1	1 sf_car_doors	Block Diagram	False
2	1.1 Engine	SubSystem	False
5	1.1.1 Ti	Inport	False
6	1.1.2 throttle	Inport	False
7	1.1.3 Integrator	Integrator	False
8	1.1.4 Sum	Sum	False
3	1.1.5 engine torque 	Lookup2D	False
9	1.1.6 engine + impeller inertia	Gain	False
10	1.1.7 Ne	Outport	False
11	1.2 Mux	Mux	False
12	1.3 User Inputs:Passing Maneuver	Signal Group	False
13	1.4 User Inputs:Gradual Acceleration	Signal Group	False
14	1.5 User Inputs:Hard	Signal Group	False

6 Scroll through the surrogate module. Notice that the objects with requirements (the engine torque block and transmission subsystem) retain their links to the DOORS formal module, as indicated by the red triangles.

7 Save the surrogate module.

Detailed Information About The Surrogate Module You Created

Notice the following information about the surrogate module that you created in “Resynchronize to Include All Simulink Objects” on page 7-22:

- The name of the surrogate module is `sf_car_doors`, as you specified in the DOORS synchronization settings dialog box.
- DOORS object headers are the names of the corresponding Simulink objects.
- The **Block Type** column identifies each object as a particular block type or a subsystem.
- If you delete a previously synchronized object from your Simulink model and then resynchronize, the **Block Deleted** column reads **true**. Otherwise, it reads **false**.

These objects are not deleted from the surrogate module. The DOORS software retains these surrogate module objects so that the RMI can recover these links if you later restore the model object.

- Each Simulink object has a unique ID in the surrogate module. For example, the ID for the surrogate module object associated with the Mux block in the preceding figure is 11.
- Before the complete synchronization, the surrogate module contained the transmission subsystem, with an ID of 3. After the complete synchronization, the transmission object retains its ID (3), but is listed farther down in the surrogate module. This order reflects the model hierarchy. The transmission object in the surrogate module retains the red arrow that indicates that it links to a DOORS formal module object.

Synchronization with IBM Rational DOORS Surrogate Modules

Synchronization is a user-initiated process that creates or updates a DOORS surrogate module. A *surrogate module* is a DOORS formal module that is a representation of a Simulink model hierarchy.

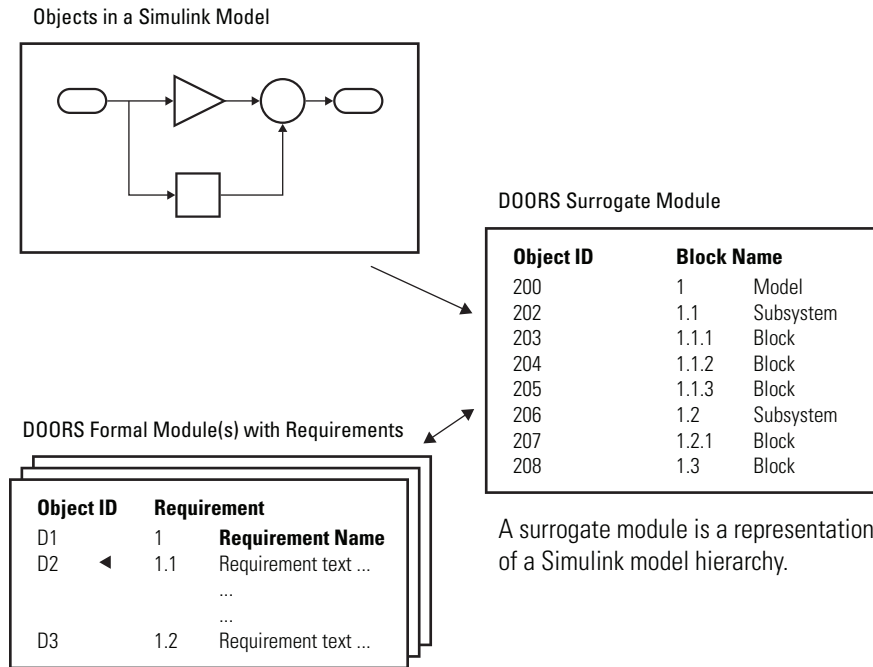
When you synchronize a model for the first time, the DOORS software creates a surrogate module. The surrogate module contains a representation of the model, depending on your synchronization settings. (To learn how to customize the links and level of detail in the synchronization, see “Customize IBM Rational DOORS Synchronization” on page 7-19.)

If you create or remove model objects or links, keep your surrogate module up to date by resynchronizing. The updated surrogate module reflects any changes in the requirements links since the previous synchronization.

Note The RMI and DOORS software both use the term *object*. In the RMI, and in this document, the term object refers to a Simulink model or block, or to a Stateflow chart or its contents.

In the DOORS software, object refers to numbered elements in modules. The DOORS software assigns each of these objects a unique object ID. In this document, these objects are referred to as DOORS objects.

You use standard DOORS capabilities to navigate between the Simulink objects in the surrogate module and requirements in other formal modules. The surrogate module facilitates navigation between the Simulink model object and the requirements, as the following diagram illustrates.



A surrogate module is a representation of a Simulink model hierarchy.

Enter requirements in the DOORS formal module and link them to objects in the DOORS surrogate module, so you can navigate from requirements to Simulink objects.

Advantages of Synchronizing Your Model with a Surrogate Module

Synchronizing your Simulink model with a surrogate module offers the following advantages:

- You can navigate from a requirement to a Simulink object without modifying the requirements modules.
- You avoid cluttering your requirements modules with inserted navigation objects.
- The DOORS database contains complete information about requirements links. You can review requirements links and verify traceability, even if the Simulink software is not running.
- You can use DOORS reporting features to analyze requirements coverage.
- You can separate the requirements tracking work from the Simulink model developers' work, as follows:
 - Systems engineers can establish requirements links to models without using the Simulink software.
 - Model developers can capture the requirements information using synchronization and store it with the model.
- You can resynchronize a model with a new surrogate module, updating any model changes or specifying different synchronization options.

Simulink Traceability Between Model Objects

- “Link Model Objects” on page 8-2
- “Link Test Cases to Requirements Documents” on page 8-4
- “Link Requirements to Simulink Data Dictionary Entries” on page 8-8
- “Link Signal Builder Blocks to Requirements and Simulink Model Objects” on page 8-10
- “Requirements Links for Library Blocks and Reference Blocks” on page 8-14
- “Navigate to Requirements from Model” on page 8-20

Link Model Objects

Link Objects in the Same Model

You can create a requirements link from one model object to another model object:

- 1 Right-click the link destination model object and select **Requirements > Select for Linking with Simulink**.
- 2 Right-click the link source model object and select **Requirements > Add Link to Selected Object**.
- 3 Right-click the link source model object again and select **Requirements**. The new link appears at the top of the **Requirements** submenu.

Link Objects in Different Models

You can create links between objects in related models. This example shows how to link model objects in `slvndemo_powerwindow_controller` and `slvndemo_powerwindow`.

- 1 Open the `slvndemo_powerwindow_controller` and `slvndemo_powerwindow` models.
- 2 In the `slvndemo_powerwindow` model window, double-click the `power_window_control_system` subsystem. The `power_window_control_system` subsystem opens.
- 3 In the `slvndemo_powerwindow/power_window_control_system` subsystem window, right-click the `control` subsystem. Select **Requirements > Select for Linking with Simulink**.
- 4 In the `slvndemo_powerwindow_controller` model window, right-click the `control` subsystem. Select **Requirements > Add Link to Selected Object**.
- 5 Right-click the `slvndemo_powerwindow_controller/control` subsystem and select **Requirements**. The new RMI link appears at the top of the **Requirements** submenu.
- 6 To verify that the links were created, in the `slvndemo_powerwindow_controller` model window, select **Analysis > Requirements > Highlight Model**.

The blocks with requirements links are highlighted.

- 7 Close the `slvnvdemo_powerwindow_controller` and `slvnvdemo_powerwindow` models.

Link Test Cases to Requirements Documents

Since requirements specify behavior in response to particular conditions, you can build test cases (test inputs, expected outputs, and assessments) from the model requirements. Test cases reproduce specific conditions using test inputs, and assess the actual model output against the expected outputs. As you develop the model, build test files that check system behavior and link them to corresponding requirements. By defining these test cases in test files, you can periodically check your model and archive results to demonstrate model stability.

Establish Requirements Traceability for Testing

If you have a Simulink Test and a Simulink Requirements license, you can link requirements to test harnesses, test sequences, and test cases. Before adding links, review “Supported Requirements Document Types” on page 5-10.


Requirements Traceability for Test Harnesses

When you edit requirements links to the component under test, the links immediately synchronize between the test harness and the main model. Other changes to the component under test, such as adding a block, synchronize when you close the test harness. If you add a block to the component under test, close and reopen the harness to update the main model before adding a requirement link.

To view items with requirements links, select **Analysis > Requirements > Highlight Model**.

Requirements Traceability for Test Sequences

In test sequences, you can link to test steps. To create a link, first find the model item, test case, or location in the document you want to link to. Right-click the test step, select **Requirements**, and add a link or open the link editor.

To highlight or unhighlight test steps that have requirements links, toggle the requirements links highlighting button  in the Test Sequence Editor toolbar. Highlighting test steps also highlights the model block diagram.

Requirements Traceability for Test Cases

If you use many test cases with a single test harness, link to each specific test case to distinguish which blocks and test steps apply to it. To link test steps or test harness blocks to test cases,

- 1 Open the test case in the Test Manager.
- 2 Highlight the test case in the test browser.
- 3 Right-click the block or test step, and select **Requirements > Link to Current Test Case**.

Requirements Traceability Example

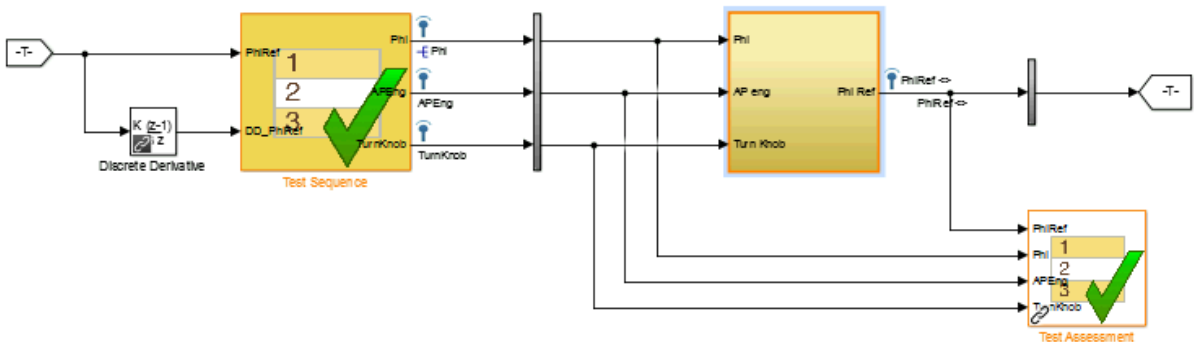
This example demonstrates adding requirements links to a test harness and test sequence. The model is a component of an autopilot roll control system. This example requires Simulink Test and Simulink Requirements.

- 1 Open the test file, the model, and the harness.

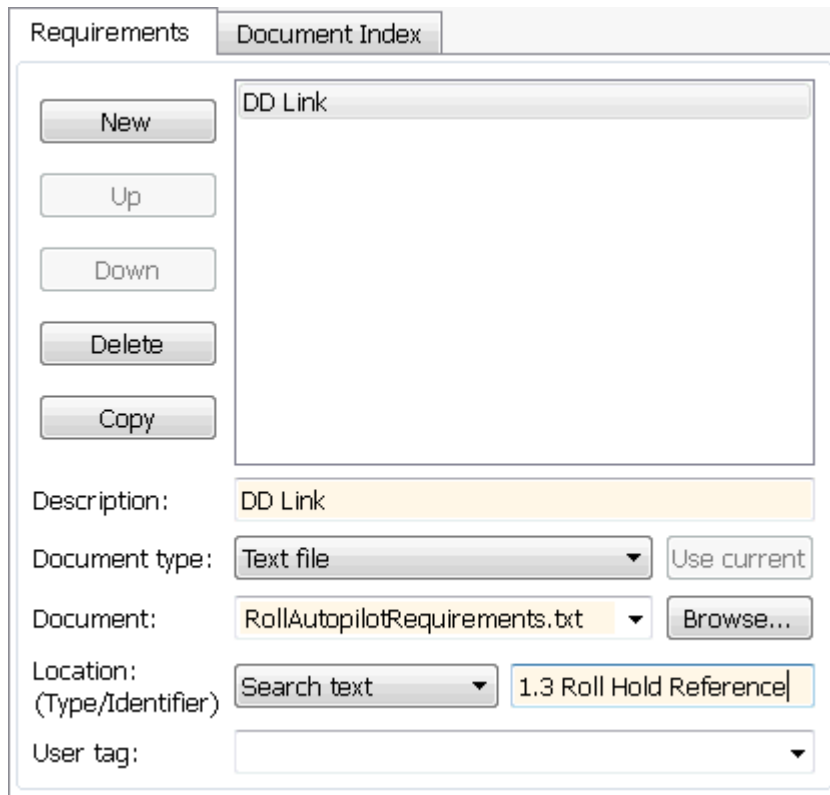
```
open AutopilotTestFile.mldatx,
open_system RollAutopilotMdlRef,
sltest.harness.open('RollAutopilotMdlRef/Roll Reference',...
'RollReference_Requirement1_3')
```

- 2 In the test harness, select **Analysis > Requirements > Highlight Model**.

The test harness highlights the Test Sequence block, component under test, and Test Assessment block.



- 3 Add traceability to the Discrete Derivative block.
 - a Right-click the Discrete Derivative block and select **Requirements > Open Link Editor**.
 - b In the **Requirements** tab, click **New**.
 - c Enter the following to establish the link:
 - Description: DD link
 - Document type: Text file
 - Document: RollAutopilotRequirements.txt
 - Location: 1.3 Roll Hold Reference



- d Click **OK**. The Discrete Derivative block highlights.

- 4 To trace to the requirements document, right-click the Discrete Derivative block, and select **Requirements > DD Link**. The requirements document opens in the editor and highlights the linked text.

1.3 Roll Hold Reference

Navigate to test harness using MATLAB command:

```
web('http://localhost:31415/matlab/feval/rmiobjnavigate?argu
```

REQUIREMENT

1.3.1 When roll hold mode becomes the active mode the roll hold

Navigate to test step using MATLAB command:

```
web('http://localhost:31415/matlab/feval/rmiobjnavigate?argu
```

1.3.1.1. The roll hold reference shall be set to zero if the act

Navigate to test step using MATLAB command:

```
web('http://localhost:31415/matlab/feval/rmiobjnavigate?argu
```

- 5 Open the Test Sequence block. Add a requirements link that links the InitializeTest step to the test case.
 - a In the Test Manager, highlight Requirement 1.3 Test in the test browser.
 - b Right-click the InitializeTest step in the Test Sequence Editor. Select **Requirements > Link to Current Test Case**.

When the requirements link is added, the Test Sequence Editor highlights the step.

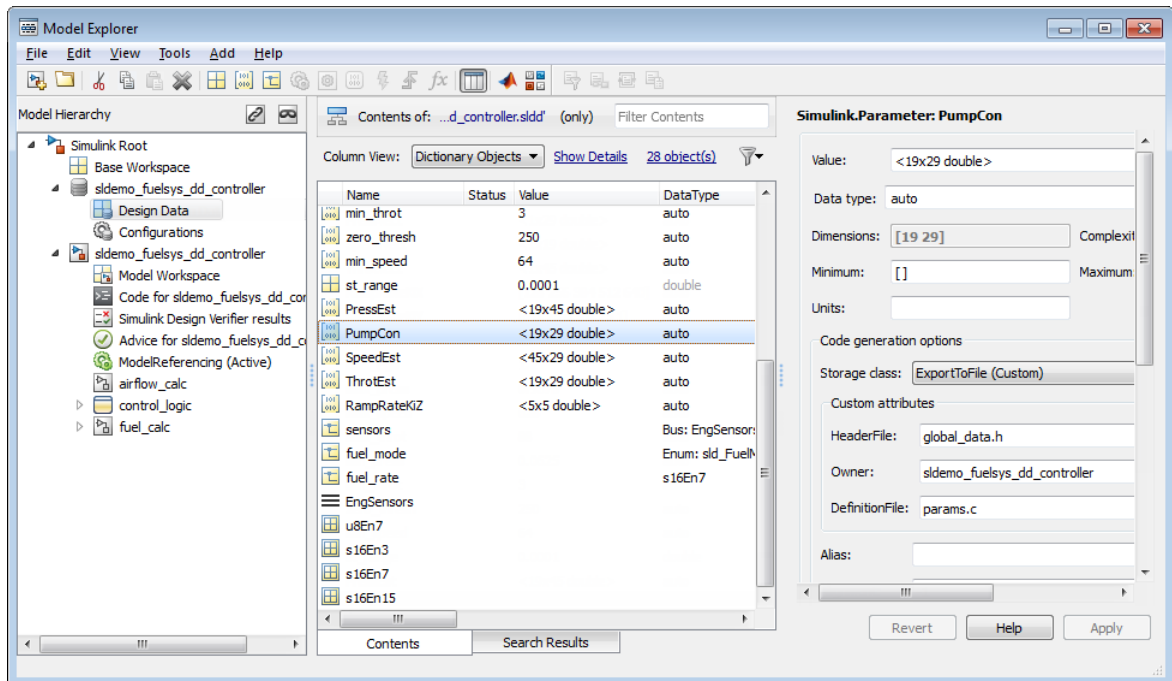
Step	Transition
<pre>InitializeTest Phi = 0; APEng = false; TurnKnob = 0; % Initializes test sequence outputs</pre>	<pre>1. true</pre>

Link Requirements to Simulink Data Dictionary Entries

You can create requirements traceability links for entries in Simulink data dictionaries. The process is similar to linking for other model objects. In the Model Explorer, right-click a data dictionary entry, select **Requirements**, and choose one of the selection-based linking options. You can also use the Link Editor.

This example demonstrates linking to a data dictionary entry.

- 1 Enter `sldemo_fuelsys_dd_controller` at the command line to open `sldemo_fuelsys_dd_controller`.
- 2 Open the linked data dictionary by clicking the data dictionary badge in the bottom left corner of the model.
- 3 In the **Model Hierarchy** pane of the Model Explorer, select **Design Data** in the `sldemo_fuelsys_dd_controller` data dictionary.
- 4 You will link the `PumpCon` parameter to the `Pumping Constant` lookup table in the model.



- 5 Open the `airflow_calc` subsystem and select the Pumping Constant lookup table.
- 6 In the Model Explorer, right-click the PumpCon parameter and select **Requirements > Link to Selection in Simulink**.

The two objects are linked.

- 7 Check the link. Right-click the PumpCon parameter and select **Requirements**, then select the navigation shortcut at the top of the **Requirements** submenu. Simulink highlights the lookup table.

Link Signal Builder Blocks to Requirements and Simulink Model Objects

Link Signal Builder Blocks to Requirements Documents


You can create links from a signal group in a Signal Builder block to a requirements document:

- 1 Open the model:

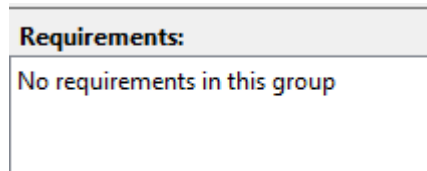
`sf_car`

- 2 In the `sf_car` model window, double-click the User Inputs block.

The Signal Builder dialog box opens, displaying four groups of signals. The Passing Maneuver signal group is the current active group. The RMI associates any requirements links that you add to the current active signal group.

- 3 At the far-right end of the toolbar, click the **Show verification settings** button . (You might need to expand the Signal Builder dialog box for this button to become visible.)

A **Requirements** pane opens on the right-hand side of the Signal Builder dialog box.



- 4 Place your cursor in the window, right-click, and select **Open Link Editor**.

The Requirements Traceability Link Editor opens.

- 5 Click **New**. In the **Description** field, enter `User input requirements`.
- 6 When you browse and select a requirements document, the RMI stores the document path as specified by the **Document file reference** option on the Requirements Settings dialog box, **Selection Linking** tab.

For information about which setting to use for your working environment, see “Document Path Storage” on page 11-48.

- 7 Browse to a requirements document and click **Open**.
- 8 In the **Location** drop-down list, select **Search text** to link to specified text in the document.
- 9 Next to the **Location** drop-down list, enter User Input Requirements.
- 10 Click **Apply** to create the link.
- 11 To verify that the RMI created the link, in the Simulink Editor, select the User Inputs block, right-click, and select **Requirements**.

The link to the new requirement is the option at the top of the submenu.

- 12 Save the `sf_car_linking` model.

Note Links that you create in this way associate requirements information with individual signal groups, not with the entire Signal Builder block.


In this example, switch to a different active group in the drop-down list to link a requirement to another signal group.

Link Signal Builder Blocks to Model Objects

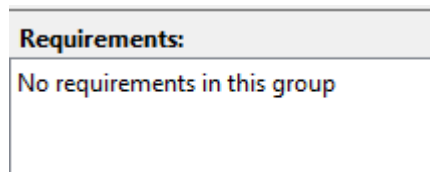
This example shows how to create links from a signal group in a Signal Builder block to a model object:

- 1 Open the `sf_car` model.
- 2 Open the `sf_car/shift_logic` chart.
- 3 Right-click upshifting and select **Requirements > Select for Linking with Simulink**.
- 4 In the `sf_car` model window, double-click the User Inputs block.

The Signal Builder dialog box opens, displaying four groups of signals. The Passing Maneuver signal group is the current active group. The RMI associates any requirements links that you add to the current active signal group.

- 5 In the Signal Builder dialog box, click the **Gradual Acceleration** tab.
- 6 At the far-right end of the toolbar, click the **Show verification settings** button . (You might need to expand the Signal Builder dialog box for this button to become visible.)

