# Simulink® Requirements™

## Reference

# MATLAB®&SIMULINK®

MathWorks®

# 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.
3 Apple Hill Drive
Natick, MA 01760-2098

**Revision History**

# Contents

# Functions — Alphabetical List

# slreq.clear

Clear requirements and links from memory

## Syntax

```
slreq.clear()
```

## Description

`slreq.clear()` clears all requirements and links loaded in memory and closes the Requirements Editor, discarding all unsaved changes.

## See Also

`slreq.LinkSet` | `slreq.ReqSet`

**Introduced in R2018a**

# slreq.convertAnnotation

Convert annotations to requirement objects

## Syntax

```
myReq = slreq.convertAnnotation(myAnnotation,myDestination)
myReq = slreq.convertAnnotation(myAnnotation,myDestination,Name,
Value)
```

## Description

`myReq = slreq.convertAnnotation(myAnnotation,myDestination)` converts a
Simulink® or a Stateflow® annotation `myAnnotation` into a requirement `myReq` and
stores it in a destination entity `myDestination`.

`myReq = slreq.convertAnnotation(myAnnotation,myDestination,Name,
Value)` converts a Simulink or a Stateflow annotation `myAnnotation` into a requirement
`myReq` and stores it in a destination entity `myDestination` using additional options
specified by one or more `Name, Value` pair arguments.

## Examples

### Convert Simulink Annotation to Requirement

```
% Find all annotations in a Simulink model
allAnnotations = find_system('controller_Model', 'FindAll', ...
'on', 'type', 'annotation');

% Create a new requirements set
newReqSet = slreq.new('myNewReqSet');

% Convert one annotation into a requirement newReq
% and add it to newReqSet
newReq = slreq.convertAnnotation(allAnnotations(1), ...
newReqSet);
```

# Input Arguments

**`myAnnotation` — Simulink or Stateflow annotation**
`Simulink.Annotation` object

The annotation to be converted, specified as a `Simulink.Annotation` object.

**`myDestination` — Converted annotation destination entity**
`slreq.Requirement` object | `slreq.ReqSet` object

The destination entity for the converted annotation, specified either as an `slreq.Requirement` or as an `slreq.ReqSet` object.

## Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'CreateLinks', true`

**`CreateLinks` — Option to create links**
`true` (default) | `false`

Option to create links when converting annotations, specified as a Boolean value.

**`KeepAnnotation` — Option to retain annotation**
`false` (default) | `true`

Option to retain the annotation after conversion, specified as a Boolean value.

**`IgnoreCallback` — Option to force annotation conversion**
`false` (default) | `true`

Option to specify annotation conversion even if a callback function is specified in the annotation, specified as a Boolean value.

**`ShowMarkup` — Option to display requirements markup**
`true` (default) | `false`

Option to display the Requirement markup after annotation conversion, specified as a Boolean value.

# Output Arguments

**myReq — Requirement**
slreq.Requirement object

The converted annotation, returned as an slreq.Requirement object.

# See Also
slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# slreq.createLink

Create traceable links

## Syntax

```
myLink = slreq.createLink(src, dest)
```

## Description

`myLink = slreq.createLink(src, dest)` creates an `slreq.Link` object `myLink` that serves as a link between the source artifact `src` and the destination artifact `dest`.

## Examples

### Create Links

```
% Create a link between the current Simulink Object and a requirement
link1 = slreq.createLink(gcb, REQ)

link1 =

  Link with properties:

          Type: 'Implement'
   Description: 'Plant Specs'
      Keywords: [0×0 char]
     Rationale: ''
     CreatedOn: 02-Sep-2017 15:49:28
     CreatedBy: 'Jane Doe'
    ModifiedOn: 21-Oct-2017 11:34:12
    ModifiedBy: 'John Doe'
      Comments: [0×0 struct]

% Create a link between a requirement and the current Stateflow object
link2 = slreq.createLink(REQ, sfgco);
```

# Input Arguments

**src — Link source artifact**
structure

The link source artifact, specified as a MATLAB® structure.

**dest — Link destination artifact**
structure

The link destination artifact, specified as a MATLAB structure.

# Output Arguments

**myLink — Link artifact**
slreq.Link object

The link between `src` and `dest`, specified as an `slreq.Link` object.

# See Also
slreq.Link | slreq.LinkSet

**Introduced in R2018a**

# slreq.editor

Open Requirements Editor

## Syntax

```
slreq.editor()
```

## Description

`slreq.editor()` opens the Requirements Editor user interface (UI) dialog box.

## See Also
`slreq.ReqSet`

**Introduced in R2018a**

# slreq.find

Find requirement, reference, and link set artifacts

## Syntax

```
myArtifacts = slreq.find('Type',ArtifactType)
myArtifact = slreq.find('Type',ArtifactType,'PropertyName',
PropertyValue)
```

## Description

myArtifacts = slreq.find('Type',ArtifactType) finds and returns all loaded
Simulink Requirements™ artifacts myArtifacts of the type specified by
ArtifactType.

myArtifact = slreq.find('Type',ArtifactType,'PropertyName',
PropertyValue) finds and returns a Simulink Requirements artifact myArtifact of the
type specified by ArtifactType matching the additional properties specified by
PropertyName and PropertyValue.

## Examples

### Find Requirements Sets

```
% Find all requirements sets

allReqSets = slreq.find('Type', 'ReqSet')

allReqSets =

  1×8 ReqSet array with properties:

    Description
    Name
```

```
        Filename
        Revision
        Dirty
        CustomAttributeNames

% Find a requirements set with matching property values
myReqSet = slreq.find('Type', 'ReqSet', 'Name', 'My_Req_Set', 'Revision', 65)

myReqSet =

  ReqSet with properties:

              Description: ''
                     Name: 'My_Req_Set'
                 Filename: 'C:\MATLAB\My_Req_Set.slreqx'
                 Revision: 65
                    Dirty: 0
    CustomAttributeNames: {}
```

## Find Requirements

```
% Find all requirements in all loaded requirements sets
allReqs = slreq.find('Type', 'Requirement')

allReqs =

  1×72 Requirement array with properties:

    Id
    Summary
    Keywords
    Description
    Rationale
    SID
    CreatedBy
    CreatedOn
    ModifiedBy
    ModifiedOn
    FileRevision
    Dirty
    Comments

% Find a requirement with matching property value
myReq = slreq.find('Type', 'Requirement', 'Id', '#19')
```

```
myReq =

  Requirement with properties:

              Id: '#19'
         Summary: 'Control Mode'
        Keywords: [0×0 char]
     Description: ''
       Rationale: ''
             SID: 19
       CreatedBy: 'Jane Doe'
       CreatedOn: 27-Feb-2017 10:15:38
      ModifiedBy: 'John Doe'
      ModifiedOn: 02-Aug-2017 15:18:55
    FileRevision: 52
           Dirty: 0
        Comments: [0×0 struct]
```

## Find Referenced Requirements

```
% Find all referenced requirements in all loaded requirements sets
allRefs = slreq.find('Type', 'Reference')

allRefs =

  1×24 Reference array with properties:

    Keywords
    Artifact
    Id
    Summary
    Description
    SID
    Domain
    SynchronizedOn
    ModifiedOn

% Find a referenced requirement with matching property value
myRef = slreq.find('Type', 'Reference', 'Id', '#26')

myRef =

  Reference with properties:
```

```
          Keywords: [0×0 char]
          Artifact: 'My_req_doc.docx'
                Id: '#26'
           Summary: 'Overview'
       Description: ''
               SID: 2
            Domain: 'linktype_rmi_word'
     SynchronizedOn: 25-Jul-2017 11:34:02
        ModifiedOn: 16-Aug-2017 13:01:55
```

## Find Link Sets

```
% Find all loaded link sets

allLinkSets = slreq.find('Type', 'LinkSet')

allLinkSets =

  1×2 LinkSet array with properties:

    Description
    Filename
    Artifact
    Domain
    Revision
    Dirty

% Find a link set with matching property values
myLinkSet = slreq.find('Type', 'LinkSet', 'Domain', 'linktype_rmi_slreq')

myLinkSet =

  LinkSet with properties:

    Description: ''
       Filename: 'C:\MATLAB\My_Reqs.slmx'
       Artifact: 'C:\MATLAB\My_Reqs.slreqx'
         Domain: 'linktype_rmi_slreq'
       Revision: 2
          Dirty: 0
```

## Input Arguments

**ArtifactType — Simulink Requirements artifact type**
`'ReqSet'` | `'Requirement'` | `'Reference'` | `'LinkSet'`

The Simulink Requirements artifact to find.

## Output Arguments

**myArtifacts — Simulink Requirements artifact array**
`slreq.ReqSet` array | `slreq.Requirement` array | `slreq.Reference` array | `slreq.LinkSet` array

Simulink Requirements artifacts, returned as arrays of the respective data type.

**myArtifact — Simulink Requirements artifact**
`slreq.ReqSet` | `slreq.Requirement` | `slreq.Reference` | `slreq.LinkSet`

Simulink Requirements artifact, returned as the respective data type.

## See Also
`find` | `find` | `find` | `slreq.LinkSet` | `slreq.Reference` | `slreq.ReqSet` | `slreq.Requirement`

**Introduced in R2018a**

# slreq.generateReport

Generate report for requirements set

## Syntax

```
myReportPath = slreq.generateReport(reqSetList, reportOpts)
```

## Description

`myReportPath = slreq.generateReport(reqSetList, reportOpts)` generates a report for the requirements sets specified by `reqSetList` using the options specified by `reportOpts` and returns the path `myReportPath` to the report.

## Examples

### Generate Requirement Report

```
% Generate a requirement report in Microsoft® Word
% format for all loaded requirements sets

% Get default report generation options structure
myReportOpts = slreq.getReportOptions();

% Specify the generated report path and file name
myReportOpts.reportPath = 'L:\My_Project\Reqs_Report.docx';

% Generate the report for all loaded requirements sets
myReport = slreq.generateReport('all', myReportOpts);
```

**Note** To generate reports in PDF and HTML formats, specify a `.pdf` or a `.html` file name as the `reportPath` value.

# Input Arguments

**reqSetList — Requirements set**
character vector (default) | `slreq.ReqSet` object | array

Requirements sets for report generation. You can specify a single requirements set or an array of requirements sets. To generate a report for all the loaded requirements sets, specify `'all'` as the `reqSetList` value. If you do not specify a value for `reqSetList`, `'all'` is used as default.

**reportOpts — Report generation options**
structure

Report generation options specified as a MATLAB structure. If `reportOpts` is not specified, the report is generated using the default options specified in `slreq.getReportOptions`.

**Options**

| Fields | Data Type | Description |
|---|---|---|
| reportPath | character vector | Generated report path. |
| templatePath | character vector | Report template path. |
| titleText | character vector | Report title. |
| authors | character vector | Report authors. |
| includes.toc | Boolean | Option to include table of contents in your report. |
| includes.links | Boolean | Option to include requirements links in your report. |
| includes.rationale | Boolean | Option to include requirements rationale in your report. |
| includes.customAttributes | Boolean | Option to include requirements set custom attributes in your report |
| includes.comments | Boolean | Option to include requirement comments in your report. |
| includes.implementationStatus | Boolean | Option to include requirement implementation status data in your report. |
| includes.verificationStatus | Boolean | Option to include requirement verification status data in your report. |
| includes.keywords | Boolean | Option to include requirement implementation status data in your report. |
| includes.emptySections | Boolean | Option to include empty sections in your report. |
| includes.revision | Boolean | Option to include requirement revision information in your report. |

# Output Arguments

**`myReportPath` — Generated report path**
character vector

The file path for the generated report, specified as a character vector.

# See Also
`slreq.getReportOptions`

## Topics
"Report Requirements Information"

**Introduced in R2018a**

# slreq.getReportOptions

Get default report generation options

## Syntax

```
myOptions = slreq.getReportOptions()
```

## Description

`myOptions = slreq.getReportOptions()` returns a structure with the default options for generating reports for requirements sets.

## Examples

### Get Report Generation Options

```
myOptions = slreq.getReportOptions()

myOptions =

  struct with fields:

      reportPath: 'L:\slreqrpt_20170826.docx'
    templatePath: 'C:\matlab\toolbox\slrequirements\slrequirements\+slreq\+report\templ
       titleText: ''
         authors: 'Jane Doe'
        includes: [1×1 struct]
```

## Output Arguments

**myOptions — Report generation options**
structure

Default options for report generation, returned as a MATLAB structure.

**Options**

| Fields | Data Type | Description |
| --- | --- | --- |
| reportPath | character vector | Generated report path. |
| templatePath | character vector | Report template path. |
| titleText | character vector | Report title. |
| authors | character vector | Report authors. |
| includes.toc | Boolean | Option to include table of contents in your report. |
| includes.links | Boolean | Option to include requirements links in your report. |
| includes.linkGroup | character vector | Option to group links by artifact or by link type. |
| includes.rationale | Boolean | Option to include requirements rationale in your report. |
| includes.customAttributes | Boolean | Option to include requirements set custom attributes in your report |
| includes.comments | Boolean | Option to include requirement comments in your report. |
| includes.implementationStatus | Boolean | Option to include requirement implementation status data in your report. |
| includes.verificationStatus | Boolean | Option to include requirement verification status data in your report. |
| includes.keywords | Boolean | Option to include requirement implementation status data in your report. |
| includes.emptySections | Boolean | Option to include empty sections in your report. |

| Fields | Data Type | Description |
|---|---|---|
| includes.revision | Boolean | Option to include requirement revision information in your report. |

## See Also
slreq.generateReport

**Introduced in R2018a**

# slreq.import

Import requirements from external documents

## Syntax

```
slreq.import(docPath)
[refCount, reqSetFilePath, reqSetObj] = slreq.import(docPath)
slreq.import(docType)
slreq.import(docPath, Name, Value)
```

## Description

`slreq.import(docPath)` imports requirements content as referenced requirements from an external document located at `docPath`. The imported requirements are saved in a new requirements set with the same name as the external document. Use this import method to import requirements content from Microsoft Office documents and from files in the Requirements Interchange Format (`.reqif` and `.reqifz`).

`[refCount, reqSetFilePath, reqSetObj] = slreq.import(docPath)` imports requirements content as referenced requirements from an external document located at `docPath` and returns the number of references imported `refCount`. The imported requirements are saved in the requirements set `reqSetObj` located at `reqSetFilePath` with the same name as the external document.

`slreq.import(docType)` imports requirements content as referenced requirements from an external document that is of a registered document type `docType`. The imported requirements are saved in a new requirements set with the same name as the external document.

`slreq.import(docPath, Name, Value)` imports requirements content as referenced requirements from an external document located at `docPath` with options specified by one or more `Name, Value` pair arguments.

# Examples

## Import Referenced Requirements

```
% Import referenced requirements from Microsoft Office documents
slreq.import('Specification002.docx');
slreq.import('D:/Projects/Requirements/Safety321.xlsx');

% Import referenced requirements from an IBM Rational DOORS Module
slreq.import('linktype_rmi_doors');
```

# Input Arguments

### docPath — Document location
character vector

The file path of the external requirements document, specified as a character vector.

