Simulink® Report Generator™

User's Guide

R2014**b**

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

Simulink® Report Generator™ User's Guide

Revision History

January 1999	First printing	New (Release 11)
December 2000	Second printing	Revised (Release 12)
June 2004	Third printing	Revised for Version 2.02 (Release 14)
August 2004	Online only	Revised for Version 2.1
October 2004	Online only	Revised for Version 2.1.1 (Release 14SP1)
December 2004	Online only	Revised for Version 2.2 (Release 14SP1+)
April 2005	Online only	Revised for Version 2.2.1 (Release 14SP2+)
September 2005	Online only	Revised for Version 2.3.1 (Release 14SP3)
March 2006	Online only	Revised for Version 3.0 (Release 2006a)
September 2006	Online only	Revised for Version 3.1 (Release 2006b)
March 2007	Fourth printing	Revised for Version 3.2 (Release 2007a)
September 2007	Fifth printing	Revised for Version 3.2.1 (Release 2007b) This publication was previously for MATLAB® and Simulink®. It is now for Simulink® only.
March 2008	Online only	Revised for Version 3.3 (Release 2008a)
October 2008	Online only	Revised for Version 3.4 (Release 2008b)
October 2008	Online only	Revised for Version 3.5 (Release 2008b+)
March 2009	Online only	Revised for Version 3.6 (Release 2009a)
September 2009	Online only	Revised for Version 3.7 (Release 2009b)
March 2010	Online only	Revised for Version 3.8 (Release 2010a)
September 2010	Online only	Revised for Version 3.9 (Release 2010b)
April 2011	Online only	Revised for Version 3.10 (Release 2011a)
September 2011	Online only	Revised for Version 3.11 (Release 2011b)
March 2012	Online only	Revised for Version 3.12 (Release 2012a)
September 2012	Online only	Revised for Version 3.13 (Release 2012b)
March 2013	Online only	Revised for Version 3.14 (Release 2013a)
September 2013	Online only	Revised for Version 3.15 (Release 2013b)
March 2014	Online only	Revised for Version 3.16 (Release 2014a)
October 2014	Online only	Revised for Version 4.0 (Release 2014b)

Contents

1 Getting Started

2 Generate System Design Description Reports

3 Creating Simulink Reports

4 Generate a Report

5 Export Simulink Models to Web Views

6 Add Content with Components

7 Compare Simulink Model XML Files

8 Components — Alphabetical List

9 Functions – Alphabetical List

10 Template-Based Report Formatting

11 Create a Report Program

Getting Started

Simulink Report Generator Product Description

Design and generate reports from models and simulations

Simulink Report Generator lets you design and generate richly formatted Microsoft® Word, HTML, and PDF reports from Simulink models and simulations. The report generator lets you automatically create artifacts for Model-Based Design, such as system design descriptions and generated code, requirements traceability, and testing reports. You can publish these artifacts in an interactive web format that colleagues can use without opening the model. You can compare Simulink models, review comparison results in an interactive XML report, and merge model differences.

Simulink Report Generator produces artifacts for DO-178, ISO 26262, IEC 61508, and related industry standards.

Key Features

- Automatic capture of simulation results and model specifications
- Report formatting based on Word and HTML report templates
- Interactive reports for viewing models, generated code, and analysis results in web browsers
- Report designer for creating custom Word, HTML, PDF, RTF, and XML reports
- Artifacts for DO-178, IEC 61508, and ISO 26262, including system design, model and code verification, and requirements documentation
- API for forms-based Word and HTML report generation
- Model differencing and merging with XML comparison tool

System Design Documentation and Results Reporting

In this section...
“Types of Reports” on page 1-3
“System Design Documentation” on page 1-3
“Results Reporting” on page 1-4

Types of Reports

Two common user goals for reports are:

- System design documentation — Capture information about the design decisions, structure, implementation, and operational details of a system.
- Results reporting — Present results of running a system.