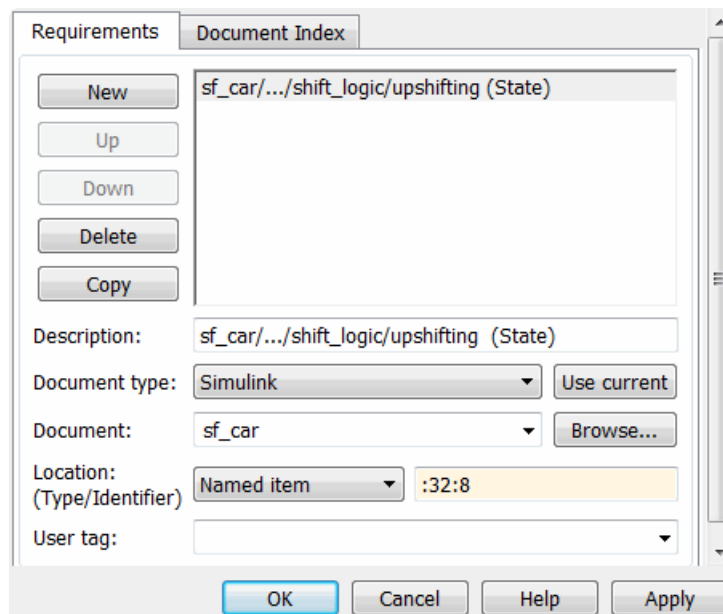
A **Requirements** pane opens on the right-hand side of the Signal Builder dialog box.



- 7 Place your cursor in the window, right-click, and select **Open Link Editor**.

The Requirements Traceability Link Editor opens.

- 8 Click **New**. In the **Description** field, enter Upshifting.
- 9 In the **Document type** field, select Simulink. Click **Use current**. The software fills in the field with the **Location: (Type/Identifier)** information for upshifting.



- 10 Click **Apply** to create the link.
- 11 In the model window, select the User Inputs block, right-click, and select **Requirements**.

The link to the new requirement is the option at the top of the submenu.

- 12** To verify that the links were created, in the `sf_car` model window, select **Analysis > Requirements > Highlight Model**.

The blocks with requirements links are highlighted.

Note Links that you create in this way associate requirements information with individual signal groups, not with the entire Signal Builder block.

- 13** Close the `sf_car` model.

Requirements Links for Library Blocks and Reference Blocks

Introduction to Library Blocks and Reference Blocks

Simulink allows you to create your own block libraries. If you create a block library, you can reuse the functionality of a block, subsystem, or Stateflow atomic subchart in multiple models.

When you copy a library block to a Simulink model, the new block is called a reference block. You can create several instances of this library block in one or more models.

The reference block is linked to the library block using a library link. If you change a library block, any reference block that is linked to the library block is updated with those changes when you open or update the model that contains the reference block.

Note For more information about reference blocks and library links, see “Libraries” (Simulink).

Library Blocks and Requirements

Library blocks themselves can have links to requirements. In addition, if a library block is a subsystem or atomic subchart, the objects inside the library blocks can have library links. You use the Requirements Management Interface (RMI) to create and manage requirements links in libraries and in models.

The following sections describe how to manage requirements links on and inside library blocks and reference blocks.

Copy Library Blocks with Requirements

When you copy a library subsystem or masked block to a model, you can highlight, view and navigate requirements links on the library block and on objects inside the library block. However, those links are not associated with that model. The links are stored with the library, not with the model.

You cannot add, modify, or delete requirements links on the library block from the context of the reference block. If you disable the link from the reference block to the library

block, you can modify requirements on objects that are inside library blocks just as you can for other block attributes when a library link has been disabled.

Manage Requirements on Reference Blocks

You use the RMI to manage requirements links on a reference block just like any other model object. You can view and navigate both local and library requirements on a reference block.

- Locally created requirements links — Can be modified or deleted without changing the library block:
 - **Manifold absolute pressure sensor**
 - **Mass airflow estimation**
- Requirements links on the library block — Cannot be modified or deleted from the context of the reference block:
 - **Speed sensor**
 - **Throttle sensor**
 - **Oxygen sensor**

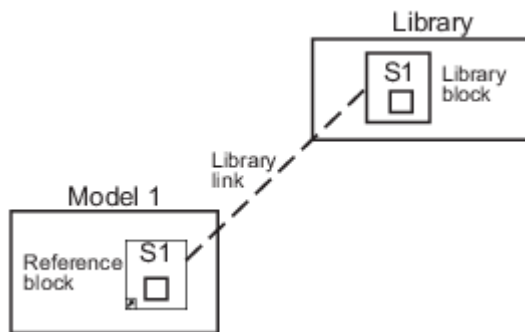
Manage Requirements Inside Reference Blocks

If your library block is a subsystem or a Stateflow atomic subchart, you can create requirements links on objects *inside* the subsystem or subchart. If you disable the link from the reference block to the library, you can add, modify, or delete requirements links on objects inside a reference block. Once you have disabled the link, the RMI treats those links as locally created links.

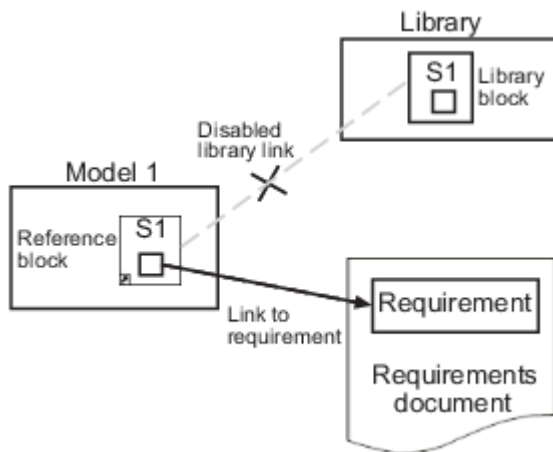
After you make changes to requirements links on objects inside a reference block, you can resolve the link so that those changes are pushed to the library block. The next time you create an instance of that library block, the changes you made are copied to the new instance of the library block.

The workflow for creating a requirement link on an object inside a reference block is:

- 1 Within a library you have a subsystem S1. Drag S1 to a model, creating a new subsystem. This subsystem is the reference block.



- 2 Disable the library link between the reference block and the library block. Keep the library loaded while you disable the link to maintain RMI data. To disable the link, select the reference block and select **Diagram > Library Link > Disable Link**.
- 3 Create a link from the object inside the reference block to the requirements document.

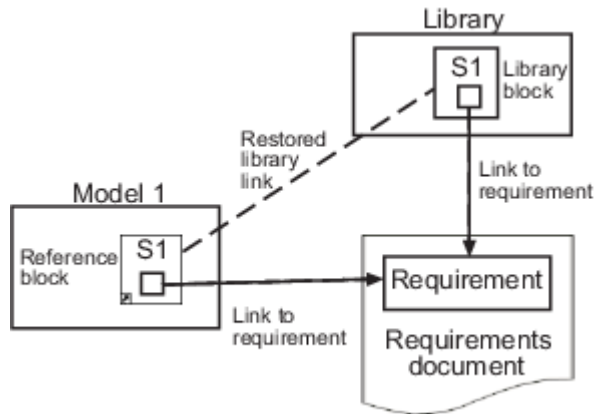


Note When linking to a requirement from inside a reference block, you can create links only in one direction: from the model to the requirements document. The RMI does not support inserting navigation objects into requirements documents for objects inside reference blocks.

- 4 Resolve the library link between the reference block and the library block:

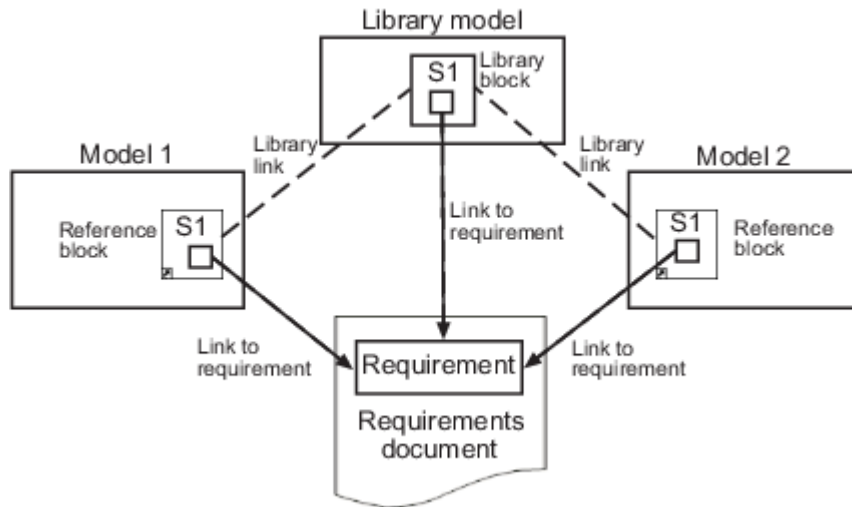
- a Select the reference block.
- b Select **Diagram > Library Link > Resolve Link**.
- c In the **Action** column, click **Push**.
- d Click **OK** to resolve the link to the library block and push the newly added requirement to the object inside the library block.

When you resolve the library link between the library block and the subsystem, Simulink pushes the new requirement link to the library block S1. The following graphic shows the new link from inside the library block S1 to the requirement.



Note If you see a message that the library is locked, you must unlock the library before you can push the changes to the library block.

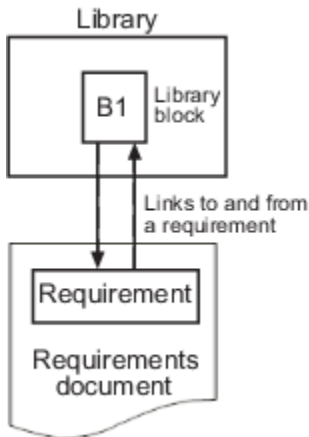
- 5 If you reuse library block S1, which now has an object with a requirement link, in another model, the new subsystem contains an object that links to that requirement.



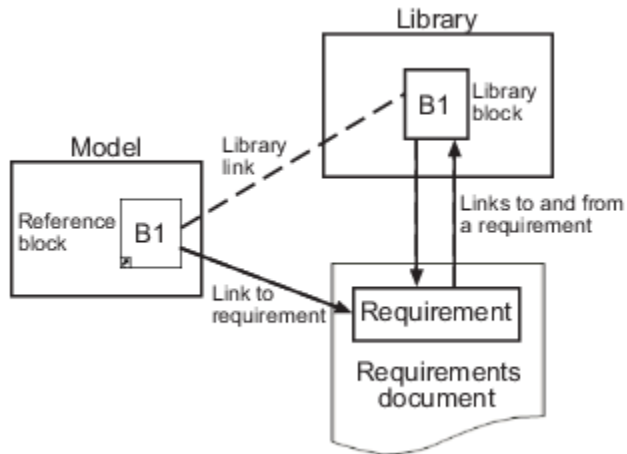
Links from Requirements to Library Blocks

If you have a requirement that links to a library block and you drag that library block to a model, the requirement does not link to the reference block; the requirement links *only* to the library block.

For example, consider the situation where you have established linking between a library block (B1 in the following graphic) and a requirement in both directions.



When you use library block B1 in a model, you can navigate from the reference block to the requirement. However, the link from the requirement still points only to library block B1, not to the reference block.



As discussed in the previous section, you can create requirements links on objects inside instances of library block after disabling library links. However, the RMI prohibits you from creating a link from the requirements document to such an object because that link would become invalid when you restored the library link.

Navigate to Requirements from Model

Navigate from Model Object

You can navigate directly from a model object to that object's associated requirement. When you take these steps, the external requirements document opens in the application, with the requirements text highlighted.

- 1 Open the example model:
`slvndemo_fuelsys_officereq`
- 2 Open the fuel rate controller subsystem.
- 3 To open the linked requirement, right-click the Airflow calculation subsystem and select **Requirements > 1. "Mass airflow estimation"**.

The Microsoft Word document `slvndemo_FuelSys_DesignDescription.docx`, opens with the section **2.1 Mass airflow estimation** selected.

Note If you are running a 64-bit version of MATLAB, when you navigate to a requirement in a PDF file, the file opens at the top of the page, not at the bookmark location.

Navigate from System Requirements Block

Sometimes you want to see all the requirements links at a given level of the model hierarchy. In such cases, you can insert a System Requirements block to collect all requirements links in a model or subsystem. The System Requirements block lists requirements links for the model or subsystem in which it resides; it does not list requirements links for model objects inside that model or subsystem, because those are at a different level of the model hierarchy.

In the following example, you insert a System Requirements block at the top level of the `slvndemo_fuelsys_officereq` model, and navigate to the requirements using the links inside the block.

- 1 Open the example model:
`slvndemo_fuelsys_officereq`
- 2 In the Simulink Editor, select **Analysis > Requirements > Highlight Model**.

- 3 Open the fuel rate controller subsystem.

The Airflow calculation subsystem has a requirements link.

- 4 Open the Airflow calculation subsystem.
- 5 In the Simulink Editor, select **View > Library Browser**.
- 6 On the **Libraries** pane, select **Simulink Requirements**.

This library contains only one block—the System Requirements block.

- 7 Drag a System Requirements block into the Airflow calculation subsystem.

The RMI software collects and displays any requirements links for that subsystem in the System Requirements block.

- 8 In the System Requirements block, double-click **1. “Mass airflow subsystem”**.

The Microsoft Word document, `slvndemo_FuelSys_DesignDescription.docx`, opens, with the section **2.1 Mass airflow estimation** selected.

MATLAB Code Traceability

Requirements Traceability for MATLAB Code Lines

Link Between MATLAB Code Lines and Requirements Information in External Documents

Create Link Using Context Menu Shortcuts

To create requirements traceability links from MATLAB code lines to selections in Microsoft Word, Microsoft Excel, or IBM Rational DOORS documents, you can use shortcuts in the Requirements Traceability context menu.

- 1 In your requirements document, select the target for the traceability link that you want to create.
- 2 In the MATLAB Editor, select the line or lines of code that you want to link.
- 3 In the MATLAB Editor, right-click your selection.
- 4 From the context menu, select **Requirements**. Depending on the type of your requirements document, select one of the following options:
 - **Link to Selection in Word**
 - **Link to Selection in Excel**
 - **Link to Selection in DOORS**

The software creates a traceability link from the selected MATLAB code range to the selection in the requirements document. It also inserts a navigation object for the selection in the requirements document. The navigation object links to the selected MATLAB code range.

Create Link Using Link Editor

You can create, edit, and delete traceability links with the Link Editor. To open the Link Editor:

- In the MATLAB Editor, select the line or lines of code that you want to link.
- Right-click your selection.
- From the context menu, select **Requirements > Open Link Editor**.

For detailed information on the Link Editor, see “Outgoing Links Editor” on page 10-7.

Enable or Disable Highlighting of Traceability Links for MATLAB Code

Enable Traceability Highlighting of MATLAB Code

To highlight traceability links in your MATLAB code, do one of the following:

- In the **View** tab, in the **Display** section, select **Highlight Traceability**.
- In the MATLAB Editor, right-click in a line of code with a traceability link. From the context menu, select **Requirements > Enable Traceability Highlighting**.

Disable Traceability Highlighting of MATLAB Code

To turn off highlighting of traceability links in your MATLAB code, do one of the following:

- In the **View** tab, in the **Display** section, clear **Highlight Traceability**.
- In the MATLAB Editor, right-click in a line of code with a traceability link. From the context menu, select **Requirements > Disable Traceability Highlighting**.

Remove Traceability Links from MATLAB Code Lines

Delete Links to Requirements from MATLAB Code Lines

To remove requirements traceability links from a line or lines of MATLAB code:

- 1 In the MATLAB Editor, right-click within a range of code that has requirements traceability links.
- 2 From the context menu, select **Requirements > Delete All Links**.

All links to requirements from this MATLAB code range are deleted. Links to this MATLAB code range from external requirements documents are not deleted.

Delete Link Targets in MATLAB Code Lines

If you have links to MATLAB code ranges from external requirements documents, you can delete the targets for these links from your MATLAB code.

To remove requirements traceability targets from a line or lines of MATLAB code, after you have deleted its outgoing links as described in the previous section:

- 1 In the MATLAB Editor, right-click within a previously linked range of code.
- 2 From the context menu, select **Requirements > Discard Named Range**.

When you discard a named range, links to that MATLAB code range from external documents no longer work. Note, however, that this does not delete navigation objects in external requirements documents.

Traceability for MATLAB Code Lines

Traceability Link Targets

You can create MATLAB code traceability links for:

- Lines of MATLAB code in a standalone file.
- Lines of MATLAB code inside a MATLAB Function block.

You can create links from a line or lines of MATLAB code to:

- Objects in Simulink models.
- Targets in Microsoft Word or Microsoft Excel documents.
- Targets in IBM Rational DOORS databases.
- Targets in text, HTML, or PDF documents.
- HTTP URLs.

Bidirectional linking is supported for targets in MATLAB, Simulink, Microsoft Word, Microsoft Excel, and IBM Rational DOORS. Bidirectional linking creates links to and from the selected link destination. To enable bidirectional linking, in the Requirements Settings dialog box, under the Selection Linking tab, select **Modify destination for bidirectional linking**. For more information, see “Selection Linking Tab” on page 5-12.

You can also create links to MATLAB code lines from any external application that supports HTTP navigation.

Traceability Links in Code Generation Reports

Embedded Coder® embeds requirements traceability links for MATLAB files saved externally from the Simulink model and referenced from MATLAB Function blocks in Simulink. Click the hyperlink in the code generation report to navigate to the corresponding requirement in the Requirements Editor.

Storage of Traceability Links

In a standalone MATLAB file, you can create, navigate, and delete traceability links for lines of code without changing the MATLAB file. The Requirements Management Interface stores requirements traceability data for a MATLAB file in a `.req` file with the same name and location as the MATLAB file.

If you want to create traceability links for lines of code in a MATLAB Function block, set the parent model to store requirements data externally. For a new model, see “Requirements Link Storage” on page 5-5. For an existing model, see “Move Internally Stored Requirements Links to External Storage” on page 5-7. When you create traceability links for code inside a MATLAB Function block, the RMI stores them in a `.req` file for the parent model. The `.req` file for the model contains requirements traceability data for linked model objects and for linked code in MATLAB Function blocks in the model.

Limitations of MATLAB Code Traceability

Overlapping Linked Ranges

The software does not support traceability links for overlapping regions of MATLAB code. If one linked range of code completely overlaps another smaller region of code, the link for the larger range overshadows the link for the smaller range. To avoid complications from overlapping linked ranges, when you create traceability links for MATLAB code lines, choose ranges of code that do not overlap.

Cut and Paste Operations

You can cut or copy a selection of code that has traceability links. When you paste that selection, the software attempts to recreate the corresponding traceability links at the paste location. Depending on paste location and code formatting, you might need to recreate the traceability links manually.

Drag Operation

If you select code that has traceability links and drag that code to a new location, you might need to recreate traceability links for the code in the new location.

MATLAB Function Block Code Traceability in Web View

Requirements linked to individual MATLAB code lines inside a MATLAB Function block appear in HTML requirements traceability reports, but do not appear the Simulink Report Generator™ Web view. See “Create and Use a Web View” (Simulink Report Generator).

Traceability for MATLAB Live Editor

Requirements traceability is not supported for MATLAB Live Editor.

URL and Custom Traceability

- “Requirement Links and Link Types” on page 10-2
- “Custom Link Types” on page 10-10

Requirement Links and Link Types

Requirements Traceability Links

When you want to navigate from a Simulink model or from a region of MATLAB code to a location inside a requirements document, you can add requirements traceability links to the model or code.

Requirements traceability links have the following attributes:

- A description of up to 255 characters.
- A requirements document path name, such as a Microsoft Word file or a module in an IBM Rational DOORS database. (The RMI supports several built-in document formats. You can also register custom types of requirements documents. See “Supported Requirements Document Types” on page 5-10.)
- A designated location inside the requirements document, such as:
 - Bookmark
 - Anchor
 - ID
 - Page number
 - Line number
 - Cell range
 - Link target
 - Tags that you define

Supported Model Objects for Requirements Linking

You can associate requirements links between the following types of Simulink model objects:

- Simulink block diagrams and subsystems
- Simulink blocks and annotations
- Simulink data dictionary entries
- Signal Builder signal groups

- Stateflow charts, subcharts, states, transitions, and boxes
- Stateflow functions
- Lines of MATLAB code
- Simulink Test Manager test cases

Links and Link Types

Requirements links are the data structures, managed by Simulink, that identify a specific location within a document. You get and set the links on a block using the `rmi` command.

Links and link types work together to perform navigation and manage requirements. The `doc` and `id` fields of a link uniquely identify the linked item in the external document. The RMI passes both of these values to the navigation command when you navigate a link from the model.

Link Type Properties

Link type properties define how links are created, identified, navigated to, and stored within the requirement management tool. The following table describes each of these properties.

Property	Description
Registration	The name of the function that creates the link type. The RMI stores this name in the Simulink model.
Label	A string to identify this link type. In the “Outgoing Links Editor” on page 10-7, this string appears on the Document type drop-down list for a Simulink or Stateflow object.
IsFile	A Boolean property that indicates if the linked documents are files within the computer file system. If a document is a file: <ul style="list-style-type: none"> • The software uses the standard method for resolving the path. • In the Outgoing Links Editor, when you click Browse, the file selection dialog box opens.

Property	Description
Extensions	An array of file extensions. Use these file extensions as filter options in the Outgoing Links Editor when you click Browse . The file extensions infer the link type based on the document name. If you registered more than one link type for the same file extension, the link type that you registered takes first priority.
LocDelimiters	A string containing the list of supported navigation delimiters. The first character in the ID of a requirement specifies the type of identifier. For example, an identifier can refer to a specific page number (#4), a named bookmark (@my_tag), or some searchable text (?search_text). The valid location delimiters determine the possible entries in the Outgoing Links Editor Location drop-down list.
NavigateFcn	The MATLAB callback invoked when you click a link. The function has two input arguments: the document field and the ID field of the link: <code>feval(LinkType.NavigateFcn, Link.document, Link.id)</code>
ContentsFcn	The MATLAB callback invoked when you click the Document Index tab in the Outgoing Links Editor. This function has a single input argument that contains the full path of the resolved function or, if the link type is not a file, the Document field contents. The function returns three outputs: <ul style="list-style-type: none"> • Labels • Depths • Locations
BrowseFcn	The MATLAB callback invoked when you click Browse in the Outgoing Links Editor. You do not need this function when the link type is a file. The function takes no input arguments and returns a single output argument that identifies the selected document.