### docType — Document type
character vector

The document type of the external requirements document, specified as a character vector.

Example: `'linktype_rmi_doors'`

## Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`'  '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'ReqSet', 'design_specs.slreqx'`

### AsReference — Option to import as references
`true` (default) | `false`

Option to import requirements as references, specified as a Boolean value. The value `false` is supported only for import from Microsoft Office documents.

**ReqSet — Requirements Set**
character vector

The name for the existing requirements set that you import requirements into, specified as a character vector.

Example: `'ReqSet', 'My_Requirements_Set'`

**RichText — Option to import rich text requirements**
`false` (default) | `true`

Option to import requirements as rich text, specified as a Boolean value.

Example: `'RichText', true`

**bookmarks — Option to import requirements using bookmarks**
`false` | `true`

Option to import requirements content using user-defined bookmarks. This value is `true` by default for Microsoft Word documents and `false` by default for Microsoft Excel® spreadsheets.

Example: `'bookmarks', false`

**match — Regular expression pattern**
character vector

Regular expression pattern for ID search in Microsoft Office documents.

Example: `'match', '^REQ\d+'`

**attributes — Attribute names**
cell array

Attribute names to import, specified as a cell array.

**Note** When importing requirements from a Microsoft Excel spreadsheet, the length of this cell array must match the number of columns specified for import using the `'columns'` argument.

Example: `'attributes', {'Test Status', 'Test Procedure'}`

**Pairs for Microsoft Excel Spreadsheets**

**`columns` — Range of columns**
`double` array

Range of columns to import, specified as a `double` array.

Example: `'columns', [1 6]`

**`rows` — Range of rows**
`double` array

Range of rows to import, specified as a `double` array.

Example: `'rows', [3 35]`

**`idColumn` — ID Column**
`double`

Column in the Microsoft Excel spreadsheet that you want to correspond to the **ID** field in your requirements set, specified as a `double`.

Example: `'idColumn', 1`

**`summaryColumn` — Summary Column**
`double`

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Summary** field in your requirements set, specified as a `double`.

Example: `'summaryColumn', 4`

**`keywordsColumn` — Keywords Column**
`double`

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Keywords** field in your requirements set, specified as a `double`.

Example: `'keywordsColumn', 3`

**`descriptionColumn` — Description Column**
`double`

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Description** field in your requirements set, specified as a `double`.

Example: `'descriptionColumn', 2`

**`rationaleColumn` — Rationale Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Rationale** field in your requirements set, specified as a `double`.

Example: `'rationaleColumn', 5`

# Output Arguments

**`refCount` — Imported referenced requirements count**
double

Number of referenced requirements imported, returned as a `double`.

**`reqSetFilePath` — Requirements set file path**
character vector

The file path of the requirements set to which you import requirements to, returned as a character vector.

**`reqSetObj` — Requirements set object**
slreq.ReqSet object

Handle to the requirements set to which you import requirements to, returned as an `slreq.ReqSet` object.

# See Also

createReferences | slreq.Reference

**Introduced in R2018a**

# slreq.load

Load requirements/link set

## Syntax

```
myReqSet = slreq.load(reqSetArtifact)
myLinkSet = slreq.load(linkSetArtifact)
```

## Description

`myReqSet = slreq.load(reqSetArtifact)` loads a requirements set `myReqSet` into memory.

`myLinkSet = slreq.load(linkSetArtifact)` loads a link set `myLinkSet` into memory.

## Input Arguments

**reqSetArtifact — Requirements set to load**
character vector

The requirements set to load, specified as a character vector.

**linkSetArtifact — Link set artifact name**
character vector

The link set to load, specified as a character vector.

## Output Arguments

**myReqSet — Loaded requirements set**
slreq.ReqSet object

The requirements set that was loaded, returned as an `slreq.ReqSet` object.

**myLinkSet — Loaded link set**
slreq.LinkSet object

The link set that was loaded, returned as an slreq.LinkSet object.


# See Also
slreq.LinkSet | slreq.ReqSet

**Introduced in R2018a**

# slreq.new

Create requirements set

## Syntax

```
newReqSet = slreq.new(reqSetName)
newReqSet = slreq.new(reqSetPath)
```

## Description

`newReqSet = slreq.new(reqSetName)` creates a requirements set `newReqSet` with the name specified by `reqSetName` in the current working folder.

`newReqSet = slreq.new(reqSetPath)` creates a requirements set `newReqSet` in the folder specified by `reqSetPath`.

**Note** The folder specified by `reqSetPath` must exist on disk.

## Examples

### Create Requirements Set

```
% Create requirements set in current working folder
myReqSet1 = slreq.new('New_Req_Set_1')

myReqSet1 =

  ReqSet with properties:

            Description: ''
                   Name: 'New_Req_Set_1'
               Filename: 'L:\New_Req_Set_1.slreqx'
               Revision: 1
                  Dirty: 1
```

```
      CustomAttributeNames: {}
                 CreatedBy: 'John Doe'
                 CreatedOn: 18-Feb-2008 20:54:52
                ModifiedBy: 'Jane Doe'
                ModifiedOn: 20-Jan-2016 12:44:12

% Create requirements set in a different directory
myReqSet2 = slreq.new('L:\Reqs_Work\New_Req_Set_2')

myReqSet2 =

  ReqSet with properties:

               Description: ''
                      Name: 'New_Req_Set_2'
                  Filename: 'L:\Reqs_Work\New_Req_Set_2.slreqx'
                  Revision: 1
                     Dirty: 1
      CustomAttributeNames: {}
                 CreatedBy: 'Jane Doe'
                 CreatedOn: 11-Jan-2009 11:33:01
                ModifiedBy: 'John Doe'
                ModifiedOn: 18-Jan-2018 09:07:32
```

## Input Arguments

### reqSetName — Requirements set name
character vector

Name of the requirements set to create, specified as a character vector.

### reqSetPath — Requirements set path
character vector

Folder to create requirements set in, specified as a character vector.

## Output Arguments

### newReqSet — Created requirements set
slreq.ReqSet object

The created requirements set, specified as an `slreq.ReqSet` object.

## See Also
`slreq.ReqSet`

**Introduced in R2018a**

# slreq.open

Open requirements set

## Syntax

```
myReqSet = slreq.open(ReqSetFilePath)
myReqSet = slreq.open(ReqSetName)
```

## Description

`myReqSet = slreq.open(ReqSetFilePath)` loads the requirements set at `ReqSetFilePath` into memory. If the requirements set is already loaded into memory, the Requirements Editor opens. If the requirements set is already loaded and the Requirements Editor is open, the specified requirements set is selected in the Requirements Editor.

`myReqSet = slreq.open(ReqSetName)` loads the requirements set named `ReqSetName` if it can be located.

## Input Arguments

**ReqSetFilePath — Requirements set file path**
character vector

The full file path of the requirements set to be loaded, specified as a character vector.

**ReqSetName — Requirements set name**
character vector

The name of the requirements set to be loaded, specified as a character vector.

## Output Arguments

**myReqSet — Requirements set object**
`slreq.ReqSet` object

Handle to the requirements set you open, returned as an `slreq.ReqSet` object.

## See Also

`slreq.ReqSet`

**Introduced in R2018a**

# rmi

Interact programmatically with Requirements Management Interface

## Syntax

```
reqlinks = rmi('createEmpty')
reqlinks = rmi('get', model)
reqlinks = rmi('get', sig_builder, group_idx)
rmi('set', model, reqlinks)
rmi('set', sig_builder, reqlinks, group_idx)
rmi('cat', model, reqlinks)
cnt = rmi('count', object)
rmi('clearAll', object)
rmi('clearAll', object, 'deep')
rmi('clearAll', object, 'noprompt')
rmi('clearAll', object, 'deep', 'noprompt')

cmdStr = rmi('navCmd', object)
[cmdStr, titleStr] = rmi('navCmd', object)
object = rmi('guidlookup', model, guidStr)
rmi('highlightModel', object)
rmi('unhighlightModel', object)
rmi('view', object, index)
dialog = rmi('edit', object)
guidStr = rmi('guidget', object)

rmi('report', model)
rmi('report', matlabFilePath)
rmi('report', dictionaryFile)
rmi('projectreport')

rmi setup
rmi register linktypename
rmi unregister linktypename
rmi linktypelist

number_problems = rmi('checkdoc')
```

```
number_problems = rmi('checkdoc', docName)
rmi('check', matlabFilePath)
rmi('check', dictionaryFile)

rmi('doorssync', model)

rmi('setDoorsLabelTemplate', template)
template = rmi('getDoorsLabelTemplate')
label = rmi('doorsLabel', moduleID, objectID)
totalModifiedLinks = rmi('updateDoorsLabels', model)
```

## Description

reqlinks = rmi('createEmpty') creates an empty instance of the requirement links data structure.

reqlinks = rmi('get', model) returns the requirement links data structure for model.

reqlinks = rmi('get', sig_builder, group_idx) returns the requirement links data structure for the Signal Builder group specified by the index group_idx.

rmi('set', model, reqlinks) sets reqlinks as the requirements links for model.

rmi('set', sig_builder, reqlinks, group_idx) sets reqlinks as the requirements links for the signal group group_idx in the Signal Builder block sig_builder.

rmi('cat', model, reqlinks) adds the requirements links in reqlinks to existing requirements links for model.

cnt = rmi('count', object) returns the number of requirements links for object.

rmi('clearAll', object) deletes all requirements links for object.

rmi('clearAll', object, 'deep') deletes all requirements links in the model containing object.

rmi('clearAll', object, 'noprompt') deletes all requirements links for object and does not prompt for confirmation.

`rmi('clearAll', object, 'deep', 'noprompt')` deletes all requirements links in the model containing `object` and does not prompt for confirmation.

`cmdStr = rmi('navCmd', object)` returns the MATLAB command `cmdStr` used to navigate to `object`.

`[cmdStr, titleStr] = rmi('navCmd', object)` returns the MATLAB command `cmdStr` and the title `titleStr` that provides descriptive text for `object`.

`object = rmi('guidlookup', model, guidStr)` returns the object name in `model` that has the globally unique identifier `guidStr`.

`rmi('highlightModel', object)` highlights all of the objects in the parent model of `object` that have requirement links.

`rmi('unhighlightModel', object)` removes highlighting of objects in the parent model of `object` that have requirement links.

`rmi('view', object, index)` accesses the requirement numbered `index` in the requirements document associated with `object`.

`dialog = rmi('edit', object)` displays the Requirements dialog box for `object` and returns the handle of the dialog box.

`guidStr = rmi('guidget', object)` returns the globally unique identifier for `object`. A globally unique identifier is created for `object` if it lacks one.

`rmi('report', model)` generates a Requirements Traceability report in HTML format for `model`.

`rmi('report', matlabFilePath)` generates a Requirements Traceability report in HTML format for the MATLAB code file specified by `matlabFilePath`.

`rmi('report', dictionaryFile)` generates a Requirements Traceability report in HTML format for the Simulink data dictionary specified by `dictionaryFile`.

`rmi('projectreport')` generates a Requirements Traceability report in HTML format for the current Simulink Project. The master page of this report has HTTP links to reports for each project item that has requirements traceability associations. For more information, see "Create Requirements Traceability Report for Simulink Project".

`rmi setup` configures RMI for use with your MATLAB software and installs the interface for use with the IBM® Rational® DOORS® software.

`rmi register linktypename` registers the custom link type specified by the function `linktypename`. For more information, see "Custom Link Type Registration".

`rmi unregister linktypename` removes the custom link type specified by the function `linktypename`. For more information, see "Custom Link Type Registration".

`rmi linktypelist` displays a list of the currently registered link types. The list indicates whether each link type is built-in or custom, and provides the path to the function used for its registration.

`number_problems = rmi('checkdoc')` checks validity of links to Simulink from a requirements document in Microsoft Word, Microsoft Excel, or IBM Rational DOORS. It prompts for the requirements document name, returns the total number of problems detected, and opens an HTML report in the MATLAB Web browser. For more information, see "Validate Requirements Links in a Requirements Document".

`number_problems = rmi('checkdoc', docName)` checks validity of links to Simulink from the requirements document specified by `docName`. It returns the total number of problems detected and opens an HTML report in the MATLAB Web browser. For more information, see "Validate Requirements Links in a Requirements Document".

`rmi('check', matlabFilePath)` checks consistency of traceability links associated with MATLAB code lines in the `.m` file `matlabFilePath`, and opens an HTML report in the MATLAB Web browser.

`rmi('check', dictionaryFile)` checks consistency of traceability links associated with the Simulink data dictionary `dictionaryFile`, and opens an HTML report in the MATLAB Web browser.

`rmi('doorssync', model)` opens the DOORS synchronization settings dialog box, where you can customize the synchronization settings and synchronize your model with an open project in an IBM Rational DOORS database.

`rmi('setDoorsLabelTemplate', template)` specifies a new custom template for labels of requirements links to IBM Rational DOORS. The default label template contains the section number and object heading for the DOORS requirement link target. To revert the link label template back to the default, enter `rmi('setDoorsLabelTemplate', '')` at the MATLAB command prompt.

`template = rmi('getDoorsLabelTemplate')` returns the currently specified custom template for labels of requirements links to IBM Rational DOORS.

label = rmi('doorsLabel', moduleID, objectID) generates a label for the
requirements link to the IBM Rational DOORS object specified by objectID in the
DOORS module specified by moduleID, according to the current template.

totalModifiedLinks = rmi('updateDoorsLabels', model) updates all IBM
Rational DOORS requirements links labels in model according to the current template.

# Examples

### Requirements Links Management in Example Model

Get a requirement associated with a block in the slvnvdemo_fuelsys_htmreq model,
change its description, and save the requirement back to that block. Define a new
requirement link and add it to the existing requirements links in the block.

Get requirement link associated with the Airflow calculation block in the
slvnvdemo_fuelsys_htmreq example model.

```
slvnvdemo_fuelsys_htmreq;
blk_with_req =  ['slvnvdemo_fuelsys_htmreq/fuel rate' 10 'controller/...
   Airflow calculation'];
reqts = rmi('get', blk_with_req);
```

Change the description of the requirement link.

```
reqts.description = 'Mass airflow estimation';
```

Save the changed requirement link description for the Airflow calculation block.

```
rmi('set', blk_with_req, reqts);
```

Create new requirement link to example document fuelsys_requirements2.htm.

```
new_req = rmi('createempty');
new_req.doc = 'fuelsys_requirements2.htm';
new_req.description = 'A new requirement';
```

Add new requirement link to existing requirements links for the Airflow calculation block.

```
rmi('cat', blk_with_req, new_req);
```

**Requirements Traceability Report for Example Model**

Create HTML report of requirements traceability data in example model.

Create an HTML requirements report for the `slvnvdemo_fuelsys_htmreq` example model.

```
rmi('report', 'slvnvdemo_fuelsys_htmreq');
```

The MATLAB Web browser opens, showing the report.

**Labels for Requirements Links to IBM Rational DOORS**

Specify a new label template for links to requirements in DOORS, and update labels of all DOORS requirements links in your model to fit the new template.