Property	Description
CreateURLFcn	<p>The MATLAB callback that constructs a path name to the requirement. This function uses the document path or URL to create a specific requirement URL. The requirement URL is based on a location identifier specified in the third input argument. The input arguments are:</p> <ul style="list-style-type: none"> • Full path name to the requirements document • Info about creating a URL to the document (if applicable) • Location of the requirement in the document <p>This function returns a single output argument specified as a character vector. Use this argument when navigating to the requirement from the generated report.</p>
IsValidDocFcn	<p>The MATLAB callback invoked when you run a requirements consistency check. The function takes one input argument—the fully qualified name for the requirements document. It returns true if the document can be located; it returns false if the document cannot be found or the document name is invalid.</p>
IsValidIdFcn	<p>The MATLAB callback invoked when you run a requirements consistency check. This function takes two input arguments:</p> <ul style="list-style-type: none"> • Fully qualified name for the requirements document • Location of the requirement in the document <p>IsValidIdFcn returns true if it finds the requirement and false if it cannot find that requirement in the specified document.</p>

Property	Description
IsValidDescFcn	<p>The MATLAB callback invoked when you run a requirements consistency check. This function has three input arguments:</p> <ul style="list-style-type: none"> • Full path to the requirements document • Location of the requirement in the document • Requirement description label as stored in Simulink <p>IsValidDescFcn returns two outputs:</p> <ul style="list-style-type: none"> • True if the description matches the requirement, false otherwise. • The requirement label in the document, if not matched in Simulink.
DetailsFcn	<p>The MATLAB callback invoked when you generate the requirements report with the Include details from linked documents option. This function returns detailed content associated with the requirement and has three input arguments:</p> <ul style="list-style-type: none"> • Full path to the requirements document • Location of the requirement in the document • Level of details to include in report (Unused) <p>The DetailsFcn returns two outputs:</p> <ul style="list-style-type: none"> • Numeric array that describes the hierarchical relationship among the fragments in the cell array • Cell array of formatted fragments (paragraphs, tables, et al.) from the requirement

Property	Description
SelectionLinkFcn	<p>The MATLAB callback invoked when you use the selection-based linking menu option for this document type. This function has two input arguments:</p> <ul style="list-style-type: none"> • Handle to the model object that will have the requirement link • True if a navigation object is inserted into the requirements document, or false if no navigation object is inserted <p>SelectionLinkFcn returns the requirements link structure for the selected requirement.</p>

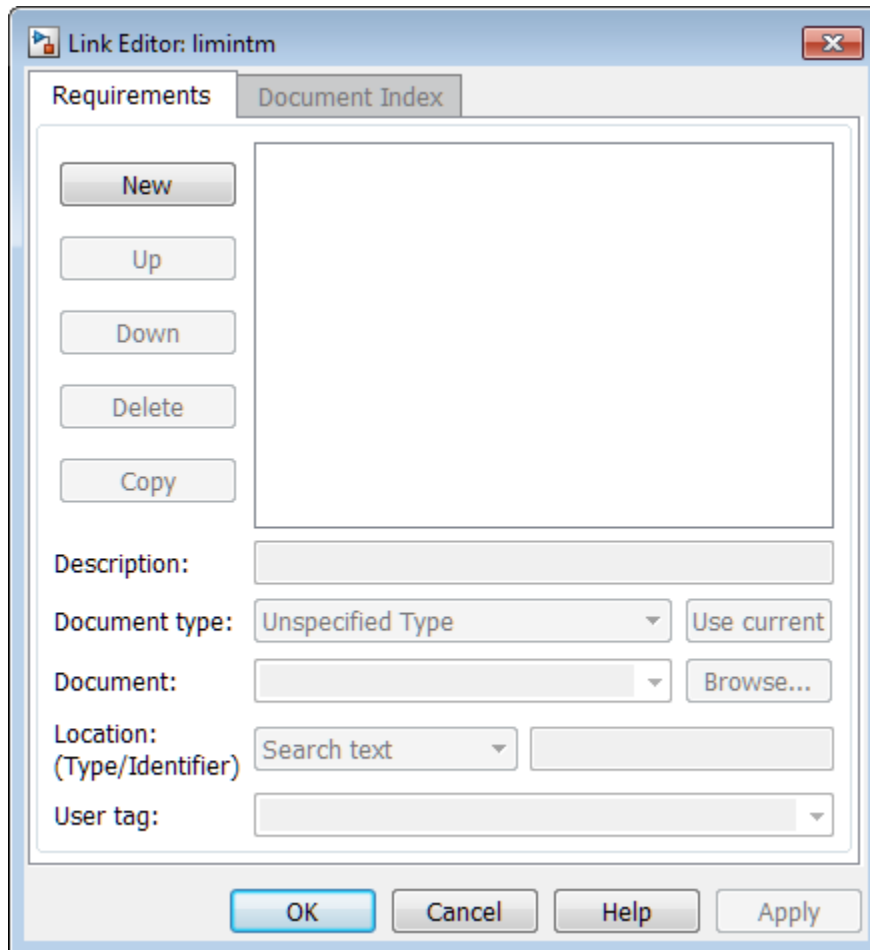
Outgoing Links Editor

Manage Requirements Traceability Links Using the Outgoing Links Editor

You can create, edit, and delete requirements traceability links using the Outgoing Links Editor. To open the Outgoing Links Editor:

- in the Simulink Editor, right-click on a model object that has a requirements traceability link. From the context menu, select **Requirements > Open Outgoing Links dialog**.
- in the MATLAB Editor, right-click inside a region of code that has a requirements traceability link. From the context menu, select **Requirements > Open Outgoing Links dialog**.

The Outgoing Links Editor opens, as shown below.



In the Outgoing Links Editor, you can:

- Create requirements links from one or more Simulink model objects or MATLAB code lines.
- Customize information about requirements links, including specifying user tags to filter requirements highlighting and reporting.
- Delete existing requirements links.
- Modify the stored order of requirements to control the order of labels in context menus for linked objects.

Requirements Tab

On the **Requirements** tab, you specify detailed information about the link, including:

- Description of the requirement (up to 255 words). If you create a link using the document index, *unless* a description already exists, the name of the index location becomes the description for the link .
- Path name to the requirements document.
- Document type (Microsoft Word, Microsoft Excel, IBM Rational DOORS, MuPAD®, HTML, text file, etc.).
- Location of the requirement (search text, named location, or page or item number).
- User-specified tag or keyword.

Document Index Tab

The **Document Index** tab is available only if you have specified a file in the **Document** field on the **Requirements** tab that supports indexing. On the **Document Index** tab, the RMI generates a list of locations in the specified requirements document for the following types of requirements documents:

- Microsoft Word
- IBM Rational DOORS
- HTML files
- MuPAD

Note The RMI cannot create document indexes for PDF files.

From the document index, select the desired requirement from the document index and click **OK**. *Unless* a description already exists, the name of the index location becomes the description for the link.

If you make any changes to your requirements document, to load any newly created locations into the document index, you must click **Refresh**. During a MATLAB session, the RMI does not reload the document index unless you click the **Refresh** button.

Custom Link Types

Create a Custom Requirements Link Type

In this example, you implement a custom link type to a hypothetical document type, a text file with the extension `.abc`. Within this document, the requirement items are identified with a special text string, `Requirement : :`, followed by a single space and then the requirement item inside quotation marks (`"`).

You will create a document index listing all the requirement items. When navigating from the Simulink model to the requirements document, the document opens in the MATLAB Editor at the line of the requirement that you want.

To create a custom link requirement type:

- 1 Write a function that implements the custom link type and save it on the MATLAB path.

For this example, the file is `rmicustabcinterface.m`, containing the function, `rmicustabcinterface`, that implements the ABC files shipping with your installation.

- 2 To view this function, at the MATLAB prompt, type:

```
edit rmicustabcinterface
```

The file `rmicustabcinterface.m` opens in the MATLAB Editor. The content of the file is:

```
function linkType = rmicustabcinterface
%RMICUSTABCINTERFACE - Example custom requirement link type
%
% This file implements a requirements link type that maps
% to "ABC" files.
% You can use this link type to map a line or item within an ABC
% file to a Simulink or Stateflow object.
%
% You must register a custom requirement link type before using it.
% Once registered, the link type will be reloaded in subsequent
% sessions until you unregister it. The following commands
% perform registration and registration removal.
%
% Register command:  >> rmi register rmicustabcinterface
% Unregister command: >> rmi unregister rmicustabcinterface
%
% There is an example document of this link type contained in the
% requirement demo directory to determine the path to the document
% invoke:
%
```

```

% >> which demo_req_1.abc

% Copyright 1984-2010 The MathWorks, Inc.

% Create a default (blank) requirement link type
linkType = ReqMgr.LinkType;
linkType.Registration = mfilename;

% Label describing this link type
linkType.Label = 'ABC file (for demonstration)';

% File information
linkType.IsFile = 1;
linkType.Extensions = {'.abc'};

% Location delimiters
linkType.LocDelimiters = '>@';
linkType.Version = ''; % not required

% Uncomment the functions that are implemented below
linkType.NavigateFcn = @NavigateFcn;
linkType.ContentsFcn = @ContentsFcn;

function NavigateFcn(filename,locationStr)
if ~isempty(locationStr)
    findId=0;
    switch(locationStr(1))
    case '>'
        lineNum = str2num(locationStr(2:end));
        openFileToLine(filename, lineNum);
    case '@'
        openFileToItem(filename,locationStr(2:end));
    otherwise
        openFileToLine(filename, 1);
    end
end

function openFileToLine(fileName, lineNum)
if lineNum > 0
    if matlab.desktop.editor.isEditorAvailable
        matlab.desktop.editor.openAndGoToLine(fileName, lineNum);
    end
else
    edit(fileName);
end

function openFileToItem(fileName, itemName)
reqStr = ['Requirement:: ' itemName ''];
lineNum = 0;
fid = fopen(fileName);
i = 1;
while lineNum == 0
    lineStr = fgetl(fid);
    if ~isempty(strfind(lineStr, reqStr))
        lineNum = i;
    end;
    if ~ischar(lineStr), break, end;
    i = i + 1;
end;

```

```
fclose(fid);
openFileToLine(fileName, lineNum);

function [labels, depths, locations] = ContentsFcn(filePath)
% Read the entire file into a variable
fid = fopen(filePath,'r');
contents = char(fread(fid));
fclose(fid);

% Find all the requirement items
fList1 = regexp(contents,'\nRequirement:: "(.*?)"', 'tokens');

% Combine and sort the list
items = [fList1{:}];
items = sort(items);
items = strcat('@',items);

if (~iscell(items) && length(items)>0)
    locations = {items};
    labels = {items};
else
    locations = [items];
    labels = [items];
end

depths = [];
```

- 3** To register the custom link type ABC, type the following MATLAB command:

```
rmi register rmicustabcinterface
```

The ABC file type appears on the “Outgoing Links Editor” on page 10-7 drop-down list of document types.

- 4** Create a text file with the .abc extension containing several requirement items marked by the Requirement:: string.

For your convenience, an example file ships with your installation. The example file is *matlabroot\toolbox\slvnv\rmidemos\demo_req_1.abc*. *demo_req_1.abc* contains the following content:

```
Requirement:: "Altitude Climb Control"
```

```
Altitude climb control is entered whenever:
|Actual Altitude- Desired Altitude | > 1500
```

```
Units:
Actual Altitude - feet
Desired Altitude - feet
```

```
Description:
```


When the autopilot is in altitude climb control mode, the controller maintains a constant user-selectable target climb rate.

The user-selectable climb rate is always a positive number if the current altitude is above the target altitude. The actual target climb rate is the negative of the user setting.

End of "Altitude Climb Control">

Requirement:: "Altitude Hold"

Altitude hold mode is entered whenever:
 $| \text{Actual Altitude} - \text{Desired Altitude} | < 30 * \text{Sample Period} * (\text{Pilot Climb Rate} / 60)$

Units:

Actual Altitude - feet

Desired Altitude - feet

Sample Period - seconds

Pilot Climb Rate - feet/minute

Description:

The transition from climb mode to altitude hold is based on a threshold that is proportional to the Pilot Climb Rate.

At higher climb rates the transition occurs sooner to prevent excessive overshoot.

End of "Altitude Hold"

Requirement:: "Autopilot Disable"

Altitude hold control and altitude climb control are disabled when autopilot enable is false.

Description:

Both control modes of the autopilot can be disabled with a pilot setting.

END of "Autopilot Disable"

Requirement:: "Glide Slope Armed"

Glide Slope Control is armed when Glide Slope Enable and Glide Slope Signal are both true.

Units:

Glide Slope Enable - Logical

Glide Slope Signal - Logical

Description:

ILS Glide Slope Control of altitude is only enabled when the pilot has enabled this mode and the Glide Slope Signal is true. This indicates the Glide Slope broadcast signal has been validated by the on board receiver.

End of "Glide Slope Armed"

Requirement:: "Glide Slope Coupled"

Glide Slope control becomes coupled when the control is armed and (Glide Slope Angle Error > 0) and Distance < 10000

Units:

Glide Slope Angle Error - Logical

Distance - feet

Description:

When the autopilot is in altitude climb control mode the controller maintains a constant user selectable target climb rate.

The user-selectable climb rate is always a positive number if the current altitude is above the target altitude the actual target climb rate is the negative of the user setting.

End of "Glide Slope Coupled"

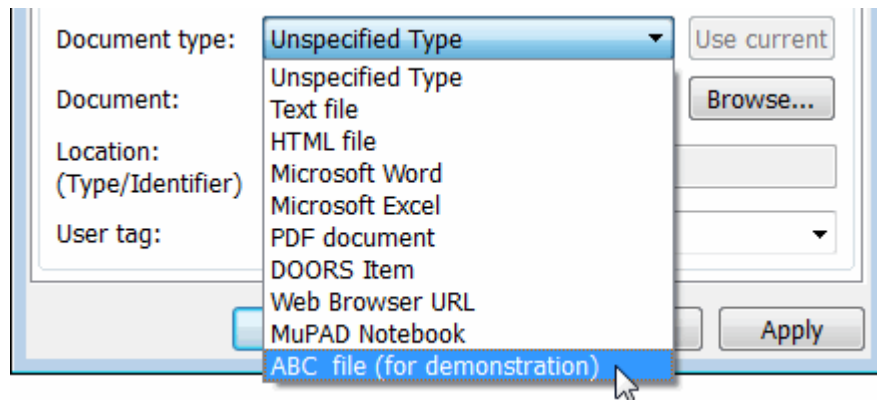
- 5 Open the following example model:

aero_dap3dof

- 6 Right-click the Reaction Jet Control subsystem and select **Requirements > Open Outgoing Links dialog**.

The Outgoing Links Editor opens.

- 7 Click **New** to add a new requirement link. The **Document type** drop-down list now contains the ABC file (for demonstration) option.



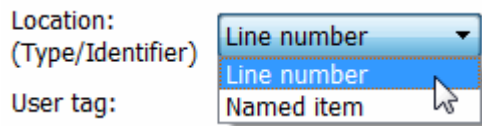
- 8 Set **Document type** to ABC file (for demonstration) and browse to the *matlabroot\toolbox\slvnx\rmidemos\demo_req_1.abc* file. The browser shows only the files with the .abc extension.
- 9 To define a particular location in the requirements document, use the **Location** field.

In this example, the `rmicustabcinterface` function specifies two types of location delimiters for your requirements:

- > — Line number in a file
- @ — Named item, such as a bookmark, function, or HTML anchor

Note The rmi reference page describes other types of requirements location delimiters.

The **Location** drop-down list contains these two types of location delimiters whenever you set **Document type** to ABC file (for demonstration).



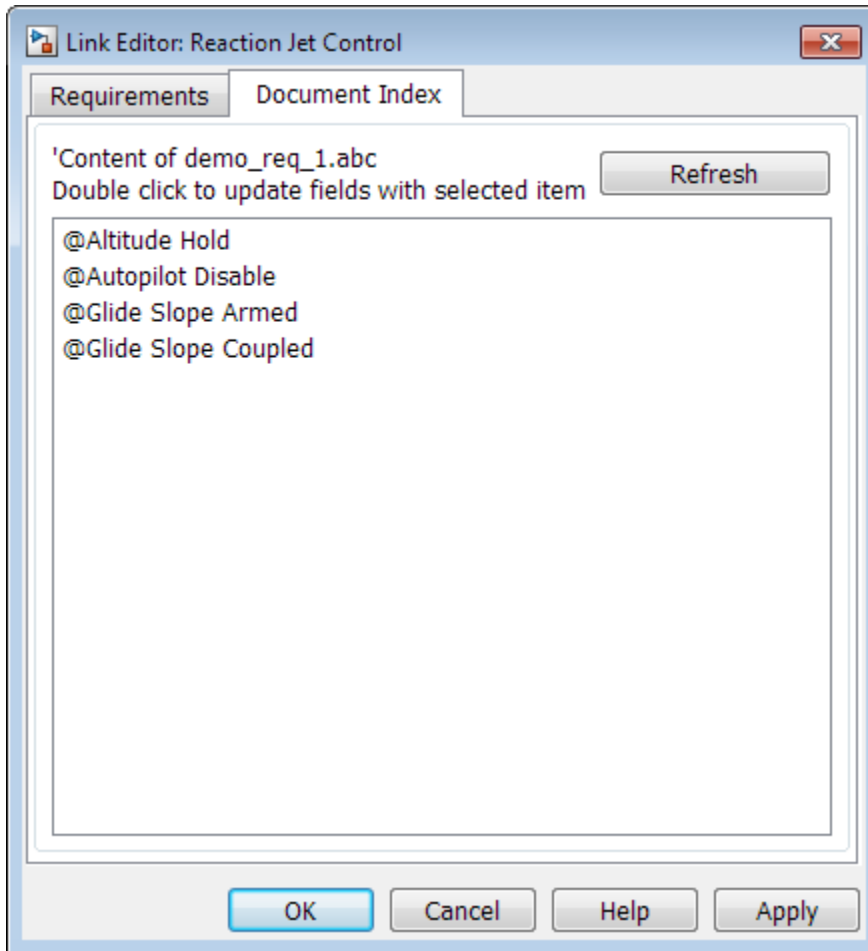
- 10** Select **Line number**. Enter the number 26, which corresponds with the line number for the Altitude Hold requirement in demo_req_1.abc.
- 11** In the **Description** field, enter Altitude Hold, to identify the requirement by name.
- 12** Click **Apply**.
- 13** Verify that the Altitude Hold requirement links to the Reaction Jet Control subsystem. Right-click the subsystem and select **Requirements > 1. "Altitude Hold"**.

Create a Document Index

A *document index* is a list of all the requirements in a given document. To create a document index, MATLAB uses file I/O functions to read the contents of a requirements document into a MATLAB variable. The RMI extracts the list of requirement items.

The example requirements document, demo_req_1.abc, defines four requirements using the string Requirement::. To generate the document index for this ABC file, the ContentsFcn function in rmicustabcinterface.m extracts the requirements names and inserts @ before each name.

For the demo_req_1.abc file, in the **Outgoing Links: Reaction Jet Control** dialog box, click the **Document Index** tab. The ContentsFcn function generates the document index automatically.



Implement Custom Link Types

To implement a custom link type:

- 1 Create a MATLAB function file based on the custom link type template, as described in "Custom Link Type Functions" on page 10-18.
- 2 Customize the custom link type file to specify the link type properties and custom callback functions required for the custom link type, as described in "Link Type Properties" on page 10-3.

- 3 Register the custom link type using the `rmi` command 'register' option, as described in “Custom Link Type Registration” on page 10-19.

Why Create a Custom Link Type?

In addition to linking to built-in types of requirements documents, you can register custom requirements document types with the Requirements Management Interface (RMI). Then you can create requirement links from your model to these types of documents.

With custom link types, you can:

- Link to requirement items in commercial requirement tracking software
- Link to in-house database systems
- Link to document types that the RMI does not support

The custom link type API allows you to define MATLAB functions that enable linking between your Simulink model and your custom requirements document type. These functions also enable new link creation and navigation between the model and documents.

For example, navigation involves opening a requirements document and finding the specific requirement record. When you click your custom link in the content menu of a linked object in the model, Simulink uses your custom link type navigation function to open the document and highlight the target requirement based on the implementation provided. The navigation function you implement uses the available API to communicate with your requirements storage application.

Typically, MATLAB runs an operating system shell command or uses ActiveX communication for sending navigation requests to external applications.

Alternatively, if your requirements are stored as custom variants of text or HTML files, you can use the built-in editor or Web browser to open the requirements document.

Custom Link Type Functions

To create a MATLAB function file, start with the custom link type template, located in:

`matlabroot\toolbox\slrequirements\linktype_examples\linktype_TEMPLATE.m`

Your custom link type function:

- Must exist on the MATLAB path with a unique function and file name.
- Cannot require input arguments.
- Must return a single output argument that is an instance of the requirements link type class.

To view similar files for the built-in link types, see the following files in *matlabroot* \toolbox\slrequirements\linktype_examples\:

```
linktype_rmi_doors.m  
linktype_rmi_excel.m  
linktype_rmi_html.m  
linktype_rmi_text.m
```

Custom Link Type Registration

Register your custom link type by passing the name of the MATLAB function file to the `rmi` command as follows:

```
rmi register mytargetfilename
```

Once you register a link type, it appears in the “Outgoing Links Editor” on page 10-7 as an entry in the **Document type** drop-down list. A file in your preference folder contains the list of registered link types, so the custom link type is loaded each time you run MATLAB.

When you create links using custom link types, the software saves the registration name and the other link properties specified in the function file. When you attempt to navigate to such a link, the RMI resolves the link type against the registered list. If the software cannot find the link type, you see an error message.

You can remove a link type with the following MATLAB command:

```
rmi unregister mytargetfilename
```

Custom Link Type Synchronization

After you implement custom link types for RMI that allow you to establish links from Simulink objects to requirements in your requirements management application (RM application), you can implement synchronization of the links between the RM application and Simulink using Simulink Requirements functions. Links can then be reviewed and

managed in your RM application environment, while changes made are propagated to Simulink.

You first create the surrogate objects in the RM application to represent Simulink objects of interest. You then automate the process of establishing traceability links between these surrogate objects and other items stored in the RM application, to match links that exist on the Simulink side. After modifying or creating new associations in the RM application, you can propagate the changes back to Simulink. You use Simulink Requirements to implement synchronization of links for custom requirements documents. However, this functionality is dependent upon the automation and inter-process communication APIs available in your RM application. You use the following Simulink Requirements functions to implement synchronization of links between RM applications and Simulink.

To get a complete list of Simulink objects that may be considered for inclusion in the surrogate module:

```
[objHs, parentIdx, isSf, objSIDs] = rmi...  
( 'getObjectsInModel', modelName);
```

This command returns:

- `objHs`, a complete list of numeric handles
- `objSIDs`, a complete list of corresponding session-independent Simulink IDs
- `isSf`, a logical array that indicates which list positions correspond to which Stateflow objects
- `parentIdx`, an array of indices that provides model hierarchy information

When creating surrogate objects in your RM application, you will need to store `objSIDs` values - not `objHs` values - because `objHs` values are not persistent between Simulink sessions.

To get Simulink object Name and Type information that you store on the RM application side:

```
[objName, objType] = rmi('getObjLabel', sLObjectHandle);
```

To query links for a Simulink object, specified by either numeric handle or SID:

```
linkInfo = rmi('getLinks', sLObjectHandle)  
linkInfo = rmi('getLinks', sigBuildertHandle, m)  
% Signal Builder group "m" use case.  
linkInfo = rmi('getLinks', [modelName objSIDs{i}]);
```


`linkInfo` is a MATLAB structure that contains link attributes. See the `rmi` function reference page for more details.

After you retrieve the updated link information from your RM application, populate the fields of `linkData` with the updated values, and propagate the changes to Simulink:

```
rmi('setLinks', sObjectHandle, linkData)
```

For an example MATLAB script implementing synchronization with a Microsoft Excel Workbook, see the following:

```
edit([matlabroot '/toolbox/slrequirements/...  
linktype_examples/slSurrogateInExcel.m'])
```

You can run this MATLAB script on the example model `slvndemo_fuelsys_officereq` to generate the Excel workbook surrogate for the model.

Review and Maintain Requirements Links

- “Highlight Model Objects with Requirements” on page 11-2
- “Navigate to Simulink Objects from External Documents” on page 11-5
- “View Requirements Details for a Selected Block” on page 11-8
- “Generate Code for Models with Requirements Links” on page 11-10
- “Create and Customize Requirements Traceability Reports” on page 11-13
- “Create Requirements Traceability Report for A Project” on page 11-33
- “Validate Requirements Links” on page 11-34
- “Delete Requirements Links from Simulink Objects” on page 11-46
- “Document Path Storage” on page 11-48

Highlight Model Objects with Requirements

To review traceability in your model, you can highlight model objects that have requirements links.

Highlight Model Objects with Requirements Using Model Editor

If you are working in the Simulink Editor and want to see which model objects in the `slvndemo_fuelsys_officereq` model have requirements, follow these steps:

- 1 Open the example model:

```
slvndemo_fuelsys_officereq
```

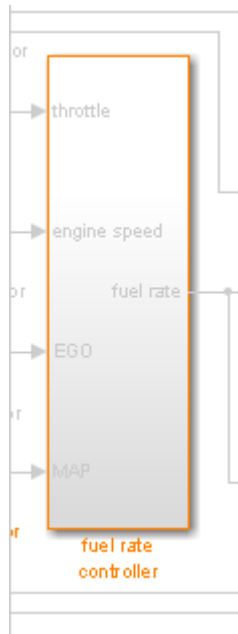
- 2 Select **Analysis > Requirements > Highlight Model**.

Two types of highlighting indicate model objects with requirements:

- Yellow highlighting indicates objects that have requirements links for the object itself.



- Orange outline indicates objects, such as subsystems, whose child objects have requirements links.



Objects that do not have requirements are colored gray.




- 3 To remove the highlighting from the model, select **Analysis > Requirements > Unhighlight Model**. Alternatively, you can right-click anywhere in the model, and select **Remove Highlighting**.

While a model is highlighted, you can still manage the model and its contents.

Highlight Model Objects with Requirements Using Model Explorer

If you are working in Model Explorer and want to see which model objects have requirements, follow these steps:

- 1 Open the example model:
`slvndemo_fuelsys_officereq`
- 2 Select **View > Model Explorer**.
- 3 To highlight all model objects with requirements, click the **Highlight items with requirements on model** icon ().

The Simulink Editor window opens, and all objects in the model with requirements are highlighted.

Note If you are running a 64-bit version of MATLAB, when you navigate to a requirement in a PDF file, the file opens at the beginning of the document, not at the specified location.

Navigate to Simulink Objects from External Documents

The RMI includes several functions that simplify creating navigation interfaces in external documents. The external application that displays your document must support an application programming interface (API) for communicating with the MATLAB software.

Provide Unique Object Identifiers

Whenever you create a requirement link for a Simulink or Stateflow object, the RMI uses a globally unique identifier for that object. This identifier identifies the object. The identifier does not change if you rename or move the object, or add or delete requirement links. The RMI uses the unique identifier only to resolve an object within a model.

Use the `rmiobjnavigate` Function

The `rmiobjnavigate` function identifies the Simulink or Stateflow object, highlights that object, and brings the editor window to the front of the screen. When you navigate to a Simulink model from an external application, invoke this function.

The first time you navigate to an item in a particular model, you might experience a slight delay while the software initializes the communication API and the internal data structures. You do not experience a long delay on subsequent navigation.

Determine the Navigation Command

To create a requirement link for a Simulink or Stateflow object, at the MATLAB prompt, use the following command to find the navigation command, where `obj` is a handle or a uniquely resolved name for the object:

```
[ navCmd, objPath ] = rmi('navCmd', obj);
```

The return values of the `navCmd` method are:

- `navCmd` — A character vector that navigates to the object when evaluated by the MATLAB software.
- `objPath` — A character vector that identifies the model object.

Send `navCmd` to the MATLAB software for evaluation when navigating from the external application to the object `obj` in the Simulink model. Use `objPath` to visually identify the target object in the requirements document.

Use the ActiveX Navigation Control

The RMI uses software that includes a special Microsoft ActiveX control to enable navigation to Simulink objects from Microsoft Word and Excel documents. You can use this same control in any other application that supports ActiveX within its documents.

The control is derived from a push button and has the Simulink icon. There are two instance properties that define how the control works. The `tooltipstring` property is displayed in the control tooltip. The `MLEvalCmd` property is the character vector that you pass to the MATLAB software for evaluation when you click the control.

Typical Code Sequence for Establishing Navigation Controls

When you create an interface to an external tool, you can automate the procedure for establishing links. This way, you do not need to manually update the dialog box fields. This type of automation occurs as part of the selection-based linking for certain built-in types, such as Microsoft Word and Excel documents.

To automate the procedure for establishing links:

- 1** Select a Simulink or Stateflow object and an item in the external document.
- 2** Invoke the link creation action either from a Simulink menu or command, or a similar mechanism in the external application.
- 3** Identify the document and current item using the scripting capability of the external tool. Pass this information to the MATLAB software. Create a requirement link on the selected object using the RMI API as follows:

- a** Create an empty link structure using the following command:

```
rmi('createempty')
```

- b** Fill in the link structure fields based on the target location in the requirements document.

- c** Attach the link to the object using the following command:

```
rmi('cat')
```

- 4** Determine the MATLAB navigation command that you must embed in the external tool, using the `navCmd` method:

```
[ navCmd, objPath ] = rmi('navCmd',obj)
```


- 5 Create a navigation item in the external document using the scripting capability of the external tool. Set the MATLAB navigation command in the property.

When using ActiveX navigation objects provided by the external tool, set the `MLEvalCmd` property to the `navCmd` and set the `tooltipstring` property to `objPath`.

You define the MATLAB code implementation of this procedure as the `SelectionLinkFcn` function in the link type definition file. The following files in `matlabroot\toolbox\shared\reqmgt\linktypes` contain examples of how to implement this functionality:

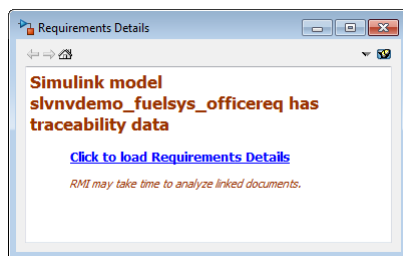
```
linktype_rmi_doors.m  
linktype_rmi_excel.m  
linktype_rmi_html.m  
linktype_rmi_text.m
```

View Requirements Details for a Selected Block

Requirements Details Workflow

When you highlight model objects with requirements, you can use the RMI Informer window to view the requirements details for a selected block. Follow this workflow:

- 1 From the menu, select **Analysis > Requirements > Highlight Model**.
- 2 In your model, highlights indicate model objects with requirements links. The RMI Informer window opens.



- 3 In the RMI Informer window, click the link to load the requirements details. If your Simulink model has links to Microsoft Word, Microsoft Excel, or IBM Rational DOORS documents, the RMI Informer displays requirements context from the requirements documents and additional link labels stored by Simulink.
- 4 Select highlighted model objects to display associated RMI links in the RMI Informer window.
- 5 Close the requirements details window to remove highlights from the Simulink model.

For more information see “Navigate to Requirements from Model” on page 8-20.

Requirements Details Limitations

Security restrictions in Microsoft Office can interrupt the requirements details loading process. Requirement details loaded from read-only Microsoft Office documents, documents stored on network drives, and documents with Microsoft Office Trust Center ActiveX control restrictions may not work with RMI. Consider enabling ActiveX controls without prompting and using requirements stored in a writable location on the MATLAB path.

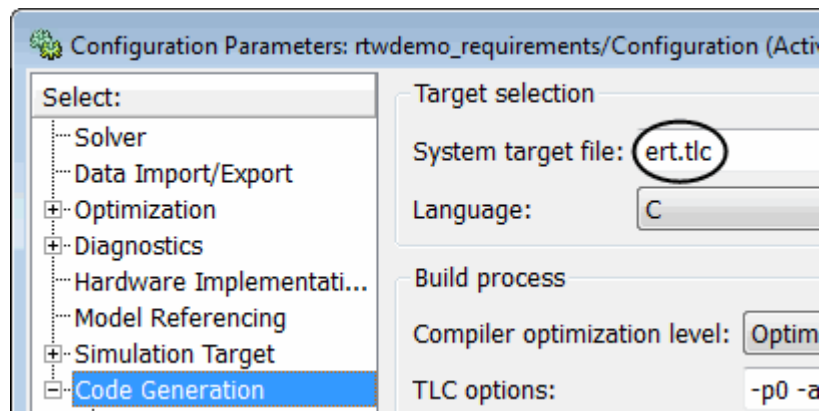
Before loading requirements details for IBM Rational DOORS links, you must be logged in to the IBM Rational DOORS Client.

Generate Code for Models with Requirements Links

To specify that generated code of an ERT target include requirements:

- 1 Open the `rtwdemo_requirements` example model.
- 2 Select **Simulation > Model Configuration Parameters**.
- 3 In the **Select** tree of the Configuration Parameters dialog box, select the **Code Generation** node.

The currently configured system target must be an ERT target.



- 4 Under **Code Generation**, select **Comments**.
- 5 In the **Custom comments** section on the right, select the **Requirements in block comments** check box.
- 6 Under **Code Generation**, select **Report**.
- 7 On the **Report** pane, select:
 - **Create code generation report**
 - **Open report automatically**
- 8 Press **Ctrl+B** to build the model.
- 9 In the code-generation report, open `rtwdemo_requirements.c`.
- 10 Scroll to the code for the Pulse Generator block, `clock`. The comments for the code associated with that block include a hyperlink to the requirement linked to that block.

```

119  rtwdemo_requirements.c
120  /* DiscretePulseGenerator: '<Root>/clock'
121  *
122  * Block requirements for '<Root>/clock':
123  * 1. Clock period shall be consistent with chirp tolerance
124  */

```

- 11 Click the link [Clock period shall be consistent with chirp tolerance](#) to open the HTML requirements document to the associated requirement.

Note When you click a requirements link in the code comments, the software opens the application for the requirements document, *except* if the requirements document is a DOORS module. To view a DOORS requirement, start the DOORS software and log in before clicking the hyperlink in the code comments.

How Requirements Information Is Included in Generated Code

After you simulate your model and verify its performance against the requirements, you can generate code from the model for an embedded real-time application. The Embedded Coder software generates code for Embedded Real-Time (ERT) targets.

If the model has any links to requirements, the Embedded Coder software inserts information about the requirements links into the code comments.

For example, if a block has a requirement link, the software generates code for that block. In the code comments for that block, the software inserts:

- Requirement description
- Hyperlink to the requirements document that contains the linked requirement associated with that block

Note

- You must have a license for Embedded Coder to generate code for an embedded real-time application.
 - If you use an external `.req` file to store your requirement links, to avoid stale comments in generated code, before code generation, you must save any change in your requirement links. For information on how to save, see “Save Requirements Links in External Storage” on page 5-5.
-

Comments for the generated code include requirements descriptions and hyperlinks to the requirements documents in the following locations.

Model Object with Requirement	Location of Code Comments with Requirements Links
Model	In the main header file, <model>.h
Nonvirtual subsystem	At the call site for the subsystem
Virtual subsystem	At the call site of the closest nonvirtual parent subsystem. If a virtual subsystem does not have a nonvirtual parent, requirement descriptions appear in the main header file for the model, <model>.h.
Nonsubsystem block	In the generated code for the block
MATLAB code line in MATLAB Function block	In the generated code for the MATLAB code line(s)

Create and Customize Requirements Traceability Reports

Create Requirements Traceability Report for Model

To create the default requirements report for a Simulink model:

- 1 Open the example model:
`slvndemo_fuelsys_officereq`
- 2 Make sure that your current working folder is writable.
- 3 In the Simulink Editor, select **Analysis > Requirements > Generate Report**.

If your model is large and has many requirements links, it takes a few minutes to create the report.

A Web browser window opens with the contents of the report. The following graphic shows the **Table of Contents** for the `slvndemo_fuelsys_officereq` model.

Requirements Report for slvndemo_fuelsys_officereq

username

17-Jun-2010 10:57:04

Table of Contents

- [1. Model Information for "slvndemo_fuelsys_officereq"](#)
- [2. Document Summary for "slvndemo_fuelsys_officereq"](#)
- [3. System - slvndemo_fuelsys_officereq](#)
- [4. System - engine gas dynamics](#)
- [5. System - fuel rate controller](#)
- [6. System - Mixing & Combustion](#)
- [7. System - Airflow calculation](#)
- [8. System - Sensor correction and Fault Redundancy](#)
- [9. System - MAP Estimate](#)
- [10. Chart - control logic](#)

A typical requirements report includes:

- Table of contents
- List of tables
- Per-subsystem sections that include:
 - Tables that list objects with requirements and include links to associated requirements documents
 - Graphic images of objects with requirements
 - Lists of objects with no requirements
 - MATLAB code lines with requirements in MATLAB Function blocks

For detailed information about requirements reports, see “Customize Requirements Traceability Report for Model” on page 11-15.

If Your Model Has Library Reference Blocks

To include requirements links associated with library reference blocks, you must select **Include links in referenced libraries and data dictionaries** under the **Report** tab of the **Requirements Settings**, as described in “Customize Requirements Report” on page 11-25.

If Your Model Has Model Reference Blocks

By default, requirements links within model reference blocks in your model are not included in requirements traceability reports. To generate a report that includes requirements information for referenced models, follow the steps in “Report for Requirements in Model Blocks” on page 11-23.

Customize Requirements Traceability Report for Model

Create Default Requirements Report

If you have a model that contains links to external requirements documents, you can create an HTML report that contains summarized and detailed information about those links. In addition, the report contains links that allow you to navigate to both the model and to the requirements documents.

You can generate a default report with information about all the requirements associated with a model and its objects.

Note If the model for which you are creating a report contains Model blocks, see “Report for Requirements in Model Blocks” on page 11-23.

Before you generate the report, add a requirement to a Stateflow chart to see information that the requirements report contains about Stateflow charts:

- 1 Open the example model:

```
slvndemo_fuelsys_officereq
```

- 2 Open the fuel rate controller subsystem.

- 3 Open the Microsoft Word requirements document:

```
matlabroot/toolbox/slvnv/rmidemos/fuelsys_req_docs/...  
slvndemo_FuelSys_RequirementsSpecification.docx
```

- 4 Create a link from the control logic Stateflow chart to a location in this document.
- 5 Keep the example model open, but close the requirements document.

To generate a default requirements report for the `slvndemo_fuelsys_officereq` model:

- 1 Select **Analysis > Requirements > Reports > Generate Report**.

The Requirements Management Interface (RMI) searches through all the blocks and subsystems in the model for associated requirements. The RMI generates and displays a complete report in HTML format.

The report is saved with the default name, `model_name_requirements.html`. If you generate a subsequent report on the same model, the new report file overwrites any earlier report file.

The report contains the following content:

Table of Contents

The **Table of Contents** lists the major sections of the report. There is one **System** section for the top-level model and one **System** section for each subsystem, Model block, or Stateflow chart.

Click a link to view information about a specific section of the model.

Requirements Report for slnvdemo_fuelsys_officereq

username

17-Jun-2010 10:57:04

Table of Contents

- [1. Model Information for "slnvdemo_fuelsys_officereq"](#)
- [2. Document Summary for "slnvdemo_fuelsys_officereq"](#)
- [3. System - slnvdemo_fuelsys_officereq](#)
- [4. System - engine gas dynamics](#)
- [5. System - fuel rate controller](#)
- [6. System - Mixing & Combustion](#)
- [7. System - Airflow calculation](#)
- [8. System - Sensor correction and Fault Redundancy](#)
- [9. System - MAP Estimate](#)
- [10. Chart - control logic](#)

List of Tables

The **List of Tables** includes links to each table in the report.

List of Tables

- 1.1. [slnvdemo_fuelsys_officereq Version Information](#)
- 2.1. [Requirements documents linked in model](#)
- 3.1. [slnvdemo_fuelsys_officereq Requirements](#)
- 3.2. [Blocks in "slnvdemo_fuelsys_officereq" that have requirements](#)
- 3.3. [Test inputs : Normal operation signal requirements](#)
- 4.1. [Blocks in "engine gas dynamics" that have requirements](#)
- 5.1. [Blocks in "fuel rate controller" that have requirements](#)
- 6.1. [Blocks in "Mixing & Combustion" that have requirements](#)
- 7.1. [slnvdemo_fuelsys_officereq/fuel rate controller/Airflow calculation Requirements](#)
- 7.2. [Blocks in "Airflow calculation" that have requirements](#)
- 8.1. [Blocks in "Sensor correction and Fault Redundancy" that have requirements](#)
- 9.1. [slnvdemo_fuelsys_officereq/fuel rate controller/Sensor correction and Fault Redundancy/MAP Estimate Requirements](#)
- 10.1. [Stateflow objects with requirements](#)

Model Information

The **Model Information** contains general information about the model, such as when the model was created and when the model was last modified.

Chapter 1. Model Information for "slvnvdemo_fuelsys_officereq"

Table 1.1. slvnvdemo_fuelsys_officereq Version Information

<i>ModelVersion</i>	1.159	<i>ConfigurationManager</i>	none
<i>Created</i>	Tue Jun 02 16:11:43 1998	<i>Creator</i>	The MathWorks Inc.
<i>LastModifiedDate</i>	Sat Jun 12 02:31:44 2010	<i>LastModifiedBy</i>	

Documents Summary

The **Documents Summary** section lists all the requirements documents to which objects in the slvnvdemo_fuelsys_officereq model link, along with some additional information about each document.

Chapter 2. Document Summary for "slvnvdemo_fuelsys_officereq"

Table 2.1. Requirements documents linked in model

ID	Document paths stored in the model	Last modified	# links
DOC1	ERROR: unable to locate slvnvdemo_FuelSys_RequirementsSpecification.docx	unknown	1
DOC2	fuelsys_req_docs\slvnvdemo_FuelSys_DesignDescription.docx	29-Oct-2009 10:56:01	8
DOC3	fuelsys_req_docs\slvnvdemo_FuelSys_RequirementsSpecification.docx	29-Oct-2009 10:56:02	6
DOC4	fuelsys_req_docs\slvnvdemo_FuelSys_TestScenarios.xlsx	29-Oct-2009 10:56:03	2

- **ID** — The ID. In this example, **DOC1**, **DOC2**, **DOC3**, and **DOC4** are short names for the requirements documents linked from this model.

Before you generate a report, in the Settings dialog box, on the **Reports** tab, if you select **User document IDs in requirements tables**, links with these short names are included throughout the report when referring to a requirements document. When you click a short name link in a report, the requirements document associated with that document ID opens.