Specify a new label template for requirements links to IBM Rational DOORS so that new links to DOORS objects are labeled with the corresponding module ID, object absolute number, and the value of the 'Backup' attribute.

```
rmi('setDoorsLabelTemplate', '%m:%n [backup=%<Backup>]');
```

Specify a new label template for requirements links to IBM Rational DOORS and set the maximum label length to (for example) 200 characters.

```
rmi('setDoorsLabelTemplate', '%h %200');
```

Update existing DOORS requirements link labels to match the new specified template in your model `example_model`. When updating labels, DOORS must be running and all linked modules must be accessible for reading.

```
rmi('updateDoorsLabels', example_model);
```

# Input Arguments

**`model` — Simulink or Stateflow model with which requirements can be associated**
name | handle

Simulink or Stateflow model with which requirements can be associated, specified as a character vector or handle.

Example: `'slvnvdemo_officereq'`

Data Types: `char`

### `object` — Model object with which requirements can be associated
name | handle

Model object with which requirements can be associated, specified as a character vector or handle.

Example: `'slvnvdemo_fuelsys_htmreq/fuel rate controller/Airflow calculation'`

Data Types: `char`

### `sig_builder` — Signal Builder block containing signal group with requirements traceability associations
name | handle

Signal Builder block containing signal group with requirements traceability associations, specified as a character vector or handle.

Data Types: `char`

### `group_idx` — Signal Builder group index
integer

Signal Builder group index, specified as a scalar.

Example: 2

Data Types: `char`

### `matlabFilePath` — MATLAB code file with requirements traceability associations
path

MATLAB code file with requirements traceability associations, specified as the path to the file.

Example:

Data Types: `char`

### `dictionaryFile` — Simulink data dictionary with requirements traceability associations
character vector

Simulink data dictionary with requirements traceability associations, specified as a character vector containing the file name and, optionally, path of the dictionary.

Example:

Data Types: `char`

### `guidStr` — Globally unique identifier for model object
character vector

Globally unique identifier for model object `object`, specified as a character vector.

Example: `GIDa_59e165f5_19fe_41f7_abc1_39c010e46167`

Data Types: `char`

### `index` — Index number of requirement linked to model object
integer

Index number of requirement linked to model object, specified as an integer.

### `docName` — Requirements document in external application
file name | path

Requirements document in external application, specified as a character vector that represents one of the following:

- IBM Rational DOORS module ID.
- path to Microsoft Word requirements document.
- path to Microsoft Excel requirements document.

For more information, see "Validate Requirements Links in a Requirements Document".

### `label` — Label for links to requirements in IBM Rational DOORS
character vector

Example:

Data Types: `char`

### `template` — Template label for links to requirements in IBM Rational DOORS
character vector

Template label for links to requirements in IBM Rational DOORS, specified as a character vector.

You can use the following format specifiers to include the associated DOORS information in your requirements links labels:

| | |
|---|---|
| %h | Object heading |
| %t | Object text |
| %p | Module prefix |
| %n | Object absolute number |
| %m | Module ID |
| %P | Project name |
| %M | Module name |
| %U | DOORS URL |
| %<ATTRIBUTE_NAME> | Other DOORS attribute you specify |

Example: '%m:%n [backup=%<Backup>]'

Data Types: char

### moduleID — IBM Rational DOORS module
DOORS module ID

IBM Rational DOORS module, specified as the unique DOORS module ID.

Example:

Data Types: char

### objectID — IBM Rational DOORS object
DOORS object ID

IBM Rational DOORS object in the DOORS module moduleID, specified as the locally unique DOORS ID.

Example:

Data Types: char

# Output Arguments

### reqlinks — Requirement links data
struct

Requirement links data, returned as a structure array with the following fields:

| | |
|---|---|
| doc | Character vector identifying requirements document |
| id | Character vector defining location in requirements document. The first character specifies the identifier type: |

| First Character | Identifier | Example |
|---|---|---|
| ? | Search text, the first occurrence of which is located in requirements document | `'?Requirement 1'` |
| @ | Named item, such as bookmark in a Microsoft Word file or an anchor in an HTML file | `'@my_req'` |
| # | Page or item number | `'#21'` |
| > | Line number | `'>3156'` |
| $ | Worksheet range in a spreadsheet | `'$A2:C5'` |

| | |
|---|---|
| linked | Boolean value specifying whether the requirement link is accessible for report generation and highlighting:<br>1 (default). Highlight model object and include requirement link in reports.<br>0 |
| description | Character vector describing the requirement |
| keywords | Optional character vector supplementing `description` |
| reqsys | Character vector identifying the link type registration name; `'other'` for built-in link types |

**`cmdStr` — Command used to navigate to model object**
character vector

Command used to navigate to model object `object`, returned as a character vector.

Example: `rmiobjnavigate('slvnvdemo_fuelsys_officereq.slx', 'GIDa_59e165f5_19fe_41f7_abc1_39c010e46167');`

**`titleStr` — Textual description of model object with requirements links**
character vector

Textual description of model object with requirements links, returned as a character vector.

Example: `slvnvdemo_fuelsys_officereq/.../Airflow calculation/Pumping Constant (Lookup2D)`

**`guidStr` — Globally unique identifier for model object**
character vector

Globally unique identifier for model object `object`, returned as a character vector.

Example: `GIDa_59e165f5_19fe_41f7_abc1_39c010e46167`

**`dialog` — Requirements dialog box for model object**
handle

Requirements dialog box for model object `object`, returned as a handle to the dialog box.

**`number_problems` — Total count of invalid links detected in external document**
integer

Total count of invalid links detected in external document `docName`.

For more information, see "Validate Requirements Links in a Requirements Document".

**`totalModifiedLinks` — Total count of DOORS requirements links updated with new label template**
integer

Total count of DOORS requirements links updated with new label template.

# See Also

`rmipref` | `rmiobjnavigate` | `rmidocrename` | `rmitag` | `rmimap.map` | `RptgenRMI.doorsAttribs`

**Introduced in R2006b**

# rmidata.export

Move requirements traceability data to external `.req` file

## Syntax

```
[total_linked,total_links] = rmidata.export
[total_linked,total_links] = rmidata.export(model)
```

## Description

`[total_linked,total_links] = rmidata.export` moves requirements traceability data associated with the current Simulink model to an external file named *model_name*`.req`. `rmidata.export` saves the file in the same folder as the model. `rmidata.export` deletes the requirements traceability data stored in the model and saves the modified model.

`[total_linked,total_links] = rmidata.export(model)` moves requirements traceability data associated with `model` to an external file named *model_name*`.req`. `rmidata.export` saves the file in the same folder as `model`. `rmidata.export` deletes the requirements traceability data stored in the model and saves the modified model.

## Input Arguments

**model**

Name or handle of a Simulink model

## Output Arguments

**total_linked**

Integer indicating the number of objects in the model that have linked requirements

**total_links**

Integer indicating the total number of requirements links in the model

# Examples

Move the requirements traceability data from the slvnvdemo_fuelsys_officereq model to an external file:

```
rmidata.export('slvnvdemo_fuelsys_officereq');
```

# See Also

rmi | rmidata.save | rmimap.map

## Topics
"Requirements Link Storage"

**Introduced in R2011b**

# rmimap.map

Associate externally stored requirements traceability data with model

## Syntax

```
rmimap.map(model,reqts_file)
rmimap.map(model,'undo')
rmimap.map(model,'clear')
```

## Description

`rmimap.map(model,reqts_file)` associates the requirements traceability data from `reqts_file` with the Simulink model `model`.

`rmimap.map(model,'undo')` removes from the `.slmx` file associated with `model` the requirements traceability data that was most recently saved in the `.slmx` file.

`rmimap.map(model,'clear')` removes from the `.slmx` file associated with `model` all requirements traceability data.

## Input Arguments

**model**

Name, handle, or full path for a Simulink model

**reqts_file**

Full path to the `.slmx` file that contains requirements traceability data for the model

## Alternatives

To load a file that contains requirements traceability data for a model:

**1**   Open the model.

**2**   Select **Analysis > Requirements > Load Links**.

> **Note**  The **Load Links** menu item appears only when your model is configured to store requirements data externally. To specify external storage of requirements data for your model, in the Requirements Settings dialog box under **Storage > Default storage location for requirements links data**, select **Store externally (in a separate *.slmx file)**.

**3**   Browse to the `.slmx` file that contains the requirements links.

**4**   Click **OK**.

# Examples

Associate an external requirements traceability data file with a Simulink model. After associating the information with the model, view the objects with linked requirements by highlighting the model.

```
open_system('slvnvdemo_powerwindowController');
reqFile = fullfile(matlabroot, 'toolbox', 'slvnv', ...
    'rmidemos', 'powerwin_reqs', ...
    'slvnvdemo_powerwindowRequirements.slmx');
rmimap.map('slvnvdemo_powerwindowController', reqFile);
rmi('highlightModel', 'slvnvdemo_powerwindowController');
```

To clear the requirements you just associated with that model, run this `rmimap.map` command:

```
rmimap.map('slvnvdemo_powerwindowController','clear');
```

# See Also

`rmi` | `rmidata.save` | `rmidata.export`

## Topics
"Requirements Link Storage"

**Introduced in R2015a**

# rmidata.save

Save requirements traceability data in external `.req` file

## Syntax

```
rmidata.save(model)
```

## Description

`rmidata.save(model)` saves requirements traceability data for a model in an external `.req` file. The model must be configured to store requirements traceability data externally. This function is equivalent to **Analysis > Requirements > Save Links** in the Simulink Editor.

## Examples

### Create New Requirement Link and Save Externally

Add a requirement link to an existing example model, and save the model requirements traceability data in an external file.

Open the example model, slvnvdemo_powerwindowController.

```
open_system('slvnvdemo_powerwindowController');
```

Specify that the model store requirements data externally.

```
rmipref('StoreDataExternally',1);
```

Create a new requirements link structure.

```
newReqLink = rmi('createEmpty');
newReqLink.description = 'newReqLink';
```

Specify the requirements document that you want to link to from the model. In this case, an example requirements document is provided.

```
newReqLink.doc = [matlabroot '\toolbox\slvnv\rmidemos\' ...
        'powerwin_reqs\PowerWindowSpecification.docx'];
```

Specify the text of the requirement within the document to which you want to link.

```
newReqLink.id = '?passenger input consists of a vector' ...
        'with three elements';
```

Specify that the new requirements link that you created be attached to the Mux4 block of the slvnvdemo_powerwindowController example model.

```
rmi('set', 'slvnvdemo_powerwindowController/Mux4', newReqLink);
```

Save the new requirement link that you just created in an external .req file associated with the model.

```
rmidata.save('slvnvdemo_powerwindowController');
```

This function is equivalent to the Simulink Editor option **Analysis > Requirements > Save Links**.

To highlight the Mux4 block, turn on requirements highlighting for the slvnvdemo_powerwindowController example model.

```
rmi('highlightModel', 'slvnvdemo_powerwindowController');
```

You can test your requirements link by right-clicking the Mux4 block. In the context menu, select **Requirements > 1. "newReqLink"**.

Close the example model.

```
close_system('slvnvdemo_powerwindowController', 0);
```

You are not prompted to save unsaved changes because you saved the requirements link data outside the model file. The model file remains unchanged.

## Input Arguments

**model — Name or handle of model with requirements links**
character vector | handle

Name of model with requirements links, specified as a character vector, or handle to model with requirements links. The model must be loaded into memory and configured to store requirements traceability data externally.

If you have a new model with no existing requirements links, configure it for external storage as described in "Requirements Link Storage". You can also use the `rmipref` command to specify storage settings.

If you have an existing model with internally stored requirements traceability data, convert that data to external storage as described in "Move Internally Stored Requirements Links to External Storage". You can also use the `rmidata.export` command to convert existing requirements traceability data to external storage.

Example: `'slvnvdemo_powerwindowController'`

Example: `get_param(gcs,'Handle')`

## See Also

`rmimap.map` | `rmidata.export`

### Topics
"Requirements Link Storage"

**Introduced in R2013b**

# rmidocrename

Update model requirements document paths and file names

## Syntax

```
rmidocrename(model_handle, old_path, new_path)
rmidocrename(model_name, old_path, new_path)
```

## Description

rmidocrename(model_handle, old_path, new_path) collectively updates the links from a Simulink model to requirements files whose names or locations have changed. model_handle is a handle to the model that contains links to the files that you have moved or renamed. old_path is a character vector that contains the existing full or partial file or path name. new_path is a character vector with the new full or partial file or path name.

rmidocrename(model_name, old_path, new_path) updates the links to requirements files associated with model_name. You can pass rmidocrename a model handle or a model file name.

When using the rmidocrename function, make sure to enter specific character vectors for the old document name fragments so that you do not inadvertently modify other links.

## Examples

For the current Simulink model, update all links to requirements files that contain the character vector 'project_0220', replacing them with 'project_0221':

```
rmidocrename(gcs, 'project_0220', 'project_0221')
Processed 6 objects with requirements, 5 out of 13 links were modified.
```

## Alternatives

To update the requirements links one at a time, for each model object that has a link:

1.  For each object with requirements, open the Requirements Traceability Link Editor by right-clicking and selecting **Requirements Traceability > Open Link Editor**.
2.  Edit the **Document** field for each requirement that points to a moved or renamed document.
3.  Click **Apply** to save the changes.

## See Also

`rmi`

**Introduced in R2009b**

# rmiobjnavigate

Navigate to model objects using unique Requirements Management Interface identifiers

## Syntax

```
rmiobjnavigate(modelPath, guId)
rmiobjnavigate(modelPath, guId, grpNum)
```

## Description

`rmiobjnavigate(modelPath, guId)` navigates to and highlights the specified object in a Simulink model.

`rmiobjnavigate(modelPath, guId, grpNum)` navigates to the signal group number `grpNum` of a Signal Builder block identified by `guId` in the model `modelPath`.

## Input Arguments

**`modelPath`**

A full path to a Simulink model file, or a Simulink model file name that can be resolved on the MATLAB path.

**`guId`**

A unique identifier that the RMI uses to identify a Simulink or Stateflow object.

**`grpNum`**

Integer indicating a signal group number in a Signal Builder block

## Examples

Open the `slvnvdemo_fuelsys_officereq` example model and get the unique identifier for the MAP Sensor block:

```
% Open example model
slvnvdemo_fuelsys_officereq;
% Get the Ssession Independent Identifier of the MAP Sensor Block
targetSID = Simulink.ID.getSID('slvnvdemo_fuelsys_officereq/MAP sensor');
```

Navigate to the MAP Sensor block using `rmiobjnavigate` and the unique identifier returned in the previous step:

```
% Split targetSID into two components
[targetModel, targetObj] = strtok(targetSID,':');
% Navigate to the MAP sensor using the model name and model guID
rmiobjnavigate(targetModel, targetObj)
```

## See Also

`rmi`

### Topics

"Use the rmiobjnavigate Function"

**Introduced in R2010b**

# rmipref

Get or set RMI preferences stored in `prefdir`

## Syntax

`rmipref`

`currentVal = rmipref(prefName)`

`previousVal = rmipref(Name,Value)`

## Description

`rmipref` returns list of `Name,Value` pairs corresponding to Requirements Management Interface (RMI) preference names and accepted values for each preference.