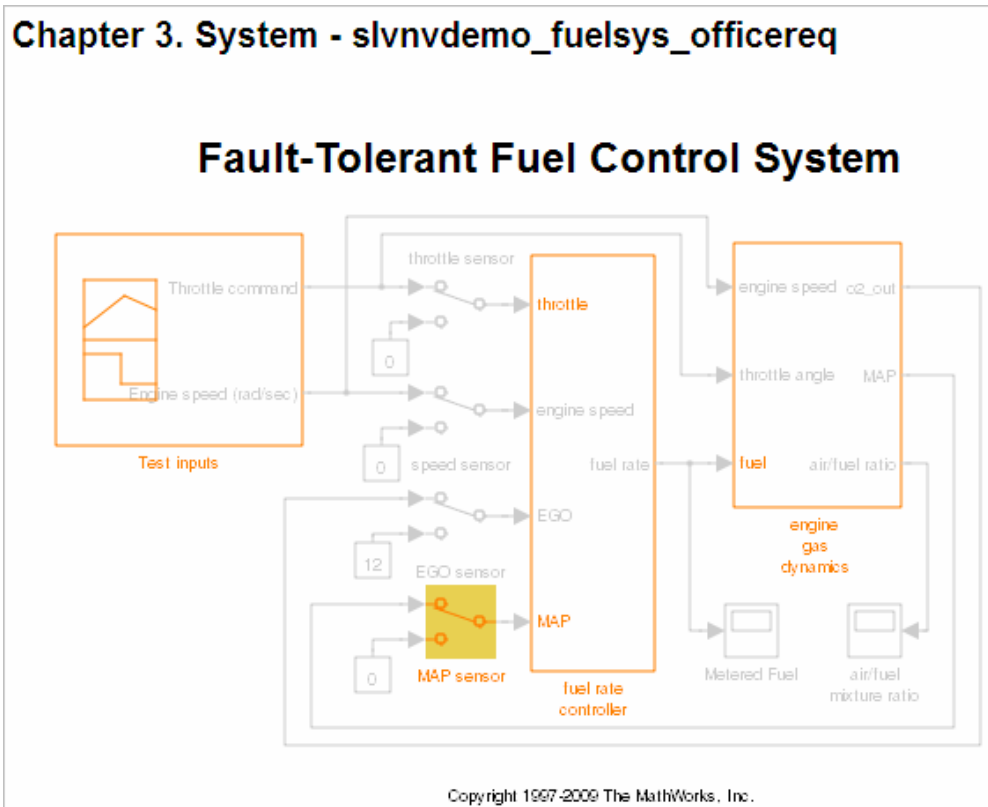
When your requirements documents have long path names that can clutter the report, select the **User document IDs in requirements tables** option. This option is disabled by default, as you can see in the examples in this section.

- **Document paths stored in the model** — Click this link to open the requirements document in its native application.
- **Last modified** — The date the requirements document was last modified.
- **# links** — The total number of links to a requirements document.

System

Each **System** section includes:

- An image of the model or model object. The objects with requirements are highlighted.



- A list of requirements associated with the model or model object. In this example, click the target document name to open the requirements document associated with the slvndemo_fuelsys_officereq model.

Table 3.1. slvndemo_fuelsys_officereq Requirements

Link#	Description	Target (document name and location ID)
1	Label: Design Description Microsoft Word Document	slvndemo_FuelSys_DesignDescription.docx

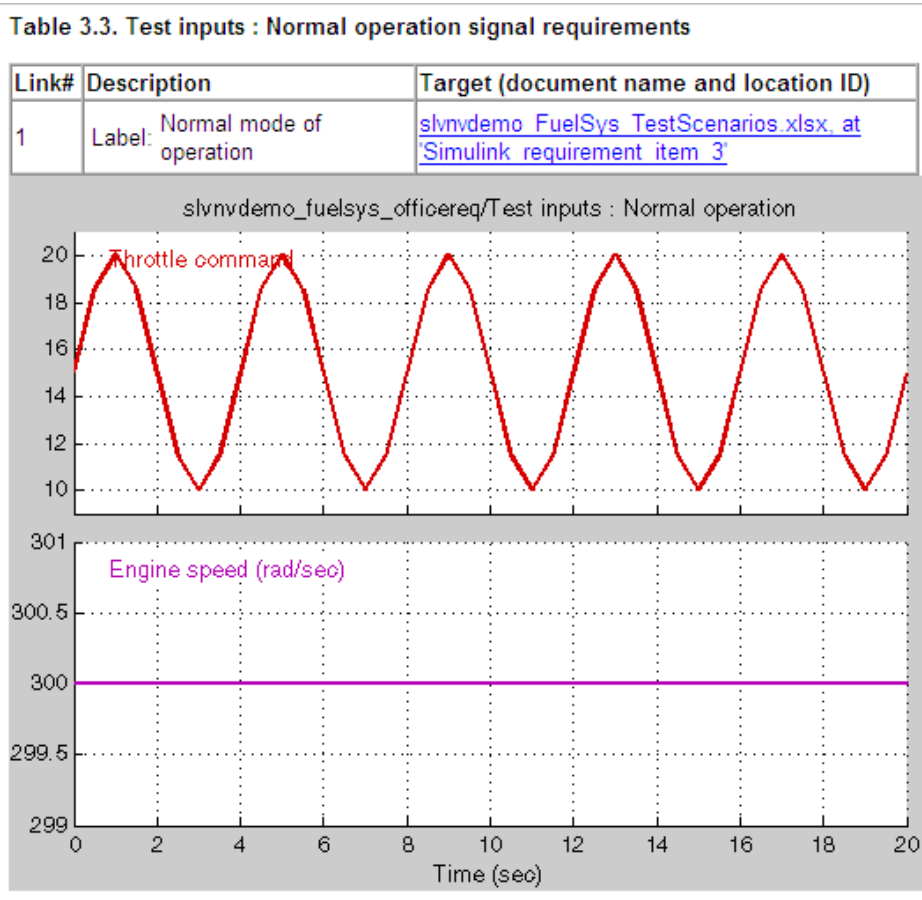
- A list of blocks in the top-level model that have requirements. In this example, only the MAP sensor block has a requirement at the top level. Click the link next to **Target:** to open the requirements document associated with the MAP sensor block.

Table 3.2. Blocks in "slvndemo_fuelsys_officereq" that have requirements

Name	Requirements
MAP sensor	<p data-bbox="538 369 1172 543">Link#1 label: "The System detects the Manifold Absolute Pressure sensor short to ground or open circuit by monitoring when the signal is below a calibratable threshold. The failure is detected within 100mSec of the occurrence and the MAP sensor reading is reverted to an estimated throttle position based on engine speed and throttle position."</p> <p data-bbox="461 552 1172 604">Target: fuelsys_req_docs\slvndemo_FuelSys_TestScenarios.xlsx, at 'Simulink requirement item 2'</p>

The preceding table does not include these blocks in the top-level model because:

- The fuel rate controller and engine gas dynamics subsystems are in dedicated chapters of the report.
- The report lists Signal Builder blocks separately, in this example, in Table 3.3.
- A list of requirements associated with each signal group in any Signal Builder block, and a graphic of that signal group. In this example, the Test inputs Signal Builder block in the top-level model has one signal group that has a requirement link. Click the link under **Target (document name and location ID)** to open the requirements document associated with this signal group in the Test inputs block.



Chart

Each **Chart** section reports on requirements in Stateflow charts, and includes:

- A graphic of the Stateflow chart that identifies each state.
- A list of elements that have requirements.

To navigate to the requirements document associated with a chart element, click the link next to **Target**.

Table 10.1. Stateflow objects with requirements

Name	Requirements
warmup	Link#1 label: "During a calibratable warm up period the oxygen sensor correction shall be disabled." Target: fuelsys_req_docs\slvndemo_FuelSys_RequirementsSpecification.docx, at 'Simulink requirement item 3'
[speed==0 & press < zero_thresh]/...	Link#1 label: "Speed sensor failure detection" Target: fuelsys_req_docs\slvndemo_FuelSys_DesignDescription.docx, at 'Simulink requirement item 6'
Rich_Mixture	Link#1 label: "Enriched mixture usage" Target: fuelsys_req_docs\slvndemo_FuelSys_DesignDescription.docx, at 'Simulink requirement item 4'
Warmup	Link#1 label: "During a calibratable warm up period the oxygen sensor correction shall be disabled." Target: fuelsys_req_docs\slvndemo_FuelSys_RequirementsSpecification.docx, at 'Simulink requirement item 3'

Report for Requirements in Model Blocks

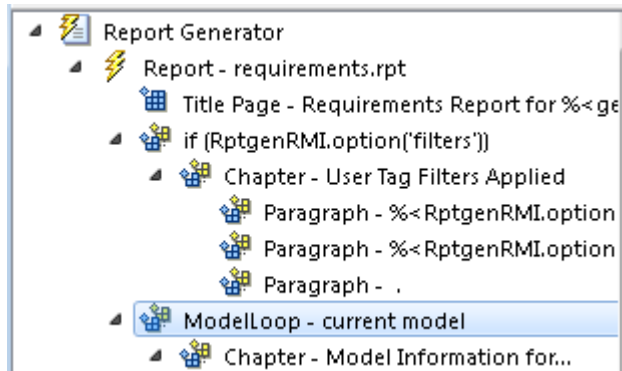
If your model contains Model blocks that reference external models, the default report does not include information about requirements in the referenced models. To generate a report that includes requirements information for referenced models, you must have a license for the Simulink Report Generator software. The report includes the same information and graphics for referenced models as it does for the top-level model.

If you have a Simulink Report Generator license, before generating a requirements report, take the following steps:

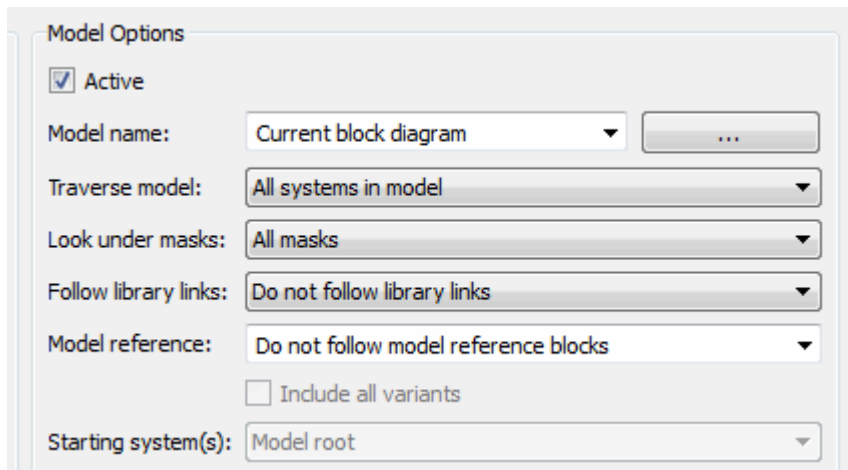
- 1 Open the model for which you want to create a requirements report. This workflow uses the example model `slvndemo_fuelsys_officereq`.
- 2 To open the template for the default requirements report, at the MATLAB command prompt, enter:


```
setedit requirements
```

- 3 In the Simulink Report Generator software window, in the far-left pane, click the **Model Loop** component.



- 4 On the far-right pane, locate the **Model reference** field. If you cannot see the drop-down arrow for that field, expand the pane.



- 5 In the **Model reference** field drop-down list, select **Follow all model reference blocks**.
- 6 To generate a requirements report for the open model that includes information about referenced models, click the **Report** icon .

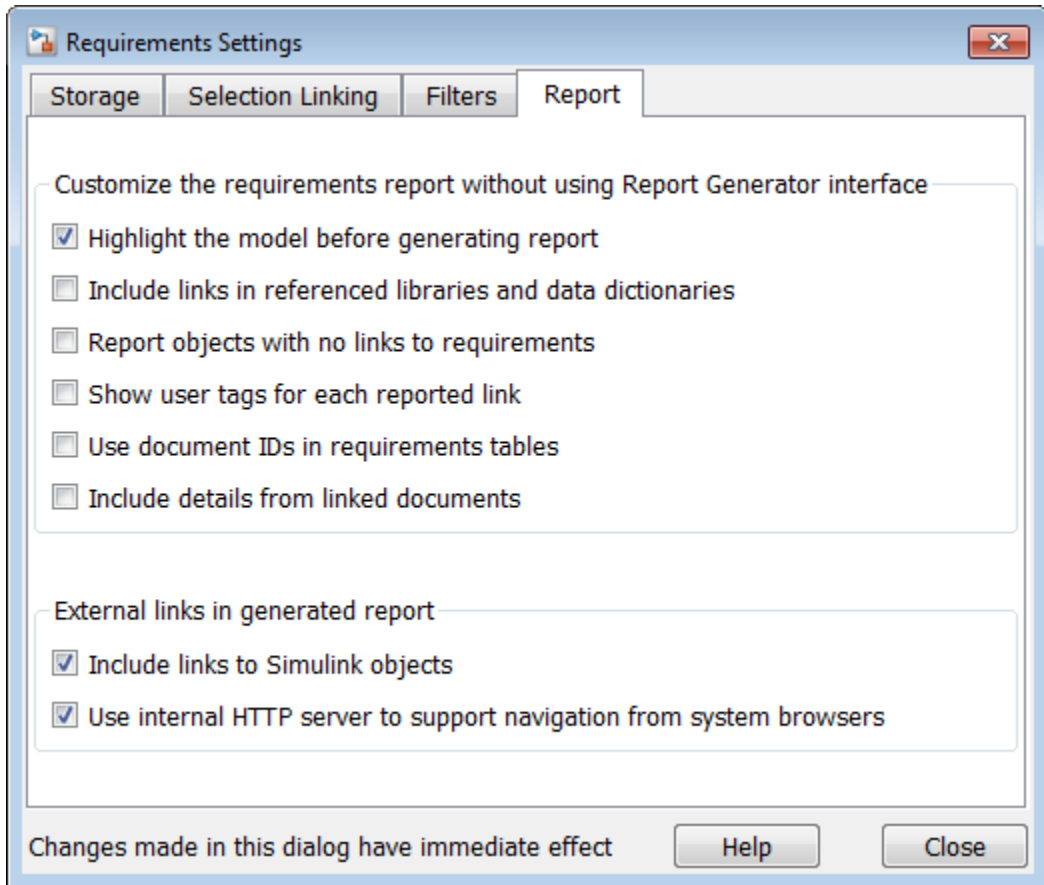
Customize Requirements Report

The Requirements Management Interface (RMI) uses the Simulink Report Generator software to generate the requirements report. You can customize the requirements report using the RMI or the Simulink Report Generator software:

- “Customize Requirements Report Using the RMI Settings” on page 11-25
- “Customize Requirements Report Using Simulink Report Generator” on page 11-29

Customize Requirements Report Using the RMI Settings

There are several options for customizing a requirements report using the Requirements Settings dialog box.



On the **Report** tab, select options that specify the contents that you want in the report.

Requirements Settings Report Option	Description
Highlight the model before generating report	Enables highlighting of Simulink objects with requirements in the report graphics.
Include links in referenced libraries and data dictionaries	Includes requirements links in referenced libraries in the generated report.
Report objects with no links to requirements	Includes lists of model objects that have no requirements.

Requirements Settings Report Option	Description
Show user tags for each reported link	Lists the user tags, if any, for each reported link.
Use document IDs in requirements tables	Uses a document ID, if available, instead of a path name in the tables of the requirements report. This capability prevents long path names to requirements documents from cluttering the report tables.
Include details from linked documents	Includes additional content from linked requirements. The following requirements documents are supported: <ul style="list-style-type: none"> • Microsoft Word • Microsoft Excel • IBM Rational DOORS
Include links to Simulink objects	Includes links from the report to objects in Simulink.
Use internal HTTP server to support navigation from system browsers	Specifies use of internal MATLAB HTTP server for navigation from generated report to documents and model objects. By selecting this setting, this navigation is available from system browsers as long as the MATLAB internal HTTP server is active on your local host. To start the internal HTTP server, at the MATLAB command prompt, type <code>rmi('httpLink')</code> .

To see how these options affect the content of the report:

- 1 Open the `slvndemo_fuelsys_officereq` model:

```
slvndemo_fuelsys_officereq
```
- 2 In the model window, select **Analysis > Requirements > Settings**.
- 3 In the Requirements Settings dialog box, click the **Report** tab.
- 4 For this example, select **Highlight the model before generating report**.

When you select this option, before generating the report, the graphics of the model that are included in the report are highlighted so that you can easily see which objects have requirements.

- 5 To close the Requirements Settings dialog box, click **Close**.
- 6 Generate a requirements report by selecting **Analysis > Requirements > Generate Report**.

The requirements report opens in a browser window so that you can review the content of the report.

- 7 If you do not want to overwrite the current report when you regenerate the requirements report, rename the HTML file, for example, `slvnvdemo_fuelsys_officereq_requirements_old.html`.

The default report file name is `model_name_requirements.html`.

- 8 In the model window, select **Analysis > Requirements > Settings**.
- 9 On the **Report** tab, select:
 - **Show user tags for each reported link** — The report lists the user tags (if any) associated with each requirement.
 - **Include details from linked documents** — The report includes additional details for requirements in the following types of requirements documents.

Requirements Document Format	Includes in the Report
Microsoft Word	Full text of the paragraph or subsection of the requirement, including tables.
Microsoft Excel	If the target requirement is a group of cells, the report includes all those cells as a table. If the target requirement is one cell, the report includes that cell and all the cells in that row to the right of the target cell.

Requirements Document Format	Includes in the Report
IBMRationalDOORS	<p>By default, the report includes:</p> <ul style="list-style-type: none"> • DOORS Object Heading • DOORS Object Text • All other attributes except Created Thru, attributes with empty string values, and system attributes that are false. <p>Use the <code>RptgenRMI.doorsAttribs</code> function to include or exclude specific attributes or groups of attributes.</p>

- 10** Close the Requirements Settings dialog box.
- 11** Generate a new requirements report by selecting **Analysis > Requirements > Generate Report**.
- 12** Compare this new report to the report that you renamed in step 7:
 - User tags associated with requirements links are included.
 - Details from the requirement content are included as specified in step 9.
- 13** When you are done reviewing the report, close the report and the model.

To see an example of including details in the requirements report, enter the following command at the MATLAB command prompt:

```
slvndemo_powerwindow_report
```

Customize Requirements Report Using Simulink Report Generator

If you have a license for the Simulink Report Generator software, you can further modify the default requirements report.

At the MATLAB command prompt, enter the following command:

```
setedit requirements
```

The Report Explorer GUI opens the requirements report template that the RMI uses when generating a requirements report. The report template contains Simulink Report Generator components that define the structure of the requirements report.

If you click a component in the middle pane, the options that you can specify for that component appear in the right-hand pane. For detailed information about using a particular component to customize your report, click **Help** at the bottom of the right-hand pane.

In addition to the standard report components, Simulink Report Generator provides components specific to the RMI in the Requirements Management Interface category.

Simulink Report Generator Component	Report Information
Missing Requirements Block Loop	Applies all child components to blocks that have no requirements
Missing Requirements System Loop	Applies all child components to systems that have no requirements
Requirements Block Loop	Applies all child components to blocks that have requirements
Requirements Documents Table	Inserts a table that lists requirements documents
Requirements Signal Loop	Applies all child components to signal groups with requirements
Requirements Summary Table	Inserts a property table that lists requirements information for blocks with associated requirements
Requirements System Loop	Applies all child components to systems with requirements
Requirements Table	Inserts a table that lists system and subsystem requirements
Data Dictionary Traceability Table	Inserts a table that links data dictionary information to requirements
MATLAB Code Traceability Table	Inserts a table that links MATLAB code to requirements
Simulink Test Suite Traceability Table	Inserts a table that links a Simulink test suite to requirements

To customize the requirements report, you can:

- Add or delete components.

- Move components up or down in the report hierarchy.
- Customize components to specify how the report presents certain information.

For more information, see the Simulink Report Generator documentation.

Generate Requirements Reports Using Simulink

When you have a model open in Simulink, the Model Editor provides two options for creating requirements reports:

System Design Description Report

The System Design Description report describes a system design represented by the current Simulink model.

You can use the System Design Description report to:

- Review a system design without having the model open.
- Generate summary and detailed descriptions of the design.
- Assess compliance with design requirements.
- Archive the system design in a format independent of the modeling environment.
- Build a customized version of the report using the Simulink Report Generator software.

To generate a System Design Description report that includes requirements information:

- 1 Open the model for which you want to create a report.
- 2 Select **File > Reports > System Design Description**.
- 3 In the Design Description dialog box, select **Requirements traceability**.
- 4 Select any other options that you want for this report.
- 5 Click **Generate**.

As the software is generating the report, the status appears in the MATLAB command window.

The report name is the model name, followed by a numeral, followed by the extension that reflects the document type (.pdf, .html, etc.).

If your model has linked requirements, the report includes a chapter, **Requirements Traceability**, that includes:

- Lists of model objects that have requirements with hyperlinks to display the objects
- Images of each subsystem, highlighting model objects with requirements

Design Requirements Report

In the Simulink Editor, the menu option **File > Reports > System Requirements** creates a requirements report, as described in “Create Default Requirements Report” on page 11-15. This menu option is equivalent to **Analysis > Requirements > Generate Report**.

To specify options for the report, select **Analysis > Requirements > Settings**. Before generating the report, on the **Report** tab, set the options that you want. For detailed information about these options, see “Customize Requirements Report” on page 11-25.

Create Requirements Traceability Report for A Project

To create a report for requirements traceability data in a project:

- 1 Open your project.
- 2 At the MATLAB command prompt, enter the following:

```
rmi('projectreport')
```

The MATLAB Web browser opens, showing the traceability report for the project.

This top-level HTML report contains a separate section for Simulink model files, MATLAB code files, and other files included in the project. For each individual file with one or more associated requirements links, a separate HTML report, or sub-report, shows the requirements traceability data for that file. The top-level report contains links to each sub-report.

If you have a MATLAB file with requirements traceability links that is not part of a project, you can create a separate report for the MATLAB file using the `rmi('report', matlabfilepath)` command. For more information, see `rmi`.

Validate Requirements Links

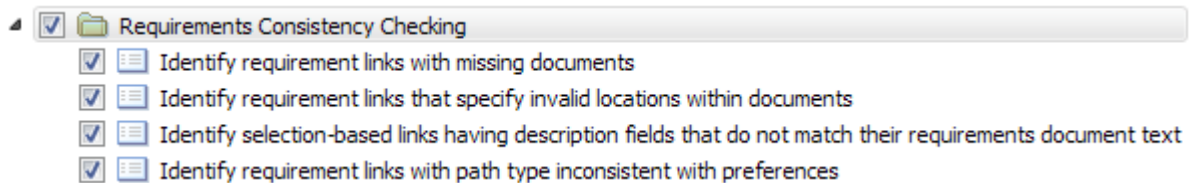
Validate Requirements Links in a Model

Check Requirements Links with the Model Advisor

To make sure that every requirements link in your Simulink model has a valid target in a requirements document, run the Model Advisor Requirements consistency checks:

- 1 Open the example model:
`slvndemo_fuelsys_officereq`
- 2 Open the Model Advisor to run a consistency check by selecting **Analysis > Requirements > Check Consistency**.

In the **Requirements Consistency Checking** category, all the checks are selected. For this tutorial, keep all the checks selected.



These checks identify the following problems with your model requirements.

Consistency Check	Problem Identified
Identify requirement links with missing documents	The Model Advisor cannot find the requirements document. This might indicate a problem with the path to the requirements document.

Consistency Check	Problem Identified
Identify requirement links that specify invalid locations within documents	The Model Advisor cannot find the designated location for the requirement. This check is implemented for: <ul style="list-style-type: none">• Microsoft Word documents• Microsoft Excel documents• IBM Rational DOORS documents• Simulink objects
Identify selection-based links having description fields that do not match their requirements document text	The Description field for the link does not match the requirements document text. When you create selection-based links, the Requirements Management Interface (RMI) saves the selected text in the link Description field. This check is implemented for: <ul style="list-style-type: none">• Microsoft Word documents• Microsoft Excel documents• IBM Rational DOORS documents• Simulink objects

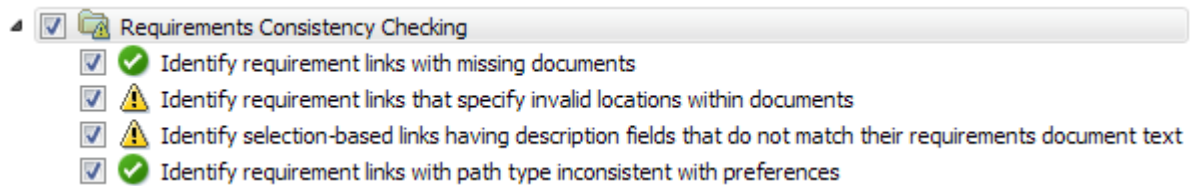
Consistency Check	Problem Identified
<p>Identify requirement links with path type inconsistent with preferences</p>	<p>The path to the requirements document does not match the Document file reference field in the Requirements Settings dialog box Selection Linking tab. This might indicate a problem with the path to the requirements document.</p> <p>On Linux systems, this check is named Identify requirement links with absolute path type. The check reports a warning for each requirements links that uses an absolute path.</p> <hr/> <p>Note For information about how the RMI resolves the path to the requirements document, see “Document Path Storage” on page 11-48.</p>

The Model Advisor checks to see if any applications that have link targets are running:

- If your model has links to Microsoft Word or Microsoft Excel documents, the consistency check requires that you close all instances of those applications. If you have one of these applications open, it displays a warning and does not continue the checks. The consistency checks must verify up-to-date stored copies of the requirements documents.
 - If your model has links to DOORS requirements, you must be logged in to the DOORS software. Your DOORS database must include the module that contains the target requirements.
- 3** For this tutorial, make sure that you close both Microsoft Word and Microsoft Excel.
- 4** Click **Run Selected Checks**.

After the check is complete:

- The green circles with the check mark indicate that two checks passed.
- The yellow triangles with the exclamation point indicate that two checks generated warnings.



The right-hand pane shows that two checks passed and two checks had warnings. The Report box includes a link to the HTML report.

Keep the Model Advisor open. The next section describes how to interpret and fix the inconsistent links.

Note To step through an example that uses the Model Advisor to check requirements links in an IBM Rational DOORS database, run the Managing Requirements for Fault-Tolerant Fuel Control System (IBM Rational DOORS) example in the MATLAB command prompt.

Fix Invalid Requirements Links Detected by the Model Advisor

In “Check Requirements Links with the Model Advisor” on page 11-34, three requirements consistency checks generate warnings in the `slvndemo_fuelsys_officereq` model.

Resolve Warning: Identify requirement links that specify invalid locations within documents

To fix the warning about attempting to link to an invalid location in a requirements document:

- 1 In the Model Advisor, select **Identify requirement links that specify invalid locations within documents** to display the description of the warning.

Inconsistencies:

The following requirements link to invalid locations within their documents. The specified location (e.g., bookmark, line number, anchor) within the requirements document could not be found. To resolve this issue, edit each requirement and specify a valid location within its requirements document.

Block	Requirements
slvnvdemo_fuelsys_officereq/fuel_rate_controller/Sensor_correction_and_Fault_Redundancy/Terminator1	This section will be deleted

This check identifies a link that specifies a location that does not exist in the Microsoft Word requirements document, `slvnvdemo_FuelSys_DesignDescription.docx`. The link originates in the Terminator1 block. In this example, the target location in the requirements document was deleted after the requirement was created.

- 2 Get more information about this link:
 - a To navigate to the Terminator1 block, under **Block**, click the hyperlink.
 - b To open the “Outgoing Links Editor” on page 10-7 for this link, under **Requirements**, click the hyperlink.
- 3 To fix the problem from the Outgoing Links dialog, do one of the following:
 - In the **Location** field, specify a valid location in the requirements document.
 - Delete the requirements link by selecting the link and clicking **Delete**.
- 4 In the Model Advisor, select the **Requirements Consistency Checking** category of checks.
- 5 Click **Run Selected Checks** again, and verify that the warning no longer occurs.

Resolve Warning: Identify selection-based links having description fields that do not match their requirements document text

To fix the warnings about the **Description** field not matching the requirements document text:

- 1 In the Model Advisor, click **Identify selection-based links having description fields that do not match their requirements document text** to display the description of the warning.

Unable to check:

- Failed to locate item @**Simulink_requirement_item_7** in **fuelsys_req_docs\slvndemo_FuelSys_DesignDescription.docx**

Inconsistencies:

The following selection-based links have descriptions that differ from their corresponding selections in the requirements documents. If this reflects a change in the requirements document, click **Update** to replace the current description in the selection-based link with the text from the requirements document (the external description).

Block	Current description	External description	
slvndemo fuelsys_officereq/Test inputs	Normal mode of operation	The simulation is run with a throttle input that ramps from 10 to 20 degrees over a period of two seconds, then back to 10 degrees over the next two seconds. This cycle repeats continuously while the engine is held at a constant speed.	Update
slvndemo fuelsys_officereq/fuel rate controller/Sensor correction and Fault Redundancy/MAP Estimate	Manifold pressure failure	Manifold pressure failure mode	Update

The first message indicated that the model contains a link to a bookmark named **Simulink_requirement_item_7** in the requirements document that does not exist.

In addition, this check identified the following mismatching text between the requirements blocks and the requirements document:

- The **Description** field in the Test inputs Signal Builder block link is **Normal mode of operation**. The requirement text is **The simulation is run with a**

throttle input that ramps from 10 to 20 degrees over a period of two seconds, then back to 10 degrees over the next two seconds. This cycle repeats continuously while the engine is held at a constant speed.

- The **Description** field in the MAP Estimate block link is **Manifold pressure failure**. The requirement text in `slvncdemo_FuelSys_DesignDescription.docx` is **Manifold pressure failure mode**.
- 2 Get more information about this link:
 - a To navigate to a block, under **Block**, click the hyperlink.
 - b To open the “Outgoing Links Editor” on page 10-7 for this link, under **Current Description**, click the hyperlink.
 - 3 Fix this problem in one of two ways:
 - In the Model Advisor, click **Update**. This action automatically updates the **Description** field for that link so that it matches the requirement.
 - In the Link Editor, manually edit the link from the block so that the **Description** field matches the selected requirements text.
 - 4 In the Model Advisor, select the **Requirements Consistency Checking** category of checks.
 - 5 Click **Run Selected Checks** again, and verify that the warning no longer occurs.

Validate Requirements Links in a Requirements Document

Check Links in a Requirements Document

To check the links in a requirements document:

- 1 At the MATLAB command prompt, enter

```
rmi('checkdoc', docName)
```

`docName` is a character vector that represents one of the following:

- Module ID for a DOORS requirements document
- Full path name for a Microsoft Word requirements document
- Full path name for a Microsoft Excel requirements document

The `rmi` function creates and displays an HTML report that lists all requirements links in the document.

The report highlights invalid links in red. For each invalid link, the report includes brief details about the problem and a hyperlink to the invalid link in the requirements document. The report groups together links that have the same problem.

- 2 Double-click the hyperlink under **Document content** to open the requirements document at the invalid link.

The navigation controls for the invalid link has a different appearance than the navigation controls for the valid links.

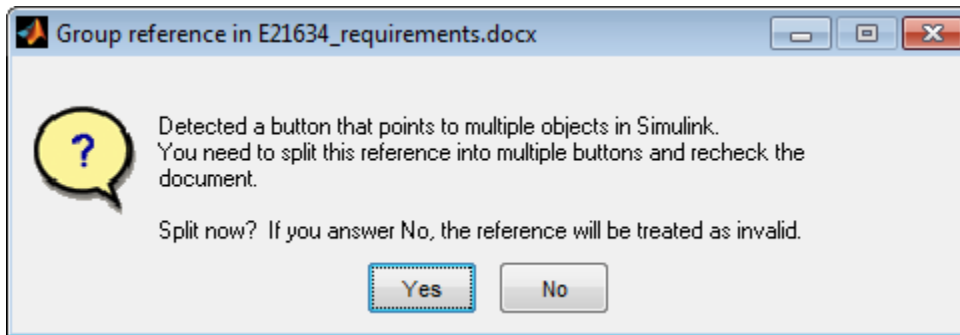
- 3 When there are invalid links in your requirements document, you have the following options:

If you want to...	Do the following...
Fix the invalid links	Follow the instructions in "Fix Invalid Links in a Requirements Document" on page 11-42.
Keep the changes to the navigation controls without fixing the invalid links	Save the requirements document.
Ignore the invalid links	Close the requirements document without saving it.

When Multiple Objects Have Links to the Same Requirement

When you link multiple objects to the same requirement, only one navigation object is inserted into the requirements document. When you double-click that navigation object, all of the linked model objects are highlighted.

If you check the requirements document using the `'checkdoc'` option of the `rmi` function and the check detects a navigation object that points to multiple objects, the check stops and displays the following dialog box.



You have two options:

- If you click **Yes**, or you close this dialog box, the RMI creates additional navigation objects, one for each model object that links to that requirement. The document check continues, but the RMI does not recheck that navigation; the report only shows one link for that requirement. To rerun the check so that all requirements are checked, at the top of the report, click **Refresh**.
- If you click **No**, the document check continues, and the report identifies that navigation object as a broken link.

Fix Invalid Links in a Requirements Document

Using the report that the `rmi` function creates, you may be able to fix the invalid links in your requirements document.

In the following example, `rmi` cannot locate the model specified in two links.

References with Unresolved Models - 1 unique problem in 2 links

Document content	Target model
Transmission Requirements	sf_car_doors.mdl
Engine Torque Requirements	

To fix invalid links:

- 1 In the report, under **Document content**, click the hyperlink associated with the invalid requirement link.

The requirements document opens with the requirement text highlighted.

- 2 In the requirements document, depending on the document format, take these steps:
 - In DOORS:
 - a Select the navigation control for an invalid link.
 - b Select **MATLAB > Select item**.
 - In Microsoft Word, double-click the navigation control.

A dialog box opens that allows you to fix, reset, or ignore all the invalid links with a given problem.

- 3 Click one of the following options.

To...	Click...
Navigate to and select a new target model or new target objects for these broken links.	Fix all
Reset the navigation controls for these invalid links to their original state, the state before you checked the requirements document.	Reset all
Make no changes to the requirements document. Any modifications rmi made to the navigation controls remain in the requirements document.	Cancel

- 4 Save the requirements document to preserve the changes made by the rmi function.

Validation of Requirements Links

Requirements links in a model can become outdated when requirements change over time. Similarly, links in requirements documents may become invalid when your Simulink model changes, for example, when the model, or objects in the model, are renamed, moved, or deleted. The Simulink Requirements software provides tools that allow you to detect and resolve these problems in the model or in the requirements document.

- “When to Check Links in a Requirements Document” on page 11-44
- “How the rmi Function Checks a Requirements Document” on page 11-44

When to Check Links in a Requirements Document

When you enable **Modify destination for bidirectional linking** and create a link between a requirement and a Simulink model object, the RMI software inserts a navigation control into your requirements document. These links may become invalid if your model changes.

To check these links, the 'checkDoc' option of the rmi function reviews a requirements document to verify that all the navigation controls represent valid links to model objects. The checkDoc command can check the following types of requirements documents:

- Microsoft Word
- Microsoft Excel
- IBM Rational DOORS


The rmi function only checks requirements documents that contain navigation controls; to check links in your Simulink model, see “Validate Requirements Links in a Model” on page 11-34.

Note For more information about inserting navigation controls in requirements documents, see:

- “Insert Navigation Objects in Microsoft Office Requirements Documents” on page 6-13
 - “Insert Navigation Objects into IBM Rational DOORS Requirements” on page 7-8
-

How the rmi Function Checks a Requirements Document

rmi performs the following actions:

- Locates all links to Simulink objects in the specified requirements document.
- Checks each link to verify that the target object is present in a Simulink model. If the target object is present, rmi checks that the link label matches the target object.
- Modifies the navigation controls in the requirements document to identify any detected problems. This allows you to see invalid links at a glance:
 - Valid link: 

- Invalid link: 

Delete Requirements Links from Simulink Objects

Delete a Single Link from a Simulink Object

If you have an obsolete link to a requirement, delete it from the model object.

To delete a single link to a requirement from a Simulink model object:

- 1 Right-click a model object and select **Requirements > Open Outgoing Links dialog**.
- 2 In the top-most pane of the Link Editor, select the link that you want to delete.
- 3 Click **Delete**.
- 4 Click **Apply** or **OK** to complete the deletion.

Delete All Links from a Simulink Object

To delete all links to requirements from a Simulink model object:

- 1 Right-click the model object and select **Requirements > Delete All Outgoing Links**
- 2 Click **OK** to confirm the deletion.

This action deletes all requirements at the top level of the object. For example, if you delete requirements for a subsystem, this action does not delete any requirements for objects inside the subsystem; it only deletes requirements for the subsystem itself. To delete requirements for child objects inside a subsystem, Model block, or Stateflow chart, you must navigate to each child object and perform these steps for each object from which you want to delete requirements.

Delete All Links from Multiple Simulink Objects

To delete all requirements links from a group of Simulink model objects in the same model diagram or Stateflow chart:

- 1 Select the model objects whose requirements links you want to delete.
- 2 Right-click one of the objects and select **Requirements > Delete All Outgoing Links**.

- 3 Click **OK** to confirm the deletion.

This action deletes all requirements at the top level of each object. It does not delete requirements for child objects inside subsystems, Model blocks, or Stateflow charts.

Document Path Storage

When you create a requirements link, the RMI stores the location of the requirements document with the link. If you use selection-based linking or browse to select a requirements document, the RMI stores the document location as specified by the **Document file reference** option on the Requirements Settings dialog box, **Selection Linking** tab. The available settings are:

- Absolute path
- Path relative to current folder
- Path relative to model folder
- Filename only (on MATLAB path)

You can also manually enter an absolute or relative path for the document location. A relative path can be a partial path or no path at all, but you must specify the file name of the requirements document. If you use a relative path, the document is not constrained to a single location in the file system. With a relative path, the RMI resolves the exact location of the requirements document in this order:

- 1 The software attempts to resolve the path relative to the current MATLAB folder.
- 2 When there is no path specification and the document is not in the current folder, the software uses the MATLAB search path to locate the file.
- 3 If the RMI cannot locate the document relative to the current folder or the MATLAB search path, the RMI resolves the path relative to the model file folder.

The following examples illustrate the procedure for locating a requirements document.

Relative (Partial) Path Example

Current MATLAB folder	C:\work\scratch
Model file	C:\work\models\controllers\pid.mdl
Document link	..\reqs\pid.html
Documents searched for (in order)	C:\work\reqs\pid.html C:\work\models\reqs\pid.html

Relative (No) Path Example

Current MATLAB folder	C:\work\scratch
Model file	C:\work\models\controllers\pid.mdl
Requirements document	pid.html
Documents searched for (in order)	C:\work\scratch\pid.html <MATLAB path dir>\pid.html C:\work\models\controllers\pid.html

Absolute Path Example

Current MATLAB folder	C:\work\scratch
Model file	C:\work\models\controllers\pid.mdl
Requirements document	C:\work\reqs\pid.html
Documents searched for	C:\work\reqs\pid.html

Requirements Management Interface

Verification and Validation

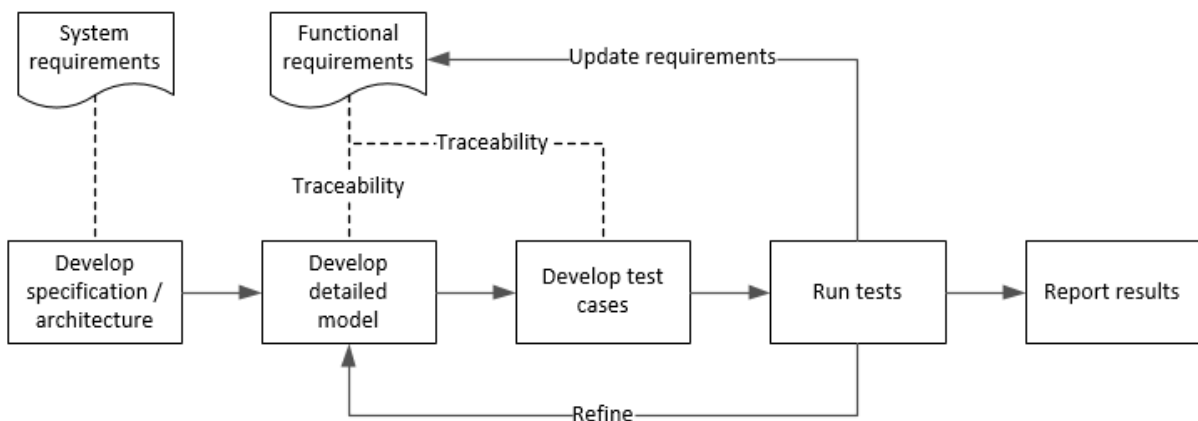
- “Test Model Against Requirements and Report Results” on page 13-2
- “Analyze a Model for Standards Compliance and Design Errors” on page 13-8
- “Perform Functional Testing and Analyze Test Coverage” on page 13-11
- “Analyze Code and Test Software-in-the-Loop” on page 13-15

Test Model Against Requirements and Report Results

Requirements - Test Traceability Overview

Traceability between requirements and test cases helps you interpret test results and see the extent to which your requirements are verified. You can link a requirement to elements that help verify it, such as test cases in the Test Manager, verify statements in a Test Sequence block, or Model Verification blocks in a model. When you run tests, a pass/fail summary appears in your requirements set.

This example demonstrates a common requirements-based testing workflow for a cruise control model. You start with a requirements set, a model, and a test case. You add traceability between the tests and the safety requirements. You run the test, summarize the verification status, and report the results.




In this example, you conduct a simple test of two requirements in the set:

- That the cruise control system transitions to disengaged from engaged when a braking event has occurred
- That the cruise control system transitions to disengaged from engaged when the current vehicle speed is outside the range of 20 mph to 90 mph.

Display the Requirements and Test Case

- 1 Create a copy of the project in a working folder. The project contains data, documents, models, and tests. Enter:

```
path = fullfile(matlabroot,'toolbox','shared','examples',...  
'verification','src','cruise')  
run(fullfile(path,'slVerificationCruiseStart'))
```

- 2 In the project models folder, open the `simulinkCruiseAddReqExample.slx` model.
- 3 Display the requirements. Click the  icon in the lower-right corner of the model canvas, and select **Requirements**. The requirements appear below the model canvas.
- 4 Expand the requirements information to include verification and implementation status. Right-click a requirement and select **Verification Status** and **Implementation Status**.

The screenshot shows the Simulink Requirements Browser interface. On the left, a Simulink block diagram for 'simulinkCruiseAddReqExample' is displayed. It features a central 'Compute target speed' block with two state machines. Inputs include 'CruiseOnOff' (boolean), 'Brake' (boolean), 'Speed' (single), 'CoastSetSw' (boolean), and 'AccelResSw' (boolean). Outputs are 'engaged' (boolean) and 'tspeed' (single). On the right, the 'Requirement: A 1.2' details are shown, including a description: 'Set Speed/Decelerate Button' and a rationale: 'The controller shall have an input button to: set the target speed to the current vehicle speed when the cruise control is **not engaged (active)** decelerate (reduce) the target speed when the cruise control is **engaged (active)**'. Below the diagram is a table of requirements.

Index	ID	Summary	Implemented	Verified
simulinkCruiseChartReqs				
1	Architecture	Architecture		
1.1	A 1.1	Enable Disable Switch		
1.2	A 1.2	Set Speed / Decelerate Button		
1.3	A 1.3	Resume Speed / Accelerate Butt...		
1.4	A 1.4	Engaged Output		
1.5	A 1.5	Target Speed Output		
1.6	A 1.6	Vehicle Speed Input		
1.7	A 1.7	Vehicle Brake Input		
2	Functional Requirements	Functional Requirements		
3	Safety Requirements	Safety Requirements		
3.1	S 3.1	Vehicle braking disengages syst...		
3.2	S 3.2	System engagement speed limit...		
3.3	S 3.3	Target speed limitations		
3.4	S 3.4	Speed outside limits disengages ...		

- 5 Open the Simulink Test file `slReqTests.mldatx` from the `tests` folder. The test file opens in the Test Manager.

Link Requirements to Tests

Link the requirements to the test case.

- 1 In the Requirements Browser, select requirement S 3.1.
- 2 In the Test Manager, expand the test file and select the **Safety Tests** test case. Expand the **Requirements** section.
- 3 In the **Requirements** section, select **Add > Link to Selected Requirement**.

The requirements browser displays the verification-type link.