`currentVal = rmipref(prefName)` returns the current value of the preference specified by `prefName`.

`previousVal = rmipref(Name,Value)` sets a new value for the RMI preference specified by `Name`, and returns the previous value of that RMI preference.

## Examples

**References to Simulink Model in External Requirements Documents**

Choose the type of reference that the RMI uses when it creates links to your model from external requirements documents. The reference to your model can be either the model file name or the full absolute path to the model file.

The value of the `'ModelPathReference'` preference determines how the RMI stores references to your model in external requirements documents. To view the current value of this preference, enter the following code at the MATLAB command prompt.

```
currentVal = rmipref('ModelPathReference')
```

The default value of the `'ModelPathReference'` preference is `'none'`.

```
currentVal =
```

```
none
```

This default value specifies that the RMI uses only the model file name in references to your model that it creates in external requirements documents.

**Automatic Application of User Tags to Selection-Based Requirements Links**

Configure the RMI to automatically apply a specified list of user tag keywords to new selection-based requirements links that you create.

Specify that the user tags `design` and `reqts` apply to new selection-based requirements links that you create.

```
previousVal = rmipref('SelectionLinkTag','design,reqts')
```

When you specify a new value for an RMI preference, `rmipref` returns the previous value of that RMI preference. In this case, `previousVal` is an empty character vector, the default value of the `'SelectionLinkTag'` preference.

```
previousVal =

     ''
```

View the currently specified value for the `'SelectionLinkTag'` preference.

```
currentVal = rmipref('SelectionLinkTag')
```

The function returns the currently specified comma-separated list of user tags.

```
currentVal =
```

```
design,reqts
```

These user tags apply to all new selection-based requirements links that you create.

**Internal Storage of Requirements Traceability Data**

Configure the RMI to embed requirements links data in the model file instead of in a separate `.req` file.

---

**Note** If you have existing requirements links for your model that are stored internally, you need to move these links into an external `.req` file before you change the storage settings for your requirements traceability data. See "Move Internally Stored Requirements Links to External Storage" for more information.

---

If you would like to embed requirements traceability data in the model file, set the `'StoreDataExternally'` preference to `0`.

```
previousVal = rmipref('StoreDataExternally',0)
```

When you specify a new value for an RMI preference, `rmipref` returns the previous value of that RMI preference. By default, the RMI stores requirements links data externally in a separate `.req` file, so the previous value of this preference was `1`.

```
previousVal =

    1
```

After you set the `'StoreDataExternally'` preference to `0`, your requirements links are embedded in the model file.

```
currentVal = rmipref('StoreDataExternally')

currentVal =

    0
```

# Input Arguments

### prefName — RMI preference name
`'BiDirectionalLinking'` | `'FilterRequireTags'` | `'CustomSettings'` | …

RMI preference name, specified as the corresponding `Name` character vector listed in "Name-Value Pair Arguments" on page 1-58.

## Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`'  '`).

Example: `'BiDirectionalLinking',true` enables bidirectional linking for your model, so that when you create a selection-based link to a requirements document, the RMI creates a corresponding link to your model from the requirements document.

**BiDirectionalLinking — Bidirectional selection linking preference**
`false` (default) | `true`

Bidirectional selection linking preference, specified as a logical value.

This preference specifies whether to simultaneously create return link from target to source when creating link from source to target. This setting applies only for requirements document types that support selection-based linking.

Data Types: `logical`

**DocumentPathReference — Preference for path format of links to requirements documents from model**
`'modelRelative'` (default) | `'absolute'` | `'pwdRelative'` | `'none'`

Preference for path format of links to requirements documents from model, specified as one of the following values.

| Value | Document reference contains... |
|---|---|
| `'absolute'` | full absolute path to requirements document. |
| `'pwdRelative'` | path relative to MATLAB current folder. |
| `'modelRelative'` | path relative to model file. |
| `'none'` | document file name only. |

For more information, see "Document Path Storage".

Data Types: `char`

**ModelPathReference — Preference for path format in links to model from requirements documents**
`'none'` (default) | `'absolute'`

Preference for path format in links to model from requirements documents, specified as one of the following values.

| Value | Model reference contains... |
|---|---|
| 'absolute' | full absolute path to model. |
| 'none' | model file name only. |

Data Types: `char`

### LinkIconFilePath — Preference to use custom image file as requirements link icon
empty character vector (default) | full image file path

Preference to use custom image file as requirements link icon, specified as full path to icon or small image file. This image will be used for requirements links inserted in external documents.

Data Types: `char`

### FilterEnable — Preference to enable filtering by user tag keywords
`false` (default) | `true`

Preference to enable filtering by user tag keywords, specified as a logical value. When you filter by user tag keywords, you can include or exclude subsets of requirements links in highlighting or reports. You can specify user tag keywords for requirements links filtering in the `'FilterRequireTags'` and `'FilterExcludeTags'` preferences. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `logical`

### FilterRequireTags — Preference for user tag keywords for requirements links
empty character vector (default) | comma-separated list of user tag keywords

Preference for user tag keywords for requirements links, specified as a comma-separated list of words or phrases in a character vector. These user tags apply to all new requirements links you create. Requirements links with these user tags are included in model highlighting and reports. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `char`

**`FilterExcludeTags` — Preference to exclude certain requirements links from model highlighting and reports**
empty character vector (default) | comma-separated list of user tag keywords

Preference to exclude certain requirements links from model highlighting and reports, specified as a comma-separated list of user tag keywords. Requirements links with these user tags are excluded from model highlighting and reports. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `char`

**`FilterMenusByTags` — Preference to disable labels of requirements links with designated user tags**
`false` (default) | `true`

Preference to disable labels of requirements links with designated user tags, specified as a logical value. When set to `true`, if a requirement link has a user tag designated in `'FilterExcludeTags'` or `'FilterRequireTags'`, that requirements link will be disabled in the Requirements context menu. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `logical`

**`FilterConsistencyChecking` — Preference to filter Model Advisor requirements consistency checks with designated user tags**
`false` (default) | `true`

Preference to filter Model Advisor requirements consistency checks with designated user tags, specified as a logical value. When set to `true`, Model Advisor requirements consistency checks include requirements links with user tags designated in `'FilterRequireTags'` and excludes requirements links with user tags designated in `'FilterExcludeTags'`. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `logical`

**`KeepSurrogateLinks` — Preference to keep DOORS surrogate links when deleting all requirements links**
empty (default) | `false` | `true`

Preference to keep DOORS surrogate links when deleting all requirements links, specified as a logical value. When set to `true`, selecting **Requirements > Delete All Links** deletes all requirements links including DOORS surrogate module requirements links.

When not set to `true` or `false`, selecting **Requirements > Delete All Links** opens a dialog box with a choice to keep or delete DOORS surrogate links.

Data Types: `logical`

### `ReportFollowLibraryLinks` — Preference to include requirements links in referenced libraries in generated report
`false` (default) | `true`

Preference to include requirements links in referenced libraries in generated report, specified as a logical value. When set to `true`, generated requirements reports include requirements links in referenced libraries.

Data Types: `logical`

### `ReportHighlightSnapshots` — Preference to include highlighting in model snapshots in generated report
`true` (default) | `false`

Preference to include highlighting in model snapshots in generated report, specified as a logical value. When set to `true`, snapshots of model objects in generated requirements reports include highlighting of model objects with requirements links.

Data Types: `logical`

### `ReportNoLinkItems` — Preference to include model objects with no requirements links in generated requirements reports
`false` (default) | `true`

Preference to include model objects with no requirements links in generated requirements reports, specified as a logical value. When set to `true`, generated requirements reports include lists of model objects that have no requirements links.

Data Types: `logical`

### `ReportUseDocIndex` — Preference to include short document ID instead of full path to document in generated requirements reports
`false` (default) | `true`

Preference to include short document ID instead of full path to document in generated requirements reports, specified as a logical value. When set to `true`, generated requirements reports include short document IDs, when specified, instead of full paths to requirements documents.

Data Types: `logical`

### ReportIncludeTags — Preference to list user tags for requirements links in generated reports
`false` (default) | `true`

Preference to list user tags for requirements links in generated reports, specified as a logical value. When set to `true`, generated requirements reports include user tags specified for each requirement link. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `logical`

### ReportDocDetails — Preference to include extra detail from requirements documents in generated reports
`false` (default) | `true`

Preference to include extra detail from requirements documents in generated reports, specified as a logical value. When set to `true`, generated requirements reports load linked requirements documents to include additional information about linked requirements. This preference applies to Microsoft Word, Microsoft Excel, and IBM Rational DOORS requirements documents only.

Data Types: `logical`

### ReportLinkToObjects — Preference to include links to model objects in generated requirements reports
`false` (default) | `true`

Preference to include links to model objects in generated requirements reports, specified as a logical value. When set to `true`, generated requirements reports include links to model objects. These links work only if the MATLAB internal HTTP server is active.

Data Types: `logical`

### SelectionLinkWord — Preference to include Microsoft Word selection link option in Requirements context menu
`true` (default) | `false`

Preference to include Microsoft Word selection link option in Requirements context menu, specified as a logical value.

Data Types: `logical`

### SelectionLinkExcel — Preference to include Microsoft Excel selection link option in Requirements context menu
`true` (default) | `false`

Preference to include Microsoft Excel selection link option in Requirements context menu, specified as a logical value.

Data Types: `logical`

### SelectionLinkDoors — Preference to include IBM Rational DOORS selection link option in Requirements context menu
`true` (default) | `false`

Preference to include IBM Rational DOORS selection link option in Requirements context menu, specified as a logical value.

Data Types: `logical`

### SelectionLinkTag — Preference for user tags to apply to new selection-based requirements links
empty character vector (default) | comma-separated list of user tag keywords

Preference for user tags to apply to new selection-based requirements links, specified as a comma-separated list of words or phrases in a character vector. These user tags automatically apply to new selection-based requirements links that you create. For more information about requirements filtering, see "Filter Requirements with User Tags".

Data Types: `char`

### StoreDataExternally — Preference to store requirements links data in external `.req` file
`false` (default) | `true`

Preference to store requirements links data in external `.req` file, specified as a logical value. This setting applies to all new models and to existing models that do not yet have requirements links. For more information about storage of requirements links data, see "Requirements Link Storage".

Data Types: `logical`

### UseActiveXButtons — Preference to use legacy ActiveX® buttons in Microsoft Office requirements documents
`false` (default) | `true`

Preference to use legacy ActiveX buttons in Microsoft Office requirements documents, specified as a logical value. The default value of this preference is `false`; requirements links are URL-based by default. ActiveX requirements navigation is supported for backward compatibility.

Data Types: `logical`

**CustomSettings — Preference for storing custom settings**
`inUse:` `0` (default) | structure array of custom field names and settings

Preference for storing custom settings, specified as a structure array. Each field of the structure array corresponds to the name of your custom preference, and each associated value corresponds to the value of that custom preference.

Data Types: `struct`

# Output Arguments

**currentVal — Current value of the RMI preference specified by `prefName`**
`true` | `false` | `'absolute'` | `'none'` | ...

Current value of the RMI preference specified by `prefName`. RMI preference names and their associated possible values are listed in "Name-Value Pair Arguments" on page 1-58.

**previousVal — Previous value of the RMI preference specified by `prefName`**
`true` | `false` | `'absolute'` | `'none'` | ...

Previous value of the RMI preference specified by `prefName`. RMI preference names and their associated possible values are listed in "Name-Value Pair Arguments" on page 1-58.

# See Also
`rmi`

## Topics
"Requirements Settings"

**Introduced in R2013a**

# rmiref.insertRefs

Insert links to models into requirements documents

## Syntax

```
[total_links, total_matches, total_inserted] = rmiref.insertRefs(
model_name, doc_type)
```

## Description

`[total_links, total_matches, total_inserted] = rmiref.insertRefs(
model_name, doc_type)` inserts ActiveX controls into the open, active requirements document of type `doc_type`. These controls correspond to links from `model_name` to the document. With these controls, you can navigate from the requirements document to the model.

## Input Arguments

**model_name**

Name or handle of a Simulink model

**doc_type**

A character vector that indicates the requirements document type:

- `'word'`
- `'excel'`

## Examples

Remove the links in an example requirements document, and then reinsert them:

1  Open the example model:

```
slvnvdemo_fuelsys_officereq
```

2  Open the example requirements document:

```
open([matlabroot strcat('/toolbox/slvnv/rmidemos/fuelsys_req_docs/',...
    'slvnvdemo_FuelSys_DesignDescription.docx')])
```

3  Remove the links from the requirements document:

```
rmiref.removeRefs('word')
```

4  Enter y to confirm the removal.

5  Reinsert the links from the requirements document to the model:

```
[total_links, total_matches, total_inserted] = ...
    rmiref.insertRefs(gcs, 'word')
```

## See Also
```
rmiref.removeRefs
```

**Introduced in R2011a**

# rmiref.removeRefs

Remove links to models from requirements documents

## Syntax

```
rmiref.removeRefs(doc_type)
```

## Description

`rmiref.removeRefs(doc_type)` removes all links to models from the open, active requirements document of type `doc_type`.

## Input Arguments

**doc_type**

A character vector that indicates the requirements document type:

- `'word'`
- `'excel'`
- `'doors'`

## Examples

Remove the links in this example requirements document:

```
open([matlabroot strcat('/toolbox/slvnv/rmidemos/fuelsys_req_docs/', ...
    'slvnvdemo_FuelSys_DesignDescription.docx')])
rmiref.removeRefs('word')
```

## See Also
rmiref.insertRefs

**Introduced in R2011a**

# rmitag

Manage user tags for requirements links

## Syntax

```
rmitag(model, 'list')
rmitag(model, 'add', tag)
rmitag(model, 'add', tag, doc_pattern)
rmitag(model, 'delete', tag)
rmitag(model, 'delete', tag, doc_pattern)
rmitag(model, 'replace', tag, new_tag)
rmitag(model, 'replace', tag, new_tag, doc_pattern)
rmitag(model, 'clear', tag)
rmitag(model, 'clear', tag, doc_pattern)
```

## Description

`rmitag(model, 'list')` lists all user tags in `model`.

`rmitag(model, 'add', tag)` adds `tag` as a user tag for all requirements links in `model`.

`rmitag(model, 'add', tag, doc_pattern)` adds `tag` as a user tag for all links in `model`, where the full or partial document name matches the regular expression `doc_pattern`.

`rmitag(model, 'delete', tag)` removes the user tag, `tag` from all requirements links in `model`.

`rmitag(model, 'delete', tag, doc_pattern)` removes the user tag, `tag`, from all requirements links in `model`, where the full or partial document name matches `doc_pattern`.

`rmitag(model, 'replace', tag, new_tag)` replaces `tag` with `new_tag` for all requirements links in `model`.

`rmitag(model, 'replace', tag, new_tag, doc_pattern)` replaces `tag` with `new_tag` for links in `model`, where the full or partial document name matches the regular expression `doc_pattern`.