3	Safety Requirements	Safety Requirements
3.1	S 3.1	Vehicle braking disengages system
3.2	S 3.2	System engagement speed limitations
3.3	S 3.3	Target speed limitations
3.4	S 3.4	Speed outside limits disengages system

Verified by: [Safety Tests](#)

Comments

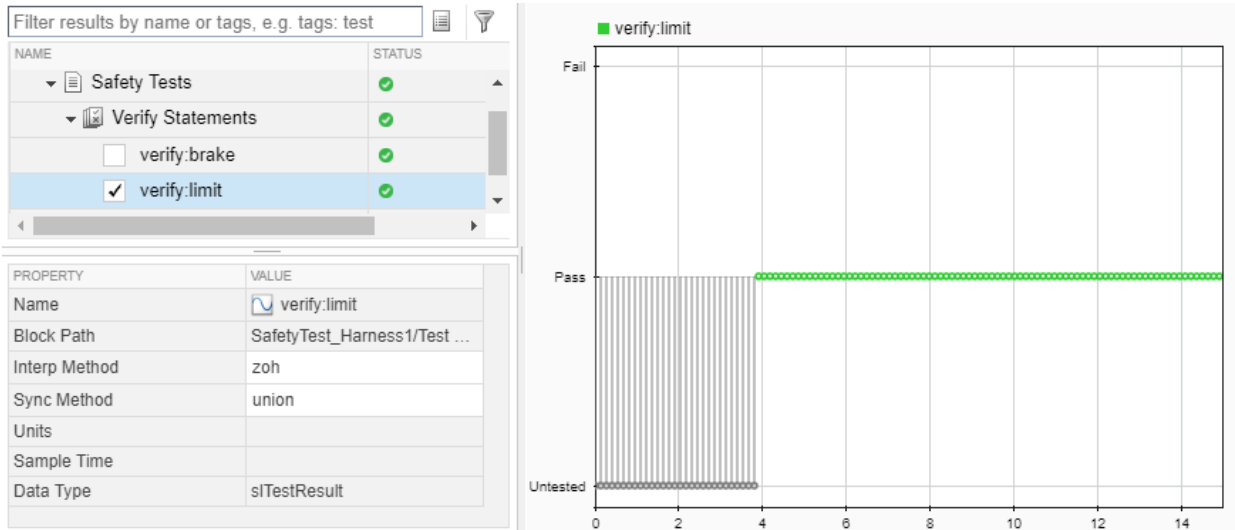
- 4 Also add a link for item S 3.4.

Run the Test

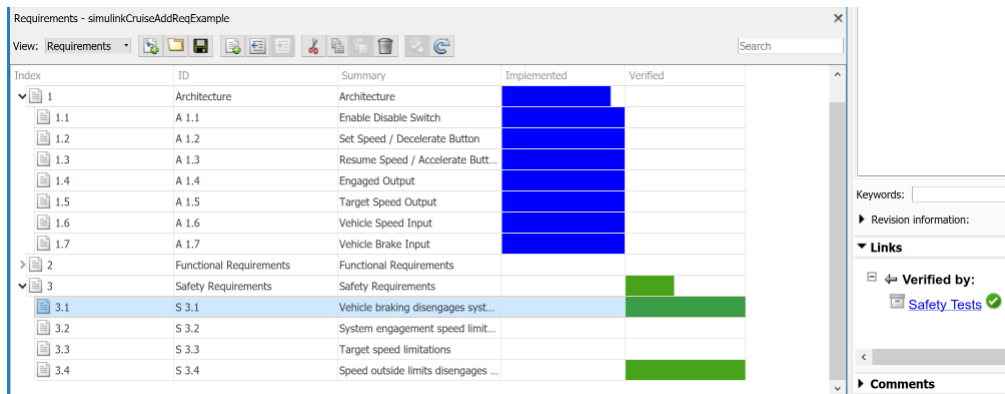
- 1 The test case uses a test harness `SafetyTest_Harness1`. In the test harness, a test sequence sets the input conditions and checks the model behavior:
 - The `BrakeTest` sequence engages the cruise control, then applies the brake. It includes the `verify` statement


```
verify(engaged == false,...
      'verify:brake',...
      'system must disengage when brake applied')
```
 - The `LimitTest` sequence engages the cruise control, then ramps up the vehicle speed until it exceeds the upper limit. It includes the `verify` statement.


```
verify(engaged == false,...
      'verify:limit',...
      'system must disengage when limit exceeded')
```
- 2 Run the test case. In the Test Manager toolstrip, click **Run**.
- 3 When the test finishes, expand the **Verify Statements** results. The Test Manager results show that both assessments pass, and the plot shows the detailed results of each `verify` statement.



- 4 In the Requirements Browser, right-click a requirement and select **Refresh Verification Status** to show the passing test results for each requirement.



Report the Results

- 1 Create a report using a custom Microsoft Word template.
 - a From the Test Manager results, right-click the test case name. Select **Create Report**.

- b** In the Create Test Result Report dialog box, set the options:
 - Title — SafetyTest
 - Results for — All Tests
 - File Format — DOCX
 - For the other options, keep the default selections.
 - c** For the **Template File**, select the ReportTemplate.dotx file in the **documents** project folder.
 - d** Enter a file name and select a location for the report.
 - e** Click **Create**.
- 2** Review the report.
- a** In the **Test Case Requirements** section, click the link to trace to the requirements document.
 - b** The **Verify Result** section contains details of the two assessments in the test, and links to the simulation output.

Name	Data Type	Units	Sample Time	Interp	Sync	Link to Plot
✔ Test Sequence/.../Verify:verify(engaged == false)	siTestResult			zoh	union	Link
✔ Test Sequence/.../VerifyHigh:verify(engaged == false)	siTestResult			zoh	union	Link

See Also

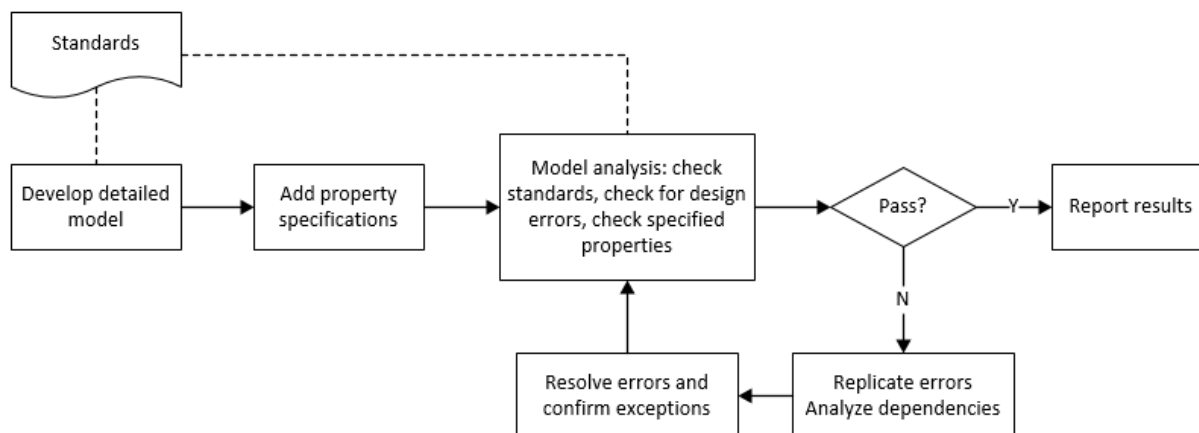
Related Examples

- “Link to Requirements” (Simulink Test)
- “Validate Requirements Links in a Model” on page 11-34
- “Customize Requirements Traceability Report for Model” on page 11-15

Analyze a Model for Standards Compliance and Design Errors

Standards and Analysis Overview

During model development, check and analyze your model to increase confidence in its quality. Check your model against standards such as MAAB style guidelines and high-integrity system design guidelines such as DO-178 and ISO 26262. Analyze your model for errors, dead logic, and conditions that violate required properties. Using the analysis results, update your model and document exceptions. Report the results using customizable templates.



Check Model for Style Guideline Violations and Design Errors

This example shows how to use the Model Advisor to check a cruise control model for MathWorks® Automotive Advisory Board (MAAB) style guideline violations and design errors. Select checks and run the analysis on the model. Iteratively debug issues using the Model Advisor and rerun checks to verify that it is in compliance. After passing your selected checks, report results.

Check Model for MAAB Style Guideline Violations

In Model Advisor, you can check that your model complies with MAAB modeling guidelines.

- 1 Create a copy of the project in a working folder. On the command line, enter

```
path = fullfile(matlabroot,'toolbox','shared','examples',...  
'verification','src','cruise')  
run(fullfile(path,'slVerificationCruiseStart'))
```

- 2 Open the model. On the command line, enter

```
open_system simulinkCruiseErrorAndStandardsExample
```

- 3 In the model window, select **Analysis > Model Advisor > Model Advisor**.

- 4 Click OK to choose `simulinkCruiseErrorAndStandardsExample` from the System Hierarchy.

- 5 Check your model for MAAB style guideline violations using Simulink Check.

- a In the left pane, in the **By Product > Simulink Check > Modeling Standards > MathWorks Automotive Advisory Board Checks** folder, select:

- **Check for indexing in blocks**
- **Check for prohibited blocks in discrete controllers**
- **Check model diagnostic parameters**

- b Right-click the **MathWorks Automotive Advisory Board Checks** node, and then select **Run Selected Checks**.

- c Click **Check model diagnostic parameters** to review the configuration parameter settings that violate MAAB style guidelines.

- d In the right pane, click the parameter links to update the values in the Configuration Parameters dialog box.

- e To verify that your model passes, rerun the check. Repeat steps c and d, if necessary, to reach compliance.

- f To generate a results report of the Simulink Check checks, select the **MathWorks Automotive Advisory Board Checks** node, and then, in the right pane click **Generate Report...**

Check Model for Design Errors

While in Model Advisor, you can also check your model for hidden design errors using Simulink Design Verifier.

- 1** In the left pane, in the **By Product > Simulink Design Verifier** folder, select **Design Error Detection**.
- 2** In the right pane, click **Run Selected Checks**.
- 3** After the analysis is complete, expand the **Design Error Detection** folder, then select checks to review warnings or errors.
- 4** In the right pane, click **Simulink Design Verifier Results Summary**. The dialog box provides tools to help you diagnose errors and warnings in your model.
 - a** Review the results on the model. Click **Highlight analysis results on model**. Click the **Compute target speed** subsystem, outlined in red. The Simulink Design Verifier Results Inspector window provides derived ranges that can help you understand the source of an error by identifying the possible signal values.
 - b** Review the harness model. The Simulink Design Verifier Results Inspector window displays information that an overflow error occurred. To see the test cases that demonstrate the errors, click **View test case**.
 - c** Review the analysis report. In the Simulink Design Verifier Results Inspector window, click **Back to summary**. To see a detailed analysis report, click **HTML** or **PDF**.

See Also

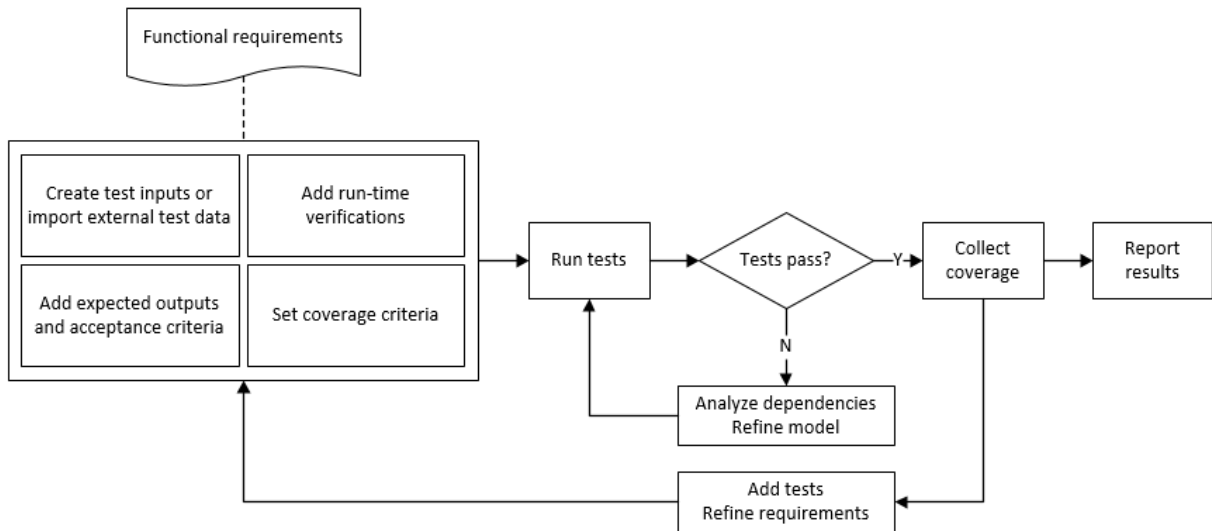
Related Examples

- “Check Model Compliance by Using the Model Advisor” (Simulink Check)
- “Collect Model Metrics Using the Model Advisor” (Simulink Check)
- “Run a Design Error Detection Analysis” (Simulink Design Verifier)
- “Prove Properties in a Model” (Simulink Design Verifier)

Perform Functional Testing and Analyze Test Coverage

Functional testing begins with building test cases based on requirements. These tests can cover key aspects of your design and verify that individual model components meet requirements. Test cases include inputs, expected outputs, and acceptance criteria.

By collecting individual test cases within test suites, you can run functional tests systematically. To check for regression, add baseline criteria to the test cases and test the model iteratively. Coverage measurement reflects the extent to which these tests have fully exercised the model. Coverage measurement also helps you to add tests and requirements to meet coverage targets.



Incrementally Increase Test Coverage Using Test Case Generation

This example shows a functional testing-based testing workflow for a cruise control model. You start with a model that has tests linked to an external requirements document, analyze the model for coverage in Simulink Coverage, incrementally increase coverage with Simulink Design Verifier, and report the results.

Explore the Test Harness and the Model

- 1 Create a copy of the project in a working folder. At the command line, enter:

```
path = fullfile(matlabroot, 'toolbox', 'shared', 'examples', ...  
'verification', 'src', 'cruise')  
run(fullfile(path, 'slVerificationCruiseStart'))
```

- 2 Open the model and the test harness. At the command line, enter:

```
open_system('simulinkCruiseAddReqExample')  
sltest.harness.open('simulinkCruiseAddReqExample', 'SafetyTest_Harness1')
```

- 3 Load the test suite from “Test Model Against Requirements and Report Results” (Simulink Test). At the command line, enter:

```
sltest.testmanager.load('slReqTests.mldatx')  
sltest.testmanager.view
```

- 4 Open the test sequence block. The sequence tests:

- That the system disengages when the brake pedal is pressed
- That the system disengages when the speed exceeds a limit

Some test sequence steps are linked to a requirements document `simulinkCruiseChartReqs.docx`.

Measure Model Coverage

- 1 In the Test Manager, enable coverage collection for the test case.
 - a Open the Test Manager. In the Simulink menu, click **Analysis > Test Manager**.
 - b In the **Test Browser**, click the `slReqTests` test file.
 - c Expand **Coverage Settings**.
 - d Under **Coverage to Collect**, select **Record coverage for referenced models**.

You specify a coverage filter to use for coverage analysis by using the **Coverage filter filename** field. The default setting honors the model configuration parameter settings. Leaving the **Coverage filter filename** field empty attaches no coverage filter.



- e Under **Coverage Metrics**, select **Decision**, **Condition**, and **MCDC**.

▼ COVERAGE SETTINGS*

▼ COVERAGE TO COLLECT

Record coverage for system under test

Record coverage for referenced models






Coverage filter filename:  

COVERAGE METRICS

<input checked="" type="checkbox"/> Decision	<input checked="" type="checkbox"/> Condition
<input checked="" type="checkbox"/> MCDC	<input type="checkbox"/> Lookup Table
<input type="checkbox"/> Signal Range	<input type="checkbox"/> Signal Size
<input type="checkbox"/> Simulink Design Verifier	<input type="checkbox"/> Saturation on integer overflow
<input type="checkbox"/> Relational Boundary	

- 2 Run the test. On the Test Manager toolstrip, click **Run**.
- 3 When the test finishes, in the Test Manager, navigate to the test case. The aggregated coverage results show that the example model achieves 50% decision coverage, 41% condition coverage, and 25% MCDC coverage.

▼ AGGREGATED COVERAGE RESULTS ?

ANALYZED MODEL	REPORT CO...	DECISION	CONDITION	MCDC
 simulinkCruiseAddReqExample	 31	50% 	41% 	25% 

+ Add Tests for Missing Coverage Export

Generate Tests to Increase Model Coverage

- 1 Use Simulink Design Verifier to generate additional tests to increase model coverage. Select the test case in the **Results and Artifacts** and open the aggregated coverage results section.
- 2 Select the test results from the previous section and then click **Add Tests for Missing Coverage**.

The **Add Tests for Missing Coverage** options open.

- 3 Under **Harness**, choose Create a new harness.
- 4 Click **OK** to add tests to the test suite using Simulink Design Verifier.
- 5 Run the updated test suite. On the Test Manager toolstrip, click **Run**. The test results include coverage for the combined test case inputs, achieving increased model coverage.

See Also

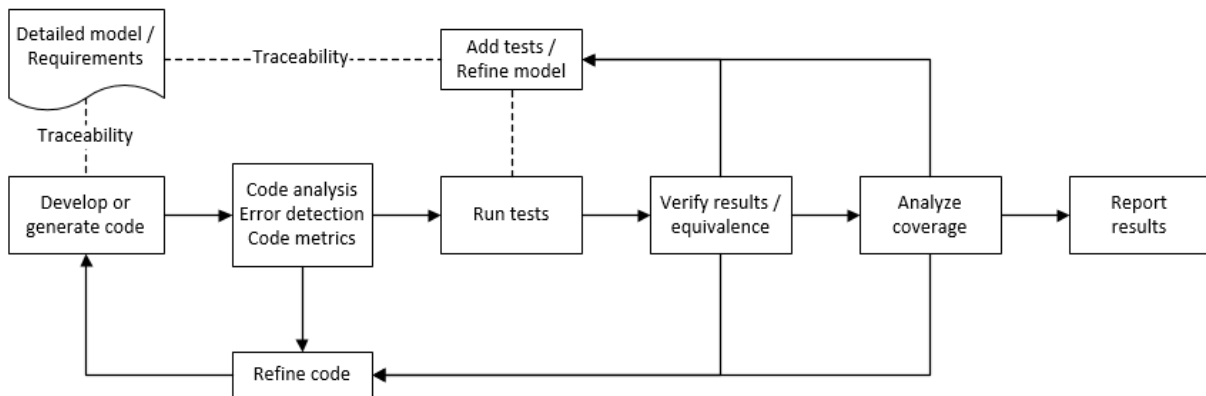
Related Examples

- “Link to Requirements” (Simulink Test)
- “Assess Model Simulation Using verify Statements” (Simulink Test)
- “Compare Model Output To Baseline Data” (Simulink Test)
- “Generate Test Cases for Model Decision Coverage” (Simulink Design Verifier)
- “Increase Test Coverage for a Model” (Simulink Test)

Analyze Code and Test Software-in-the-Loop

Code Analysis and Testing Software-in-the-Loop Overview

Analyze code to detect errors, check standards compliance, and evaluate key metrics such as length and cyclomatic complexity. Typically for handwritten code, you check for run-time errors with static code analysis and run test cases that evaluate the code against requirements and evaluate code coverage. Based on the results, refine the code and add tests. For generated code, demonstrate that code execution produces equivalent results to the model by using the same test cases and baseline results. Compare the code coverage to the model coverage. Based on test results, add tests and modify the model to regenerate code.



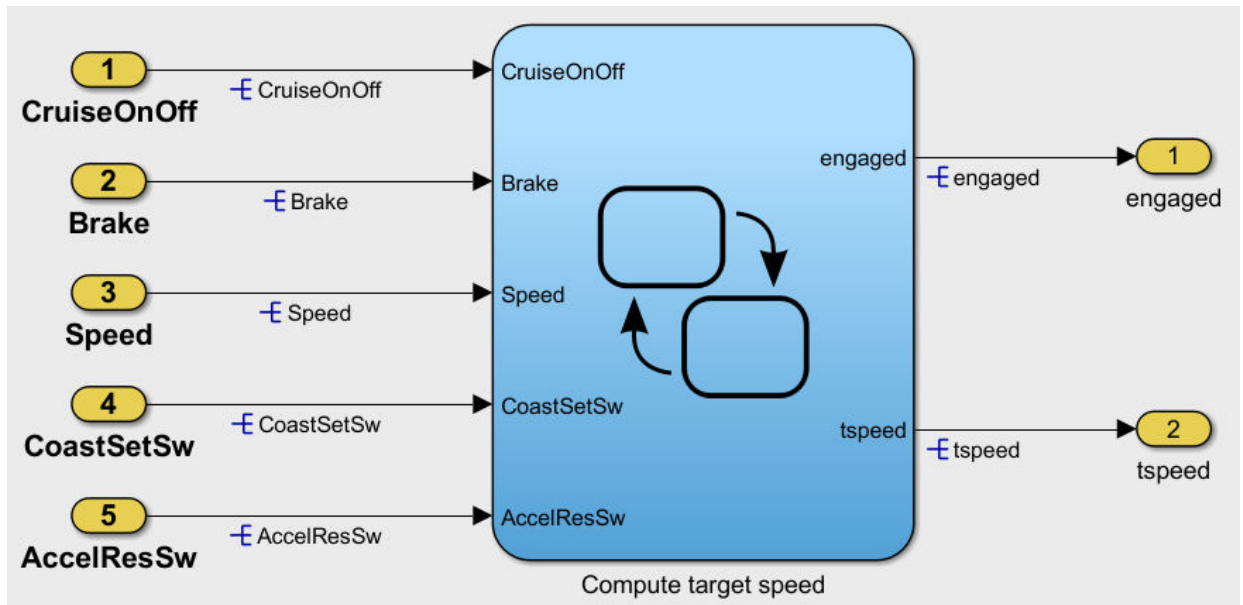
Analyze Code for Defects, Metrics, and MISRA C:2012

This workflow describes how to check if your model produces MISRA® C:2012 compliant code and how to check your generated code for code metrics, code defects, and MISRA compliance. To produce more MISRA compliant code from your model, you use the code generation and Model Advisor. To check whether the code is MISRA compliant, you use the Polyspace MISRA C:2012 checker and report generation capabilities. For this example, you use the model `simulinkCruiseErrorAndStandardsExample`. To open the model:

- 1 Open the project.

```
path = fullfile(matlabroot,'toolbox','shared','examples',...
'verification','src','cruise')
run(fullfile(path,'slVerificationCruiseStart'))
```

- From the project, open the model `simulinkCruiseErrorAndStandardsExample`.

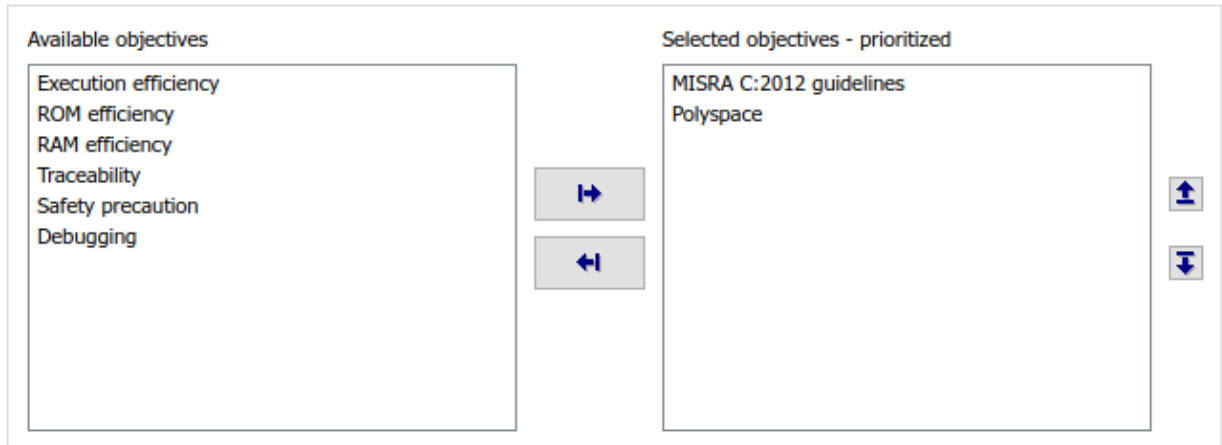


Run Code Generator Checks

Before you generate code from your model, there are steps that you can take to generate code more compliant with MISRA C and more compatible with Polyspace. This example shows how to use the Code Generation Advisor to check your model before generating code.

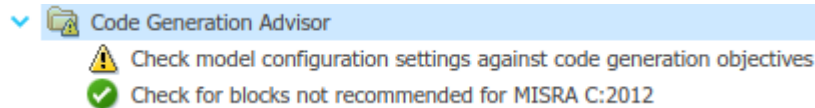
- Right-click `Compute target speed` and select **C/C++ > Code Generation Advisor**.
- Select the Code Generation Advisor folder. Add the Polyspace objective. The MISRA C:2012 guidelines objective is already selected.

Code Generation Objectives (System target file: ert.tlc)



3 Click **Run Selected Checks**.

The Code Generation Advisor checks whether there are any blocks or configuration settings that are not recommended for MISRA C:2012 compliance and Polyspace code analysis. For this mode, the check for incompatible blocks passes, but there are some configuration settings that are incompatible with MISRA compliance and Polyspace checking.



4 Click on check that was not passed. Accept the parameter changes by selecting **Modify Parameters**.

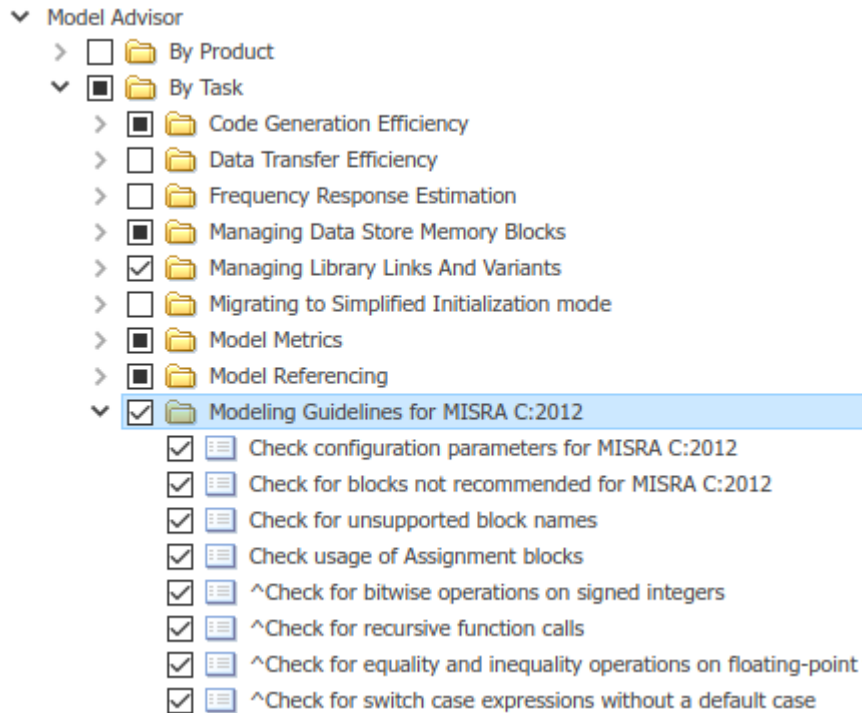
5 Rerun the check by selecting **Run This Check**.

Run Model Advisor Checks

Before you generate code from your model, there are steps you can take to generate code more compliant with MISRA C and more compatible with Polyspace. This example shows you how to use the Model Advisor to check your model further before generating code.

For more checking before generating code, you can also run the Modeling Guidelines for MISRA C:2012.

- 1 At the bottom of the Code Generation Advisor window, select **Model Advisor**.
- 2 Under the **By Task** folder, select the **Modeling Guidelines for MISRA C:2012** advisor checks.



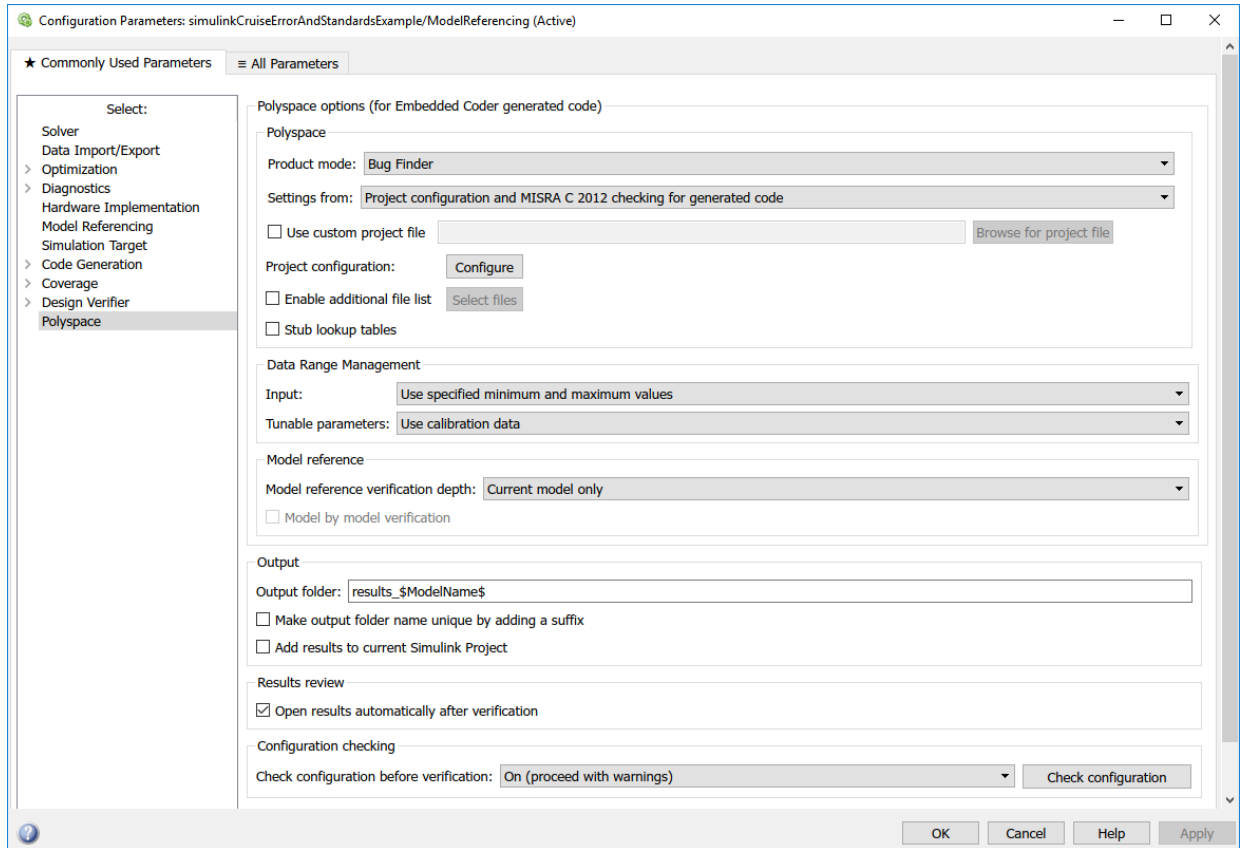
- 3 Click **Run Selected Checks** and review the results.
- 4 If any of the tasks fail, make the suggested modifications and rerun the checks until the MISRA modeling guidelines pass.

Generate and Analyze Code

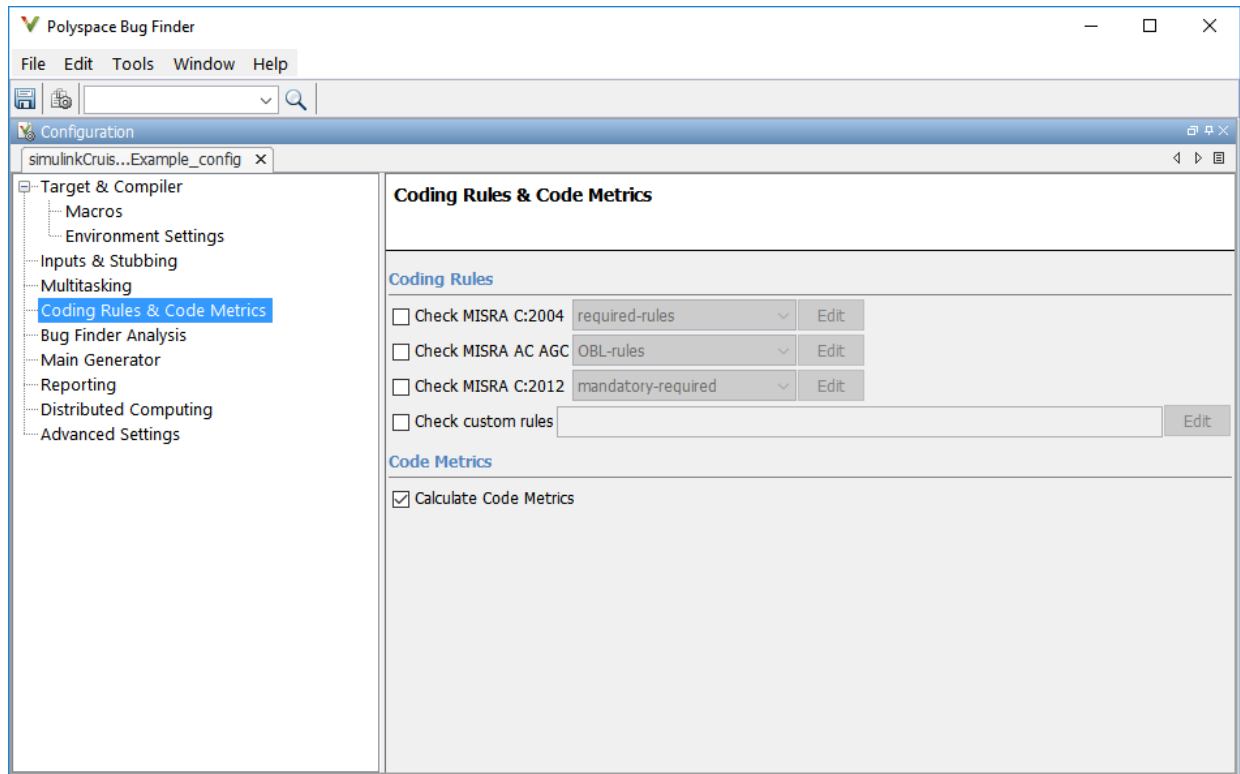
After you have done the model compliance checking, you can now generate code. With Polyspace, you can check your code for compliance with MISRA C:2012 and generate reports to demonstrate compliance with MISRA C:2012.

- 1 In the Simulink editor, right-click Compute target speed and select **C/C++ > Build This Subsystem**.
- 2 Use the default settings for the tunable parameters and select **Build**.

- 3 After the code is generated, right-click Compute target speed and select **Polyspace > Options**.



- 4 Click the **Configure** (Polyspace Bug Finder) button. This option allows you to choose more advanced Polyspace analysis options in the Polyspace configuration window.



- 5 On the same pane, select **Calculate Code Metrics**. This option turns on code metric calculations for your generated code.
- 6 Save and close the Polyspace configuration window.
- 7 From your model, right-click Compute target speed and select **Polyspace > Verify Code Generated For > Selected Subsystem**.

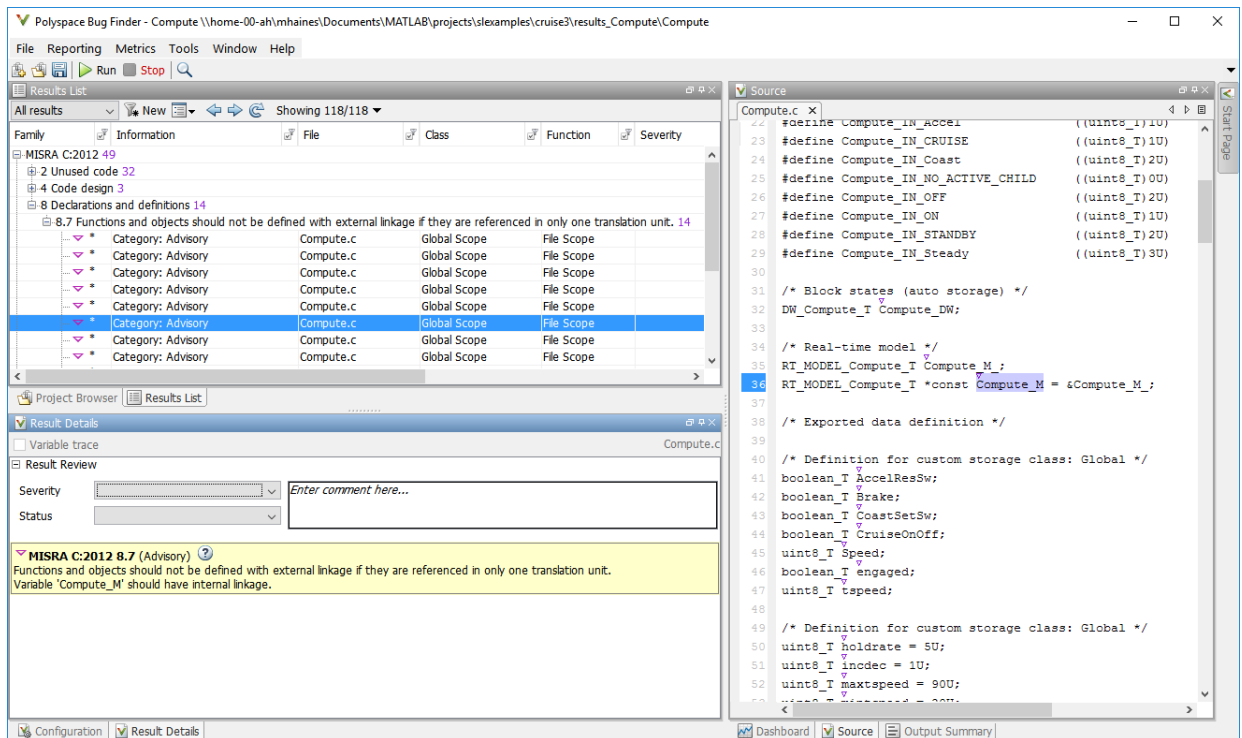
Polyspace Bug Finder analyzes the generated code for a subset of MISRA checks and defect checks. You can see the progress of the analysis in the MATLAB Command Window. Once the analysis is finished, the Polyspace environment opens.

Review Results

After you run a Polyspace analysis of your generated code, the Polyspace environment shows you the results of the static code analysis. There are 50 MISRA C:2012 coding rule violations in your generated code.

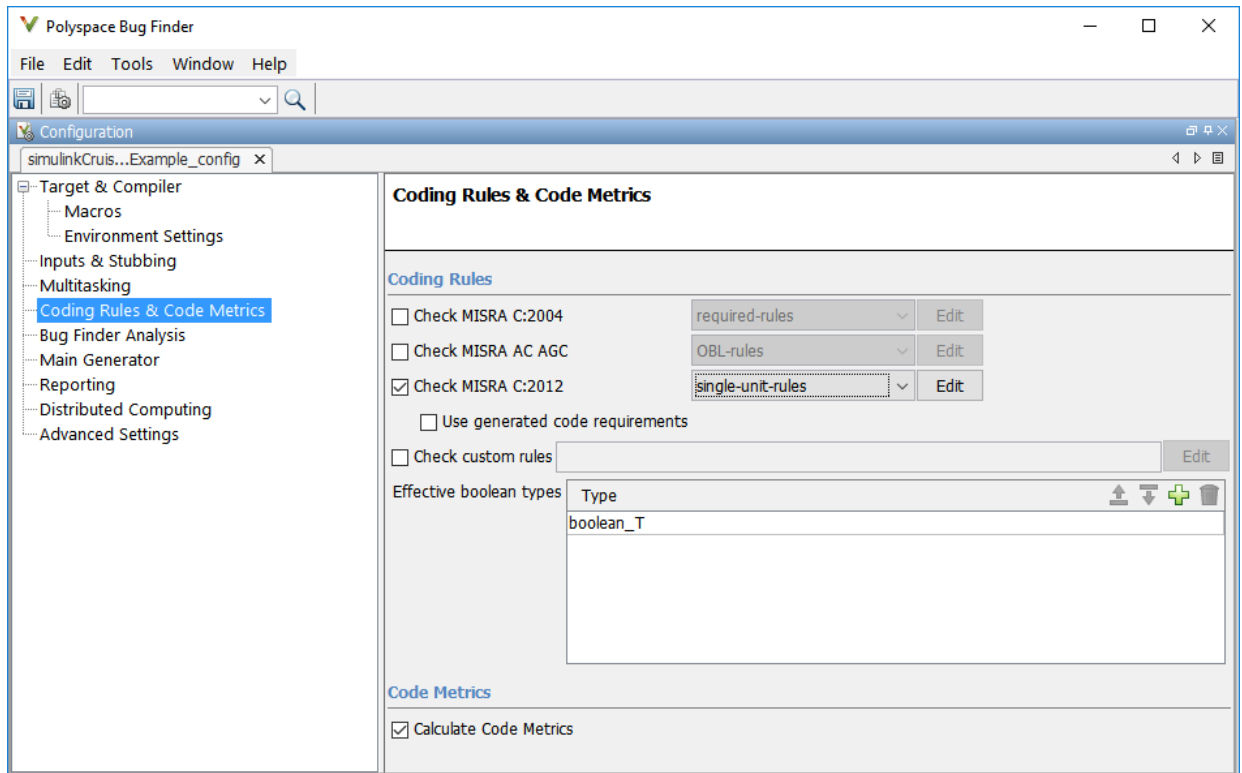
- 1 Expand the tree for rule 8.7 and click through the different results.

Rule 8.7 states that functions and objects should not be global if the function or object is local. As you click through the 8.7 violations, you can see that these results refer to variables that other components also use, such as `CruiseOnOff`. You can annotate your code or your model to justify every result. But, because this model is a unit in a larger program, you can also change the configuration of the analysis to check only a subset of MISRA rules.



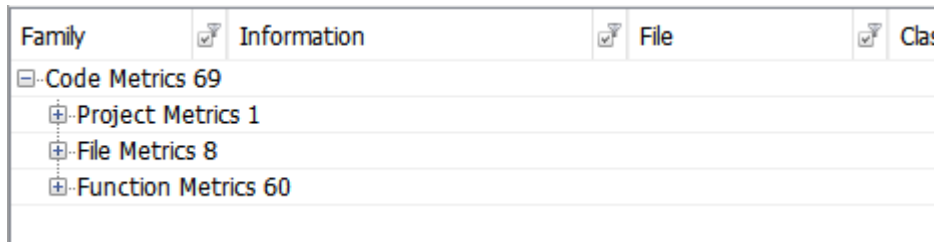
- 2 In your model, right-click Compute target speed and select **Polyspace > Options**.
- 3 Set the **Settings from** (Polyspace Bug Finder) option to **Project configuration**. This option allows you to choose a subset of MISRA rules in the Polyspace configuration.
- 4 Click the **Configure** button.
- 5 In the Polyspace Configuration window, on the **Coding Rules & Code Metrics** pane, select the check box **Check MISRA C:2012** and from the drop-down list, select

single-unit-rules. Now, Polyspace checks only the MISRA C:2012 rules that are applicable to a single unit.



- 6 Save and close the Polyspace configuration window.
- 7 Rerun the analysis with the new configuration.

When the Polyspace environment reopens, there are no MISRA results, only code metric results. The rules Polyspace showed previously were found because the model was analyzed by itself. When you limited the rules Polyspace checked to the single-unit subset, no violations were found.



When this model is integrated with its parent model, you can add the rest of the MISRA C:2012 rules.

Generate Report

To demonstrate compliance with MISRA C:2012 and report on your generated code metrics, you must export your results. This section shows you how to generate a report after the analysis. If you want to generate a report every time you run an analysis, see [Generate report](#).

- 1 If they are not open already, open your results in the Polyspace environment.
- 2 From the toolbar, select **Reporting > Run Report**.
- 3 Select **BugFinderSummary** as your report type.
- 4 Click **Run Report**.

The report is saved in the same folder as your results.

- 5 To open the report, select **Reporting > Open Report**.

See Also

Related Examples

- “Run Polyspace Analysis on Code Generated with Embedded Coder” (Polyspace Bug Finder)
- “Test Two Simulations for Equivalence” (Simulink Test)
- “Export Test Results and Generate Reports” (Simulink Test)