`rmitag(model, 'clear', tag)` deletes all requirements links that have the user tag, `tag`.

`rmitag(model, 'clear', tag, doc_pattern)` deletes all requirements links that have the user tag, `tag`, and link to the full or partial document name specified in `doc_pattern`.

## Input Arguments

**model**

Name of or handle to Simulink or Stateflow model with which requirements are associated.

**tag**

Character vector specifying user tag for requirements links.

**doc_pattern**

Regular expression to match in the linked requirements document name. Not case sensitive.

**new_tag**

Character vector that indicates the name of a user tag for a requirements link. Use this argument when replacing an existing user tag with a new user tag.

## Examples

Open the `slvnvdemo_fuelsys_officereq` example model, and add the user tag `tmptag` to all objects with requirements links:

```
open_system('slvnvdemo_fuelsys_officereq');
rmitag(gcs, 'add', 'tmptag');
```

Remove the user tag `test` from all requirements links:

```
open_system('slvnvdemo_fuelsys_officereq');
rmitag(gcs, 'delete', 'test');
```

Delete all requirements links that have the user tag `design`:

```
open_system('slvnvdemo_fuelsys_officereq');
rmitag(gcs, 'clear', 'design');
```

Change all instances of the user tag `tmptag` to `safety requirement`, where the document filename extension is `.docx`:

```
open_system('slvnvdemo_fuelsys_officereq');
rmitag(gcs, 'replace', 'tmptag', ...
       'safety requirements', '\.docx');
```

# See Also

rmi | rmidocrename

## Topics

"User Tags and Requirements Filtering"

**Introduced in R2010a**

# RptgenRMI.doorsAttribs

IBM Rational DOORS attributes in requirements report

## Syntax

RptgenRMI.doorsAttribs (action,attribute)

## Description

RptgenRMI.doorsAttribs (action,attribute) specifies which DOORS object attributes to include in the generated requirements report.

## Input Arguments

**action**

Character vector that specifies the desired action for what content to include from a DOORS record in the generated requirements report. Valid values for this argument are as follows.

| Value | Description |
|---|---|
| 'default' | Restore the default settings for the DOORS system attributes to include in the report.<br><br>The default configuration includes the **Object Heading** and **Object Text** attributes, and all other attributes, except:<br><br>• **Created Thru**<br>• System attributes with empty string values<br>• System attributes that are false |
| 'show' | Display the current settings for the DOORS attributes to include in the report. |

| Value | Description |
|---|---|
| `'type'` | Include or omit groups of DOORS attributes from the report.<br><br>If you specify `'type'` for the first argument, valid values for the second argument are:<br><br>• `'all'` — Include all DOORS attributes in the report.<br>• `'user'` — Include only user-defined DOORS in the report.<br>• `'none'` — Omit all DOORS attributes from the report. |
| `'remove'` | Omit specified DOORS attributes from the report. |
| `'all'` | Include specified DOORS attributes in the report, even if that attribute is currently excluded as part of a group. |
| `'nonempty'` | Enable or disable the empty attribute filter:<br><br>• Enter `RptgenRMI.doorsAttribs('nonempty', 'off')` to omit all empty attributes from the report.<br>• Enter `RptgenRMI.doorsAttribs('nonempty', 'on')` to include empty user-defined attributes. The report never includes empty system attributes. |

**Default:**

**attribute**

Character vector that qualifies the `action` argument.

## Output Arguments

**result**

- True if `RptgenRMI.doorsAttribs` modifies the current settings.
- For `RptgenRMI.doorsAttribs('show')`, this argument is a cell array of character vectors that indicate which DOORS attributes to include in the requirements report, for example:

```
>> RptgenRMI.doorsAttribs('show')

ans =
```

```
'Object Heading'
'Object Text'
'$AllAttributes$'
'$NonEmpty$'
'-Created Thru'
```

- The **Object Heading** and **Object Text** attributes are included by default.
- `'$AllAttributes$'` specifies to include all attributes associated with each DOORS object.
- `'$Nonempty$'` specifies to exclude all empty attributes.
- `'-Created Thru'` specifies to exclude the **Created Thru** attribute for each DOORS object.

# Examples

Limit the DOORS attributes in the requirements report to user-defined attributes:

```
RptgenRMI.doorsAttribs('type', 'user');
```

Omit the content of the **Last Modified By** attribute from the requirements report:

```
RptgenRMI.doorsAttribs('remove', 'Last Modified By');
```

Include the content of the **Last Modified On** attribute in the requirements report, even if system attributes are not included as a group:

```
RptgenRMI.doorsAttribs('add', 'Last Modified On');
```

Include empty system attributes in the requirements report:

```
RptgenRMI.doorsAttribs('nonempty', 'off');
```

Omit the **Object Heading** attribute from the requirements report. Use this option when the link label is always the same as the **Object Heading** for the target DOORS object and you do not want duplicate information in the requirements report:

```
RptgenRMI.doorsAttribs('remove', 'Object Heading');
```

## See Also

`rmi`

**Introduced in R2011b**

# slwebview_req

Export Simulink system to Web views with requirements

## Syntax

```
filename = slwebview_req(sysname)
filename = slwebview_req(sysname,Name,Value)
```

## Description

`filename = slwebview_req(sysname)` exports the system `sysname` and its children to a web page `filename` with contextual requirements information for the system displayed on a separate panel of the layered model structure Web view.

`filename = slwebview_req(sysname,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

**Note** You can use `slwebview_req` only if you have also installed Simulink Report Generator™.

## Examples

### Export All Layers

Export all the layers (including libraries and masks) from the system `gcs` to the file `filename`

```
filename = slwebview_req(gcs, 'LookUnderMasks', 'all',
'FollowLinks', 'on')
```

# Input Arguments

**sysname — The system to export to a Web view file**
character vector containing the path to the system | handle to a subsystem or block diagram | handle to a chart or subchart

Exports the specified system or subsystem and its child systems to a Web view file, with contextual requirements information for the system displayed on a separate panel of the layered model structure Web view. By default, child systems of the `sysname` system are also exported. Use the `SearchScope` name-value pair to export other systems, in relation to `sysname`.

Example: 'sysname'

## Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (' '). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example:

**SearchScope — Systems to export, relative to the sysname system**
'CurrentAndBelow' (default) | 'Current' | 'CurrentAndAbove' | 'All'

'CurrentAndBelow' exports the Simulink system or the Stateflow chart specified by `sysname` and all systems or charts that it contains.

'Current' exports only the Simulink system or the Stateflow chart specified by `sysname`.

'CurrentAndAbove' exports the Simulink system or the Stateflow chart specified by the `sysname` and all systems or charts that contain it.

'All' exports all Simulink systems or Stateflow charts in the model that contains the system or chart specified by `sysname`.

Data Types: `char`

### LookUnderMasks — Specifies whether to export the ability to interact with masked blocks
`'none'` (default) | `'all'`

`'none'` does not export masked blocks in the Web view. Masked blocks are included in the exported systems, but you cannot access the contents of the masked blocks.

`'all'` exports all masked blocks.

Data Types: `char`

### FollowLinks — Specifies whether to follow links into library blocks
`'off'` (default) | `'on'`

`'off'` does not allow you to follow links into library blocks in a Web view.

`'on'` allows you to follow links into library blocks in a Web view.

Data Types: `char`

### FollowModelReference — Specifies whether to access referenced models in a Web view
`'off'` (default) | `'on'`

`'off'` does not allow you to access referenced models in a Web view.

`'on'` allows you to access referenced models in a Web view.

Data Types: `char`

### ViewFile — Specifies whether to display the Web view in a Web browser when you export the Web view
`'on'` (default) | `'off'`

`'on'` displays the Web view in a Web browser when you export the Web view.

`'off'` does not display the Web view in a Web browser when you export the Web view.

Data Types: `char`

### ShowProgressBar — Specifies whether to display the status bar when you export a Web view
'on' (default) | 'off'

`'on'` displays the status bar when you export a Web view.

`'off'` does not display the status bar when you export a Web view.

Data Types: `char`

# Output Arguments

### filename — The name of the HTML file for displaying the Web view
character vector

Reports the name of the HTML file for displaying the Web view. Exporting a Web view creates the supporting files, in a folder.

# Tips

A Web view is an interactive rendition of a model that you can view in a Web browser. You can navigate a Web view hierarchically to examine specific subsystems and to see properties of blocks and signals.

You can use Web views to share models with people who do not have Simulink installed.

Web views require a Web browser that supports Scalable Vector Graphics (SVG).

# See Also
slwebview_cov

**Introduced in R2015a**

# Classes — Alphabetical List

# slreq.Link

Work with link objects

# Description

When you establish a traceable association between artifacts, Simulink Requirementscreates an `slreq.Link` object to store source and destination data of the link.

# Construction

`link = slreq.createLink(src, dest)` creates an `slreq.Link` object `link` with source and destination artifacts specified by `src` and `dest` respectively. The `slreq.Link` object is stored in the Link set file that belongs to `src`.

`outLinks = slreq.outLinks(src)` returns an array of `slreq.Link` objects `outLinks` that contains the outgoing links from the source artifact `src`.

`inLinks = slreq.inLinks(dest)` returns an array of `slreq.Link` objects `inLinks` that contains the incoming links to the destination artifact `dest`.

## Input Arguments

### `src` — Link source artifact
structure

Link source artifact, specified as a MATLAB structure.

### `dest` — Link destination artifact
structure

Link destination artifact, specified as a MATLAB structure.

## Output Arguments

**link — Link object**
slreq.Link object

Handle to a link, returned as an slreq.Link object.

**outLinks — Outgoing links**
slreq.Link object array

Array of outgoing links.

**inLinks — Incoming links**
slreq.Link object array

Array of incoming links.

# Properties

**CreatedOn — Date link was created**
datetime value

The date on which the link was created, specified as a datetime value. The software populates this property.

**CreatedBy — Link creator**
character vector

The name of the individual or organization who created the link.

**ModifiedOn — Date link was modified**
datetime value

The date on which the link was last modified, specified as a datetime value. The software populates this property.

**ModifiedBy — Link modifier**
character vector

The name of the individual or organization who last modified the link.

**Comments — Link comments**
structure array

The comments that are attached with the link, returned as a structure.

**Type — Link type enumeration**
`'Implement'` | `'Verify'` | `'Relate'` | `'Derive'` | `'Refine'`

The relationship between the source and the destination artifacts. For more information, see "Link Types".

**Description — Link description**
character vector

Link descriptive text, specified as a multi-line character vector.

**Keywords — Link keywords**
character array

Link keywords, specified as character array.

**Rationale — Link rationale**
character vector

Link rationale text, specified as a multiline character vector.

# Methods

| | |
|---|---|
| destination | Get link destination artifact |
| LinkSet | Return parent link set |
| source | Get link source artifact |

# Examples

## Create Links

```
% Create a link between the current Simulink Object and a requirement
link1 = slreq.createLink(gcb, REQ)
```

```
link1 =

  Link with properties:

          Type: 'Implement'
   Description: 'Plant Specs'
      Keywords: [0×0 char]
      Rationale: ''
      CreatedOn: 02-Sep-2017 15:49:28
      CreatedBy: 'Jane Doe'
     ModifiedOn: 21-Oct-2017 11:34:12
     ModifiedBy: 'John Doe'
       Comments: [0×0 struct]

% Create a link between a requirement and the current Stateflow object
link2 = slreq.createLink(REQ, sfgco);
```

## Get Incoming Links

```
% Get the handle to a requirements set
myReqSet = slreq.find('Type', 'ReqSet', 'Name', 'Design_Spec');

% Get the handle to a requirement in myReqSet
myReq = find(myReqSet, 'Type', 'Requirement', 'Id', 'R1.1');

% Query incoming links to myReq
inLinks = slreq.inLinks(myReqs);
```

## Get Outgoing Links

```
% Load a link set and get link sources
myLinkSet = slreq.load('fuelsys.slx');
allSrcs = myLinkSet.sources();

% Get outgoing links
myLinks1 = slreq.outLinks(allSrcs(1));
```

# See Also
slreq.LinkSet | slreq.createLink

**Introduced in R2018a**

# slreq.LinkSet

Work with link sets

# Description

Instances of `slreq.LinkSet` are Link Set objects. Links are organized in Link Sets. Each Link Set is associated with a source artifact such as a Simulink model or a data dictionary and is serialized into a separate file which stores the links associated with it. The default location and name of the Link set file matches that of the source artifact.

# Construction

`allLinkSets = slreq.find('Type', 'LinkSet')` finds and returns an array of all loaded `slreq.LinkSet` objects `allLinkSets`.

`myLinkSet = slreq.find('Type', 'LinkSet', 'Name', ArtifactName)` finds and returns an `slreq.LinkSet` object `myLinkSet` matching the artifact name specified by `ArtifactName`.

`myLinkSet = slreq.load(ArtifactName)` loads an `slreq.LinkSet` object `myLinkSet` matching the artifact name specified by `ArtifactName`.

## Input Arguments

**ArtifactName — Link set artifact name**
character vector

The name of the link set artifact, specified as a character vector.

## Output Arguments

**allLinkSets — Link sets**
`slreq.LinkSet` array

Array of all loaded link sets.

**myLinkSet — Link set**
slreq.LinkSet object

Link set, returned as an slreq.LinkSet object.

# Properties

**Filename — Link set file path**
character vector

The file path of the link set, specified as a character vector.

**Artifact — Container identifier**
character vector

Top-level container identifier, such as a Microsoft Office document name, an IBM Rational DOORS Module unique ID, Simulinkmodel name, or Simulink Test™ Test Manager file name.

**Domain — Link set custom link type**
character vector

The custom link type of the links in the link set. For more information, see "Custom Link Types".

Example: linktype_rmi_excel, linktype_rmi_doors

**Revision — Link set revision number**
scalar

Link set revision number, specified as a scalar.

**Dirty — Unsaved changes indicator**
0 | 1

Indicates if the link set has unsaved changes. 0 for no unsaved changes and 1 for unsaved changes.

**Description — Link set description**
character vector

Link set description text, specified as a character vector.

## Methods

| save | Save link set |
| sources | Get link sources |

## Examples

```
% Find a link set
myLinkSet1 = slreq.find('Type', 'LinkSet', 'Name', 'Project_req')

myLinkSet1 =

  LinkSet with properties:

    Description: ''
       Filename: 'Project_req.slmx'
       Artifact: 'Project_req.slreqx'
         Domain: 'linktype_rmi_slreq'
       Revision: 2
          Dirty: 0

myLinkSet2 = slreq.load('fuelsys.slx')

myLinkset2 =

  LinkSet with properties:

    Description: ''
       Filename: 'C:\MATLAB\My_Files\fuelsys_linkset.slmx'
       Artifact: 'D:\Work\Design_Specs\fuelsys.slx'
         Domain: 'linktype_rmi_simulink'
       Revision: 2
          Dirty: 0

% Set the link set description
myLinkset2.Description = 'Link set for the fuel system'

myLinkset2 =

  LinkSet with properties:
```

```
Description: 'Link set for the fuel system'
   Filename: 'C:\MATLAB\My_Files\fuelsys_linkset.slmx'
   Artifact: 'D:\Work\Design_Specs\fuelsys.slx'
     Domain: 'linktype_rmi_simulink'
   Revision: 2
      Dirty: 1
```

## See Also

`rmimap.map` | `slreq.Link`

**Introduced in R2018a**

# slreq.Reference

Work with external requirement proxy objects

## Description

Instances of `slreq.Reference` are proxies for external requirement objects that a third-party external application manages and maintains. Referenced requirement objects are read-only but can be synchronized from an external application and can exist only within a requirements set.

## Construction

`ref = find(rs, 'Type', 'Reference', 'PropertyName', PropertyValue)` finds and returns a referenced requirement or a set of referenced requirements `ref` in the requirements set `rs` specified by the properties matching `PropertyName` and `PropertyValue`.

`ref = add(rs, 'Artifact', FileName, 'PropertyName', PropertyValue)` adds a referenced requirement `ref` to a requirements set `rs` which references requirements from the external document specified by `FileName` with properties and custom attributes specified by `PropertyName` and `PropertyValue`.

### Input Arguments

**rs — Requirements set object**
`slreq.ReqSet` object

Requirements set file, specified as an `slreq.ReqSet` object.

**FileName — Container identifier**
character vector

File name for a top-level container identifier, such as a Microsoft Office document name or an IBM Rational DOORS Module unique ID.

## Output Arguments

### `ref` — Referenced requirement
`slreq.Reference` object

Referenced requirement, specified as an `slreq.Reference` object.

# Properties

### `Artifact` — Container identifier
character vector

Top-level container identifier, like a Microsoft Office document name or an IBM Rational DOORS Module unique ID.

### `Id` — Reference ID
character vector

Reference ID, returned as a character vector.

### `Summary` — Referenced requirement summary
character vector

Referenced requirement summary text, returned as a character vector.

### `Description` — Referenced requirement description
character vector

Referenced requirement description text, returned as a multiline character vector.

### `SID` — Referenced requirement Session Independent Identifier
character vector

The Session Independent Identifier corresponding to the referenced requirement.

### `Domain` — Requirements document custom link type
character vector

The custom link type of the requirements document. For more information, see "Custom Link Types".

Example: `'linktype_rmi_doors'`, `'linktype_rmi_excel'`

**SynchronizedOn — Date and time referenced requirement was last synchronized**
datetime value

The date and time the referenced requirement was last synchronized with the external document, specified as a datetime value. The software automatically populates this property.

**ModifiedOn — Date referenced requirement was modified**
datetime value

The date the referenced requirement was last modified, specified as a datetime value. The software automatically populates this property.

# Methods

| | |
|---|---|
| add | Add referenced requirements |
| children | Find children references |
| find | Find referenced requirements |
| getAttribute | Get referenced requirement attributes |
| parent | Find parent item of referenced requirement |
| reqSet | Return parent requirements set |
| setAttribute | Set referenced requirement custom attributes |

# Examples

## Get the Handle to a Referenced Requirement

```
% Find a referenced requirement with Id R9 in a requirements set rs
ref = find(rs, 'Type', 'Reference', 'Id', 'R9')

ref =

  Reference with properties:

        Keywords: [0×0 char]
        Artifact: 'Req_doc.docx'
```

```
           Id: 'R9'
      Summary: 'System overview'
  Description: ''
          SID: 3
       Domain: 'linktype_rmi_word'
SynchronizedOn: 25-Jul-2017 11:34:02
```

## See Also

slreq.ReqSet | slreq.Requirement | slreq.import

**Introduced in R2018a**

# slreq.ReqSet

Work with Requirements sets

# Description

Instances of `slreq.ReqSet` are Requirement Set objects.

# Construction

`newReqSet = slreq.new(reqSetName)` creates a requirement set named `reqSetName` in the current working folder.

`newReqSet = slreq.new(reqSetPath)` creates a requirement set on the specified path.

## Input Arguments

**reqSetName — Requirement set name**
character vector

Name of the requirement set, specified as a character vector.

Example: `'Design Requirements'`

**reqSetPath — Requirement set file name and path**
character vector

The file name and path of the requirement set, specified as a character vector.

Example: `'C:\MATLAB\myReqSet.slreqx'`

## Output Arguments

**newReqSet — Requirements set**
slreq.ReqSet object

An instance of the `slreq.ReqSet` object.

# Properties

**Name — Requirements set name**
character vector

Name of the requirements set, specified as a character vector.

**Filename — Requirements set file path**
character vector

The file path of the requirements set, specified as a character vector.

**Revision — Requirements set revision number**
scalar

Requirements set revision number, specified as a scalar.

**CreatedBy — Requirements set creator**
character vector

The name of the individual or organization who created the requirements set.

**CreatedOn — Date requirements set was created**
datetime value

The date the requirements set was created, specified as a `datetime` value. The software automatically populates this property.

**ModifiedBy — Requirements set modifier**
character vector

The name of the individual or organization who last modified the requirements set.

**ModifiedOn — Date requirements set was modified**
datetime value

The date the requirements set was last modified, specified as a `datetime` value. The software automatically populates this property.

**Description — Requirements set description**
character vector

Requirements set description text, specified as a character vector.

**Dirty — Unsaved changes indicator**
0 | 1

Indicates if the requirements set has unsaved changes. 0 for no unsaved changes, and 1 for unsaved changes.

**CustomAttributeNames — Custom attributes associated with the requirements set**
cell array of character vectors

Requirements set custom attribute names, specified as a cell array of character vectors.

# Methods

| | |
|---|---|
| close | Close a requirements set |
| createReferences | Create read-only references to requirement items in third-party documents |
| find | Find requirements in requirements set that have matching attribute values |
| importFromDocument | Import editable requirements from external documents |
| save | Save a requirements set |

# Examples

## Create and Instantiate a Requirements Set Object

```
% Create a new requirements set
rs = slreq.new('My_Requirements_Set');

% Save and close the requirements set - saving creates a .slreqx file
save(rs);
close(rs);
```

```
% Load an existing requirements set
rs1 = slreq.load('Design_Specifications');

% Open the requirements set in the Requirements Editor
slreq.open(rs1);
```

## See Also

slreq.Link | slreq.LinkSet | slreq.Reference | slreq.Requirement

**Introduced in R2018a**

# slreq.Requirement

Work with Requirement objects

## Description

Instances of `slreq.Requirement` are Requirement objects that you manage solely inside Simulink Requirements and that do not have a persistent association with artifacts managed by external applications. Requirement objects can exist only within a requirements set.

## Construction

`req = find(rs, 'PropertyName', PropertyValue)` finds and returns a requirement `req` in the requirements set `rs` with additional requirement properties specified by `PropertyName` and `PropertyValue`.

`req = add(rs, 'PropertyName', PropertyValue)` adds a requirement `req` to the requirement set `rs` with additional requirement properties specified by `PropertyName` and `PropertyValue`.

### Input Arguments

**rs — Requirements set object**
`slreq.ReqSet` object

Requirements set file, specified as an `slreq.ReqSet` object.

### Output Arguments

**req — Requirement object**
`slreq.Requirement` object

Handle to a requirement, returned as an `slreq.Requirement` object.

## Properties

**`Id` — Requirement custom ID**
character vector

Custom ID of the requirement, returned as a character vector. You cannot use spaces and `'#'` in custom IDs.

**`Summary` — Requirement summary**
character vector

Requirement summary text, specified as a one-line, plain text character vector.

**`Keywords` — Requirement keywords**
character array

Requirement keywords, specified as character array.

**`Description` — Requirement description**
character vector

Requirement description text, specified as a multiline character vector.

**`Rationale` — Requirement rationale**
character vector

Requirement rationale text, specified as a multiline character vector.

**`SID` — Requirement Session Independent Identifier**
character vector

The Session Independent Identifier corresponding to the requirement.

**`CreatedOn` — Date requirement was created**
datetime value

The date on which the requirement was created, specified as a `datetime` value. The software populates this property.

**`CreatedBy` — Requirement creator**
character vector

The name of the individual or organization who created the requirement.

**ModifiedOn — Date requirement was modified**
datetime value

The date on which the requirement was last modified, specified as a `datetime` value. The software populates this property.

**ModifiedBy — Requirement modifier**
character vector

The name of the individual or organization who last modified the requirement.

**FileRevision — Requirement revision number**
scalar

Requirement revision number, specified as a scalar.

**Dirty — Unsaved changes indicator**
0 | 1

Indicates if the requirement has unsaved changes. `0` for no unsaved changes and `1` for unsaved changes.

**Comments — Requirement comments**
structure array

The comments that are attached with the requirement, returned as a structure.

# Methods

| | |
|---|---|
| add | Add requirement to requirements set |
| children | Find child requirements of a requirement |
| demote | Demote requirements |
| find | Find requirements that have matching attribute values |
| getAttribute | Get requirement attributes |
| parent | Find parent item of requirement |
| promote | Promote requirements |
| reqSet | Return parent requirements set |
| setAttribute | Set requirement attributes |

# Examples

## Find a Requirement in a Requirements Set

```
% Find a requirement with ID 77 in a requirements set rs
req = find(rs, 'Type', 'Requirement', 'ID', '77');

req =

  Requirement with properties:

              Id: '77'
         Summary: 'Test Spec'
        Keywords: [0×0 char]
     Description: ''
       Rationale: ''
             SID: 80
       CreatedBy: 'John Doe'
       CreatedOn: 05-Oct-2007 16:09:38
      ModifiedBy: 'Jane Doe'
      ModifiedOn: 21-Dec-2016 11:10:05
        Comments: [0×0 struct]
```

## See Also

slreq.Link | slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

**3**

# Methods — Alphabetical List

# destination

Get link destination artifact

## Syntax

```
dest = destination(myLink)
```

## Description

`dest = destination(myLink)` returns the source artifact `dest` of the link `myLink`.

## Input Arguments

**myLink — Link object**
`slreq.Link` object

Link, specified as an `slreq.Link` object.

## Output Arguments

**dest — Destination artifact**
`struct`

The link destination artifact, returned as a MATLAB structure.

## Examples

### Get Link Destination

```
% Load a requirement set file and select one link
rs = slreq.load('C:\MATLAB\My_Req_Set.slreqx');
```

```
allReqs = find(rs, 'Type', 'Requirement');
req = allReqs(1);
allIncomingLinks = inLinks(req);
myLink = allIncomingLinks(1);

% Get link destination
myDestination = destination(myLink)

myDestination =

  struct with fields:

     reqSet: 'My_Req_Set'
     domain: 'linktype_rmi_slreq'
    summary: 'My Requirement'
    details: ''
         id: ''
        sid: 12
```

## See Also
LinkSet | slreq.Link | source

**Introduced in R2018a**

# LinkSet

Return parent link set

## Syntax

```
lks = linkSet(myLink)
```

## Description

`lks = linkSet(myLink)` returns the parent link set `lks` to which the link `myLink` belongs.

## Input Arguments

**myLink — Link object**
slreq.Link object

Link, specified as an `slreq.Link` object.

## Output Arguments

**lks — Parent link set**
slreq.LinkSet object

Parent link set of the link `myLink`, returned as an `slreq.LinkSet` object.

## Examples

### Query Link Set Information

```
% Load a requirement set file and select one requirement
rs = slreq.load('C:\MATLAB\My_Req_Set.slreqx');
```

```
allReqs = find(rs, 'Type', 'Requirement');
req = allReqs(1);

% Find the incoming links that belong to req
allInLinks = inLinks(req);

% Query link set information
myParentLinkSet = linkSet(allInLinks)

myParentLinkSet =

  LinkSet with properties:

    Description: ''
       Filename: 'model_controller.slmx'
       Artifact: 'model_controller.slx'
         Domain: 'linktype_rmi_simulink'
       Revision: 4
          Dirty: 0
```

## See Also
destination | slreq.Link | source

**Introduced in R2018a**

# source

Get link source artifact

## Syntax

```
src = source(myLink)
```

## Description

`src = source(myLink)` returns the source artifact `src` of the link `myLink`.

## Input Arguments

**myLink — Link object**
slreq.Link object

Link, specified as an `slreq.Link` object.

## Output Arguments

**src — Source artifact**
struct

The link source artifact, returned as a MATLAB structure.

## Examples

### Get Link Source

```
% Load a requirement set file and select one link
rs = slreq.load('C:\MATLAB\My_Req_Set.slreqx');
```

```matlab
allReqs = find(rs, 'Type', 'Requirement');
req = allReqs(1);
allIncomingLinks = inLinks(req);
myLink = allIncomingLinks(1);

% Get link source
mySource = source(myLink)

mySource =

  struct with fields:

      domain: 'linktype_rmi_simulink'
    artifact: 'controller_model.slx'
          id: ':241'
```

## See Also

LinkSet | destination | slreq.Link

**Introduced in R2018a**

# save

Save link set

## Syntax

```
save(lks)
save(lks, filePath)
```

## Description

save(lks) saves the link set lks by using its file name.

save(lks, filePath) saves the link set lks and updates its Name and Filename properties.

## Input Arguments

**lks — Link set file**
slreq.LinkSet object

Link set file, specified as an slreq.LinkSet object.

**filePath — File name and path**
character vector

The file name and path of the link set, specified as a character vector.

Example: 'C:\MATLAB\myLinkSet.slmx'

# Examples

## Save Link Set File

```
% Load a link set file
myLinkSet = slreq.load(get_param('fuelsys', 'Name'));

% Save the link set file
save(myLinkSet);

% Save the link set file by another name
save(myLinkSet, 'C:\MATLAB\Files\MyLinkSet1.slmx');
```

# See Also
slreq.LinkSet | sources

**Introduced in R2018a**

# sources

Get link sources

## Syntax

```
linkSetSources = sources(lks)
```

## Description

`linkSetSources = sources(lks)` returns an array of structures `linkSetSources` that contains the link sources of all the links in the link set `lks`.

## Input Arguments

**lks — Link set**
`slreq.LinkSet` object

Instance of an `slreq.LinkSet` object.

## Output Arguments

**linkSetSources — Link set sources**
structure

Link set source data, returned as a MATLAB structure.

## Examples

### Get Link Sources

```
% Load a link set and get link sources
myLinkSet = slreq.load('fuelsys.slx');
```

```
mySources = sources(myLinkSet)

mySources =

  1×16 struct array with fields:

    domain
    artifact
    id
```

## See Also
save | slreq.LinkSet

**Introduced in R2018a**

# add

Add referenced requirements

## Syntax

```
refNew = add(rs, 'Artifact',FileName,'PropertyName',PropertyValue)
refChild = add(ref,'Artifact',FileName,'PropertyName',PropertyValue)
```

## Description

`refNew = add(rs, 'Artifact',FileName,'PropertyName',PropertyValue)` adds a referenced requirement `refNew` to a requirements set `rs` which references requirements from the external document specified by `FileName` with properties and custom attributes specified by `PropertyName` and `PropertyValue`.

`refChild = add(ref,'Artifact',FileName,'PropertyName',PropertyValue)` adds a referenced child requirement `refChild` to a referenced requirement `ref` which references requirements from the external document specified by `FileName` with properties and custom attributes specified by `PropertyName` and `PropertyValue`.

## Input Arguments

**rs — Requirements set file**
`slreq.ReqSet` object

Requirements set file, specified as an `slreq.ReqSet` object.

**ref — Referenced requirement**
`slreq.Reference` object

Referenced requirement, specified as an `slreq.Reference` object.

**FileName — Container identifier**
character vector

File name for a top-level container identifier, such as a Microsoft Office document name or an IBM Rational DOORS Module unique ID.

## Output Arguments

**refNew — Referenced requirement**
slreq.Reference object

The referenced requirement that was added, returned as an slreq.Reference object.

**refChild — Referenced child requirement**
slreq.Reference object

The referenced child requirement that was added, returned as an slreq.Reference object.

## Examples

### Add a Referenced Requirement

```
% Load a requirements set file

rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');

% The parent external document for rs is Req_doc.docx
% Add a top-level referenced requirement to rs
newRef1 = add(rs, 'Artifact', 'crs_req.docx', 'Id', '5.0', 'Summary', 'Additional Requi

% Add a child referenced requirement to newRef1
newRef2 = add(newRef1, 'Artifact', 'crs_req.docx', 'Id', '5.1', 'Summary', 'Additional
```

## See Also
slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

# children

Find children references

## Syntax

```
childRefs = children(ref)
```

## Description

`childRefs = children(ref)` returns the child referenced requirements `childRefs` of the `slreq.Reference` object `ref`.

## Input Arguments

**ref — Referenced requirement instance**
`slreq.Reference` object

Reference to a requirement specified as an `slreq.Reference` object.

## Output Arguments

**childRef — Child references**
`slreq.Reference` object | `slreq.Reference` object array

The child referenced requirements belonging to the referenced requirement `ref`, returned as `slreq.Reference` objects.

# Examples

## Find Child References

```
% Load a requirements set file and find referenced requirements
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
allRefs = find(rs, 'Type', 'Reference')

allRefs =

  1×32 Reference array with properties:

    Keywords
    Artifact
    Id
    Summary
    Description
    SID
    Domain
    SynchronizedOn
    ModifiedOn

ref1 = allRefs(1);

% Find the children of ref1
childRef = children(ref1)

childRef =

  Reference with properties:

          Keywords: [0×0 char]
          Artifact: 'Req_doc.docx'
                Id: 'R1.1'
           Summary: 'References'
       Description: ''
               SID: 2
            Domain: 'linktype_rmi_word'
    SynchronizedOn: 26-Jul-2015 15:45:22
        ModifiedOn: 27-Jul-2015 12:00:13
```

## See Also

parent | slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

# find

Find referenced requirements

## Syntax

```
refs = find(rs, 'Type', 'Reference', 'PropertyName', PropertyValue)
```

## Description

`refs = find(rs, 'Type', 'Reference', 'PropertyName', PropertyValue)` finds and returns a referenced requirement or a set of referenced requirements `refs` in the requirements set `rs` specified by the properties matching `PropertyName` and `PropertyValue`.

## Input Arguments

**rs — Requirements set**
slreq.ReqSet object

Requirements set specified as an `slreq.ReqSet` object.

## Output Arguments

**refs — Referenced requirements**
slreq.Reference object | slreq.Reference object array

Referenced requirements, returned as `slreq.Reference` objects.

## Examples

### Find Referenced Requirements That Have Matching Attribute Values

```matlab
% Load a requirements set file
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');

% Find all referenced requirements with Id R10 in the requirements set
matchedRefs = find(rs, 'Type', 'Reference', 'Id', 'R10')

matchedRefs =

  Reference with properties:

          Keywords: [0×0 char]
          Artifact: 'Req_doc.docx'
                Id: 'R10'
           Summary: 'System overview'
       Description: ''
               SID: 3
            Domain: 'linktype_rmi_word'
      SynchronizedOn: 23-Jul-2017 12:47:23
```

## See Also

slreq.Reference | slreq.ReqSet | slreq.find

**Introduced in R2018a**

# getAttribute

Get referenced requirement attributes

## Syntax

```
val = getAttribute(ref, propertyName)
```

## Description

`val = getAttribute(ref, propertyName)` gets a referenced requirement property.

## Input Arguments

**ref — Referenced requirement instance**
slreq.Reference object

Reference to a requirement specified as an `slreq.Reference` object.

**propertyName — Referenced requirement property**
character vector

Referenced requirement property name.

Example: `'SID', 'CreatedOn', 'Summary'`

## Examples

### Get Referenced Requirement Attributes

```
% Load a requirements set file and get the handle to
% one referenced requirement

rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
```

```
ref1 = find(rs, 'Type', 'Reference', 'Id', 'R10.1');

% Get the Summary of ref1
summaryRef1 = getAttribute(ref1, 'Summary')

summaryRef1 =

  'Control Algorithm'
```

## See Also
setAttribute | slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

# parent

Find parent item of referenced requirement

## Syntax

```
parentObj = parent(ref)
```

## Description

`parentObj = parent(ref)` returns the parent object `parentObj` of the `slreq.Reference` object `req`.

## Input Arguments

**ref — Referenced requirement instance**
`slreq.Reference` object

Referenced requirement specified as an `slreq.Reference` object.

## Output Arguments

**parentObj — Parent object**
`slreq.Reference` object | `slreq.ReqSet` object

The parent of the referenced requirement `ref`, returned as an `slreq.Reference` object or as an `slreq.ReqSet` object.

# Examples

## Find Parent References

```matlab
% Load a requirements set file and find referenced requirements
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
refs = find(rs, 'Type', 'Reference')

refs =

  1×32 Reference array with properties:

    Keywords
    Artifact
    Id
    Summary
    Description
    SID
    Domain
    SynchronizedOn
    ModifiedOn

% Find the parent of the first reference element
parentRef1 = parent(refs(1));

parentRef1 =

  ReqSet with properties:

              Description: ''
                     Name: 'My_Requirements_Set_1'
                 Filename: 'C:\MATLAB\My_Requirements_Set_1.slreqx'
                 Revision: 6
                    Dirty: 1
    CustomAttributeNames: {}
```

# See Also
children | slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

# reqSet

Return parent requirements set

## Syntax

```
rsout = reqSet(ref)
```

## Description

`rsout = reqSet(ref)` returns the parent requirements set `rsout` to which the referenced requirement `ref` belongs.

## Input Arguments

**ref — Referenced requirement object**
slreq.Reference object

Referenced requirement, specified as a `slreq.Reference` object.

## Output Arguments

**rsout — Parent requirements set**
slreq.ReqSet object

The parent requirements set of the referenced requirement `ref`, returned as an `slreq.ReqSet` object.

## Examples

### Query Requirements Set Information

```matlab
% Load a new requirements set file and select one referenced requirement
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
allRefs = find(rs,'Type','Reference');
ref = allRefs(1);

% Query which requirements set ref belongs to
reqSet(ref)

ans =

  ReqSet with properties:

              Description: ''
                     Name: 'My_Requirements_Set_1'
                 Filename: 'C:\MATLAB\My_Requirements_Set_1.slreqx'
                 Revision: 65
                    Dirty: 0
      CustomAttributeNames: {}
```

## See Also
parent | slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

# setAttribute

Set referenced requirement custom attributes

## Syntax

setAttribute(ref, propertyName, propertyValue)

## Description

setAttribute(ref, propertyName, propertyValue) sets a referenced requirement property. Use this method to set the values of custom attributes that you define for your requirements set.

## Input Arguments

**ref — Referenced requirement instance**
slreq.Reference object

Referenced requirement specified as an slreq.Reference object.

**propertyName — Referenced requirement custom attribute**
character vector

Referenced requirement custom attribute name.

Example: 'Priority'

**propertyValue — Referenced requirement custom attribute value**
character vector

Referenced requirement custom attribute name, specified as a character vector.

Example: 'High', 'Medium'

## Examples

### Set Referenced Requirement Custom Attribute

```matlab
% Load a requirements set file and get the handle to one requirement
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
ref1 = find(rs, 'Type', 'Reference', 'ID', 'R20.1');

% Set the Priority of req1
setAttribute(ref1, 'Priority', 'Low');
```

## See Also

getAttribute | slreq.Reference | slreq.ReqSet

**Introduced in R2018a**

# close

Close a requirements set

## Syntax

```
close(rs)
```

## Description

`close(rs)` closes a requirements set.

## Input Arguments

**rs — Requirements set file**
`slreq.ReqSet` object

Requirements set file, specified as an `slreq.ReqSet` object.

## Examples

### Close a Requirement Set

```
% Create a new requirements set file
rs1 = slreq.new('C:\MATLAB\My_Requirements_Set_1.slreqx');

% Save the requirements set file
save(rs1);

% Close the requirements set file
close(rs1);
```

# See Also
`slreq.ReqSet`

**Introduced in R2018a**

# createReferences

Create read-only references to requirement items in third-party documents

## Syntax

```
createReferences(rs, pathToFile, Name, Value)
createReferences(rs, reqFormat, Name, Value)
```

## Description

`createReferences(rs, pathToFile, Name, Value)` creates read-only references to requirements content in an external document at `pathToFile` by using additional `Name, Value` arguments to specify import options.

`createReferences(rs, reqFormat, Name, Value)` creates read-only references to requirements content in an external document corresponding to the specified registered document type specified by `reqFormat` by using additional `Name, Value` arguments to specify import options.

## Input Arguments

**rs — Requirements set file**
`slreq.ReqSet` object

Requirements set file, specified as a `slreq.ReqSet` object.

**pathToFile — File path**
character vector

Path to the requirements document.

Example: `'C:\MATLAB\System_Requirements.docx'`

**reqFormat — Registered document type label**
character vector

Custom registered document type label that you create by using a Custom Document Type extension API.

Example: `'linktype_rmi_doors'`

## Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'columns', '[1 8]', 'RichText', true`

### ReqSet — Requirements Set
`slreq.ReqSet` object

The name of the existing requirements set that you import references to requirements into, specified as the comma-separated pair of `'ReqSet'` and a valid requirements set file name.

Example: `'ReqSet', 'My_Requirements_Set'`

### RichText — Requirements content imported as rich text
`false` (default) | `true`

Option to import requirements content as rich text, specified as the comma-separated pair consisting of `'RichText'` and `true` or `false`.

Example: `'RichText', true`

### bookmarks — Use custom bookmarks in Microsoft Word and Microsoft Excel
`true` | `false`

Option to use custom bookmarks in Microsoft Word documents and Microsoft Excel spreadsheets to import requirements content, specified as the comma-separated pair consisting of `'bookmarks'` and `true` or `false`.

Example: `'bookmarks', false`

### match — Regular expression
character vector

Import requirements by using regular expression pattern matching, specified as the comma-separated pair consisting of `'match'` and a regular expression pattern.

Example: `'match', '^REQ\d+'`

### `columns` — Range of columns
double array

Range of columns to import. This option is applicable only for Microsoft Excel spreadsheets.

Example: `'columns', [1 6]`

### `rows` — Range of rows
double array

Range of rows to import. This option is applicable only for Microsoft Excel spreadsheets.

Example: `'rows', [3 35]`

### `attributes` — Attribute names
cell array

Attribute names to import, specified as a cell array.

---

**Note** When importing requirements from a Microsoft Excel spreadsheet, the length of this cell array must match the number of columns that you specified for import by using the `'columns'` option.

---

Example: `'attributes', {'Test Status', 'Test Procedure'}`

### `idColumn` — ID Column
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **ID** field in the requirements set.

Example: `'idColumn', 1`

### `summaryColumn` — Summary Column
double

**3-31**

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Summary** field in the requirements set.

Example: `'summaryColumn', 4`

**keywordsColumn — Keywords Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Keywords** field in the requirements set.

Example: `'keywordsColumn', 3`

**descriptionColumn — Description Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Description** field in the requirements set.

Example: `'descriptionColumn', 2`

**rationaleColumn — Rationale Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Rationale** field in the requirements set.

Example: `'rationaleColumn', 5`

# Examples

## Create Read-Only References to Requirements in Microsoft Office Documents

```
% Create a new requirements set and save it

rs = slreq.new('newReqSet');
save(rs);

% Create read-only rich text references to requirements
% in a Word document
createReferences(rs, 'C:\Work\Requirements_Spec.docx', ...
```

```
'RichText', true);

% Create read-only plain text references to requirements
% in an Excel spreadsheet
createReferences(rs, 'C:\Work\Design_Spec.xlsx', ...
'columns', [2 6], 'rows', [3 32], 'idColumn', 2, ...
'summaryColumn', 3);
```

## See Also
slreq.Reference | slreq.ReqSet | slreq.import

**Introduced in R2018a**

# find

Find requirements in requirements set that have matching attribute values

## Syntax

```
myReq = find(rs, 'PropertyName', 'PropertyValue')
```

## Description

`myReq = find(rs, 'PropertyName', 'PropertyValue')` finds and returns an `slreq.Requirement` object myReq in the requirements set `rs` specified by the properties matching `PropertyName` and `PropertyValue`.

## Input Arguments

**rs — Requirements set file**
slreq.ReqSet object

Requirements set file, specified as a `slreq.ReqSet` object.

## Output Arguments

**myReq — Requirement object**
slreq.Requirement object

Requirement, returned as an `slreq.Requirement` object.

## Examples

### Find Requirements That Have Matching Attribute Values

```matlab
% Load a requirements set file
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');

% Find all editable requirements in the requirement set
allReqs = find(rs, 'Type', 'Requirement');

% Find all referenced requirements in the requirement set
allRefs = find(rs, 'Type', 'Reference');

% Find all requirements with a certain ID
matchedReqs = find(rs, 'ID', 'R1.1');
```

## See Also
slreq.ReqSet | slreq.find

**Introduced in R2018a**

# importFromDocument

Import editable requirements from external documents

## Syntax

```
importFromDocument(rs, pathToFile, Name, Value)
```

## Description

importFromDocument(rs, pathToFile, Name, Value) imports editable requirements with contents duplicated from an external document at `pathToFile` using by additional `Name, Value` arguments to specify import options.

## Input Arguments

**rs — Requirements set file**
slreq.ReqSet object

Requirements set file, specified as a `slreq.ReqSet` object.

**pathToFile — File path**
character vector

Path to the requirements document that you want to import editable requirements from.

Example: `'C:\MATLAB\System_Requirements.docx'`

**ReqSet — Requirements Set**
character vector

The name for the existing requirements set that you import requirements into, specified as a character vector.

Example: `'ReqSet', 'My_Requirements_Set'`

**`RichText` — Option to import rich text requirements**
`false` (default) | `true`

Option to import requirements as rich text, specified as a Boolean value.

Example: `'RichText', true`

**`bookmarks` — Option to import requirements using bookmarks**
`false` | `true`

Option to import requirements content using user-defined bookmarks. This value is `true` by default for Microsoft Word documents and `false` by default for Microsoft Excel spreadsheets.

Example: `'bookmarks', false`

**`match` — Regular expression pattern**
character vector

Regular expression pattern for ID search in Microsoft Office documents.

Example: `'match', '^REQ\d+'`

**`attributes` — Attribute names**
cell array

Attribute names to import, specified as a cell array.

---

**Note** When importing requirements from a Microsoft Excel spreadsheet, the length of this cell array must match the number of columns specified for import using the `'columns'` argument.

---

Example: `'attributes', {'Test Status', 'Test Procedure'}`

**Pairs for Microsoft Excel Spreadsheets**

**`columns` — Range of columns**
`double` array

Range of columns to import, specified as a `double` array.

Example: `'columns', [1 6]`

**rows — Range of rows**
double array

Range of rows to import, specified as a `double` array.

Example: `'rows', [3 35]`

**idColumn — ID Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **ID** field in your requirements set, specified as a `double`.

Example: `'idColumn', 1`

**summaryColumn — Summary Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Summary** field in your requirements set, specified as a `double`.

Example: `'summaryColumn', 4`

**keywordsColumn — Keywords Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Keywords** field in your requirements set, specified as a `double`.

Example: `'keywordsColumn', 3`

**descriptionColumn — Description Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Description** field in your requirements set, specified as a `double`.

Example: `'descriptionColumn', 2`

**rationaleColumn — Rationale Column**
double

Column in the Microsoft Excel spreadsheet that you want to correspond to the **Rationale** field in your requirements set, specified as a `double`.

Example: `'rationaleColumn', 5`

# Examples

## Import Editable Requirements from Microsoft Office Documents

```
% Create a new requirements set and save it
rs = slreq.new('newReqSet');
save(rs);

% Import editable requirements as rich text from a Word document
importFromDocument(rs, 'C:\Work\Requirements_Spec.docx', ...
 'RichText', true);

% Import editable requirements from an Excel spreadsheet
importFromDocument(rs, 'C:\Work\Design_Spec.xlsx', ...
'columns', [2 6], 'rows', [3 32], 'idColumn', 2, ...
'summaryColumn', 3);
```

# See Also
createReferences | slreq.ReqSet

**Introduced in R2018a**

# save

Save a requirements set

## Syntax

```
save(rs)
save(rs, filePath)
```

## Description

`save(rs)` saves a requirements set by using its file name.

`save(rs, filePath)` saves a requirements set and updates its Name and Filename properties.

## Input Arguments

**rs — Requirements set file**
`slreq.ReqSet` object

Requirements set file, specified as a `slreq.ReqSet` object.

**filePath — File name and path**
character vector

The file name and path of the requirements set, specified as a character vector.

Example: `'C:\MATLAB\myReqSet.slreqx'`

# Examples

## Save Requirements Set File

```
% Create the requirements set file
rs = slreq.new('C:\MATLAB\My Requirements Set.slreqx');

% Save the requirements set file
save(rs);

% Save the requirements set file as another requirements set file
save(rs, 'C:\MATLAB\Another Requirements Set.slreqx');
```

# See Also
```
slreq.ReqSet
```

**Introduced in R2018a**

# add

Add requirement to requirements set

## Syntax

```
req = add(reqObj, 'PropertyName', PropertyValue)
```

## Description

`req = add(reqObj, 'PropertyName', PropertyValue)` adds a requirement `req` to a requirements object `reqObj` with properties and custom attributes specified by `PropertyName` and `PropertyValue`.

## Input Arguments

**reqObj — Requirements object**
slreq.ReqSet object | slreq.Requirement object

Requirements set or requirement objects, specified as an `slreq.ReqSet` or as an `slreq.Requirement`object.

## Output Arguments

**req — Requirement**
slreq.Requirement object

The requirement that was added, returned as an `slreq.Requirement` object.

## Examples

### Add a Requirement to a Requirements Set

```matlab
% Load a requirements set file

rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');

% Add a top-level requirement to the requirements set
req1 = add(rs, 'Id', '5', 'Summary', 'Additional Requirement');

% Add a child requirement to the requirement req1
req2 = add(req1, 'Id', '5.1', 'Summary', 'Additional Child Requirement');
```

## See Also
slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# children

Find child requirements of a requirement

## Syntax

```
childReqs = children(req)
```

## Description

`childReqs = children(req)` returns the child requirements `childReqs` of the `slreq.Requirement` object `req`.

## Input Arguments

**req — Requirement instance**
slreq.Requirement object

Requirement specified as an `slreq.Requirement` object.

## Output Arguments

**childReqs — Child requirements**
slreq.Requirement object | slreq.Requirement object array

The child requirements belonging to the requirement `req`, returned as `slreq.Requirement` objects.

# Examples

## Find Child Requirements

```
% Load a requirements set file and add three new requirements

rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
req1 = add(rs, 'Id', '5', 'Summary' , 'Additional Requirement');
req2 = add(req1, 'Id', '5.1', 'Summary', 'Additional Child Requirement 1');
req3 = add(req1, 'Id', '5.2', 'Summary', 'Additional Child Requirement 2');

% Find the children of req1
childReqs = children(req1);

childReqs =

  1×2 Requirement array with properties:

    Id
    Summary
    Keywords
    Description
    Rationale
    SID
    CreatedBy
    CreatedOn
    ModifiedBy
    ModifiedOn
    FileRevision
    Comments
```

# See Also
parent | slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# demote

Demote requirements

## Syntax

```
deomote(req)
```

## Description

deomote(req) demotes the slreq.Requirement object req one level down in the hierarchy.

## Input Arguments

**req — Requirement instance**
slreq.Requirement object

Requirement specified as an slreq.Requirement object.

## Examples

### Demote Requirements

```
% Load a requirements set file and add two new requirements
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
req1 = add(rs, 'Id', '5', 'Summary' , 'Additional Requirement');
req2 = add(req1, 'Id', '5.1', 'Summary' , 'Child Requirement');

% Demote req2
demote(req2);

% Find the parent of req2
parentReq = parent(req2);
```

```
parentReq =

  ReqSet with properties:

              Description: ''
                     Name: 'My_Requirements_Set_1'
                 Filename: 'C:\MATLAB\My_Requirements_Set_1.slreqx'
                 Revision: 6
                    Dirty: 1
      CustomAttributeNames: {}
```

## See Also
promote | slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# find

Find requirements that have matching attribute values

## Syntax

```
reqs = find(rs, 'PropertyName', PropertyValue)
```

## Description

`reqs = find(rs, 'PropertyName', PropertyValue)` returns a requirement or set of requirements `reqs` in the requirements set `rs` specified by the properties that match `PropertyName` and `PropertyValue`.

## Input Arguments

**rs — Requirements set**
`slreq.ReqSet` object

Requirements set specified as an `slreq.ReqSet` object.

## Output Arguments

**reqs — Requirements**
`slreq.Requirement` object | `slreq.Requirement` object array

Requirements, returned as `slreq.Requirement` objects.

# Examples

## Find Requirements That Have Matching Attribute Values

```
% Load a requirements set file
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');

% Find all editable requirements with ID R1.1 in the requirements set
matchedReqs = find(rs, 'Type', 'Requirement', 'ID', 'R1.1');
```

# See Also

slreq.Requirement | slreq.find

**Introduced in R2018a**

# getAttribute

Get requirement attributes

## Syntax

```
val = getAttribute(req, propertyName)
```

## Description

`val = getAttribute(req, propertyName)` gets a requirement property that is specified by `propertyName`.

## Input Arguments

**req — Requirement instance**
`slreq.Requirement` object

Requirement specified as an `slreq.Requirement` object.

**propertyName — Requirement property**
character vector

Requirement property name.

Example: `'SID', 'CreatedOn', 'Summary'`

## Examples

### Get Requirement Attributes

```
% Load a requirements set file and get the handle to one requirement
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
req1 = find(rs, 'Type', 'Requirement', 'ID', 'R1.1');
```

```
% Get the Summary of req1
summaryReq1 = getAttribute(req1, 'Summary')

summaryReq1 =

  'Functional Requirements'
```

## See Also
setAttribute | slreq.Requirement

**Introduced in R2018a**

# parent

Find parent item of requirement

## Syntax

```
parentObj = parent(req)
```

## Description

`parentObj = parent(req)` returns the parent object `parentObj` of the `slreq.Requirement` object `req`.

## Input Arguments

**req — Requirement instance**
`slreq.Requirement` object

Requirement specified as an `slreq.Requirement` object.

## Output Arguments

**parentObj — Parent object**
`slreq.Requirement` object | `slreq.ReqSet` object

The parent of the requirement `req`, returned as an `slreq.Requirement` object or as an `slreq.ReqSet` object.

# Examples

## Find Parent Objects of Requirements

```
% Load a requirements set file and add two new requirements

rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
req1 = add(rs, 'Id', '5', 'Summary' , 'Additional Requirement');
req2 = add(req1, 'Id', '5.1', 'Summary' , 'Additional Child Requirement');

% Find the parent of req2
parentReq1 = parent(req2)

parentReq1 =

  Requirement with properties:

             Id: '5'
        Summary: 'Additional Requirement'
       Keywords: [0×0 char]
    Description: ''
      Rationale: ''
            SID: 10
      CreatedBy: 'John Doe'
      CreatedOn: 05-Oct-2007 16:09:38
     ModifiedBy: 'Jane Doe'
     ModifiedOn: 21-Dec-2016 11:10:05
       Comments: [0×0 struct]

% Find the parent of req1
parentReq2 = parent(req1)

parentReq2 =

  ReqSet with properties:

             Description: ''
                    Name: 'My_Requirements_Set_1'
                Filename: 'C:\MATLAB\My_Requirements_Set_1.slreqx'
                Revision: 6
                   Dirty: 1
     CustomAttributeNames: {}
```

## See Also

children | slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# promote

Promote requirements

## Syntax

```
promote(req)
```

## Description

`promote(req)` promotes the `slreq.Requirement` object `req` one level up in the hierarchy.

## Input Arguments

**req — Requirement instance**
`slreq.Requirement` object

Requirement specified as an `slreq.Requirement` object.

## Examples

### Find Requirements with Matching Attribute Values

```
% Load a requirements set file and add two new requirements
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
req1 = add(rs, 'Id', '5', 'Summary' , 'Additional Requirement');
req2 = add(req1, 'Id', '5.1', 'Summary' , 'Child Requirement');

% Promote req2
promote(req2);

% Find the parent of req2
parentReq = parent(req2);
```

```
parentReq =

  ReqSet with properties:

            Description: ''
                   Name: 'My_Requirements_Set_1'
               Filename: 'C:\MATLAB\My_Requirements_Set_1.slreqx'
               Revision: 6
                  Dirty: 1
    CustomAttributeNames: {}
```

## See Also
demote | slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# reqSet

Return parent requirements set

## Syntax

```
rsout = reqSet(req)
```

## Description

rsout = reqSet(req) returns the parent requirements set rsout to which the requirement req belongs.

## Input Arguments

**req — Requirement object**
slreq.Requirement object

Requirement, specified as an slreq.Requirement object.

**rsout — Parent requirements set**
slreq.ReqSet object

The parent requirements set of the requirement req, returned as an slreq.ReqSet object.

## Examples

### Query Requirements Set Information

```
% Load a new requirements set file and select one requirement
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
allReqs = find(rs, 'Type', 'Requirement');
req = allReqs(1);
```

```
% Query which requirements set req belongs to
reqSet(req);

ans =

  ReqSet with properties:

            Description: ''
                   Name: 'My_Requirements_Set_1'
               Filename: 'C:\MATLAB\My_Requirements_Set_1.slreqx'
               Revision: 63
                  Dirty: 0
      CustomAttributeNames: {}
```

## See Also
parent | slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# setAttribute

Set requirement attributes

## Syntax

```
setAttribute(req, propertyName, propertyValue)
```

## Description

`setAttribute(req, propertyName, propertyValue)` sets a requirement property.

## Input Arguments

**req — Requirement instance**
`slreq.Requirement` object

Requirement specified as an `slreq.Requirement` object.

**propertyName — Requirement property**
character vector

Requirement property name.

Example: `'SID', 'CreatedOn', 'Summary'`

**propertyValue — Requirement property value**
character vector

Requirement property value.

Example: `'Test Requirement', 'R1.3.1'`

# Examples

## Set Requirement Attributes

```matlab
% Load a requirements set file and get the handle to one requirement
rs = slreq.load('C:\MATLAB\My_Requirements_Set_1.slreqx');
req1 = find(rs, 'Type', 'Requirement', 'ID', 'R2.1');

% Set the Summary of req1
setAttribute(req1, 'Summary', 'Controller Requirement');

req1

req1 =

  Requirement with properties:

              Id: 'R2.1'
         Summary: 'Controller Requirement'
        Keywords: [0×0 char]
     Description: ''
       Rationale: ''
             SID: 21
       CreatedBy: 'Jane Doe'
       CreatedOn: 27-Feb-2014 10:15:38
      ModifiedBy: 'John Doe'
      ModifiedOn: 02-Aug-2017 13:49:40
    FileRevision: 43
           Dirty: 1
        Comments: [0×0 struct]
```
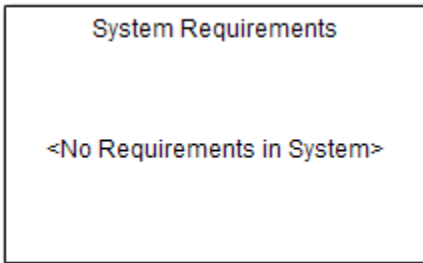
# See Also
getAttribute | slreq.ReqSet | slreq.Requirement

**Introduced in R2018a**

# Block Reference

# System Requirements

List system requirements in Simulink models

```
System Requirements


<No Requirements in System>
```

## Library

Simulink Requirements

## Description

The System Requirements block lists the system-level requirements associated with a model or subsystem. This block is dynamically populated. It displays system requirements associated with the level of hierarchy in which the block appears in the model. It does not list requirements associated with individual blocks in the model. To ensure that all requirement links are listed in the System Requirements block:

1  Right-click the background of your model.
2  Select **Requirements at This Level**.
3  From the top of the context menu, verify that all the requirements you want to list appear in the System Requirements block.

You can place this block anywhere in your model. It does not connect to other Simulink blocks. You can have only one System Requirements block in a given subsystem.

When you insert this block into your Simulink model, it is populated with the system requirements, as shown in the `Airflow Calculation` subsystem of the `slvnvdemo_fuelsys_officereq` example.
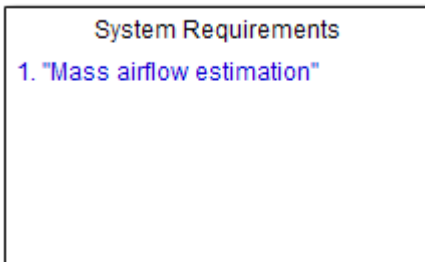
Each of the listed requirements is an active link to the requirements document. When you double-click a requirement label, the associated requirements document opens in its editor window, scrolled to the target location.

# Parameters

**Block Title**

The title of the system requirements list in the model. The default title is `System Requirements`. You can enter a customized title, for example, `Engine Requirements`.

**Introduced before R2006a**