You use a very similar workflow for creating and generating both kinds of reports. However, some components are particularly useful for one or the other kind of report.

System Design Documentation

System documentation has many uses, including:

- Capturing design decisions
- Recording implementation details
- Communicating the system design and interfaces among groups

You can view information about a system without creating a report. For more information, see “Interactive Model Exploration” on page 1-6.

When you create a Simulink Report Generator report to provide system design documentation, the report captures information about the system design directly from the model. Each time that you generate the report, you see up-to-date documentation for the design.

The following table includes some examples of components that are useful for system design documentation reports.

System Information	Examples of Components to Use
Requirements	`Requirements Summary Table` (for requirements specified with Simulink Verification and Validation™)
System layout	`System Hierarchy`, `System Snapshot`
Model configuration	`Model Configuration Set`, `Model Advisor Report`
Block parameter settings	`Simulink Dialog Snapshot`, `Block Loop`
Properties	`Simulink Property Table`, `Simulink Summary Table`
Variables	`Variable Table`, `Simulink Workspace Variable`
System documentation included in a model	`Documentation`, `Simulink Name`

Results Reporting

Results reporting has many uses, including:

- Model regression testing
- Verifying and validating designs
- Exploring design alternatives
- Optimizing designs

Simulink provides several tools for examining the results of a simulation. Simulation data includes signal, time, output, state, and data store logging data. You can set up your model to export simulation data to the MATLAB® workspace during simulation for later retrieval, analysis, and postprocessing. You can use several approaches to explore the simulation data. For details, see "Export Simulation Data" in the Simulink documentation.

You can report on results without creating a report. For more information, see "Interactive Model Exploration" on page 1-6.

The following table includes some examples of components that are useful results reports.

Results Information	Examples of Components to Use
Signal values	`Scope Snapshot`, `Block Loop`

Results Information	Examples of Components to Use
Simulation processing	`Model Simulation`, `Model Configuration Set`, `Fixed-Point Logging Options`
Figures generated with MATLAB	`Figure Snapshot`, `To Workspace Plot`
Generated code	`Code Generation Summary`, `Import Generated Code`

You can use components such as the Model Simulation component to control how the model simulates. Other components, such as the Scope Snapshot, show the results of the simulation.

Interactive Model Exploration

In this section...

Simulink Report Generator Tools

The Simulink Report Generator provides tools that you can use to obtain information about models, without creating report setup files:

- Predefined standard reports
- Web View
- XML Comparison report

Predefined Standard Reports

From the Simulink Editor, you can access two predefined, standard Simulink Report Generator reports:

- System Design Description
- Design Requirements (requires Simulink Verification and Validation)

The System Design Description report provides summary or detailed information about a system design represented by a model. You can choose report options with the report dialog box, or you can create a customized version using the Report Explorer. For details, see "Generate a System Design Description".

You can use the System Design Description report setup file as a starting point for creating a setup file for your own report.

You can also generate an HTML model report for Stateflow® charts.

You must have the Simulink Verification and Validation product installed to use the Design Requirements report. The Design Requirements report includes information about all the requirements associated with the model and its objects. For details, see "Customize Requirements Traceability Report for Model" in the Simulink Verification and Validation documentation.

Web View

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

Web views are useful for presenting models to audiences and for sharing models with colleagues who do not have MathWorks® products installed.

For details, see “Export Models to Web View Files”.

XML Comparison Report

You can use Simulink Report Generator software to compare XML text renditions of different versions of a Simulink model and to compare the XML renditions to explore differences between versions of the model. You can also compare the XML text renditions of two different models.

For details, see “Model Comparison”.

Simulink Model Exploration Tools

You can generate a model report, which is an HTML document that describes the structure and content of a model. The model report includes block diagrams of the model and its subsystems and the settings of its block parameters. To generate the report, from the Simulink Editor, select **File > Print Details**. For more information, see “Generate a Model Report”.

To interactively focus on specific elements of a model (for example, blocks, signals, and properties) without navigating through the model diagram or chart, using the Model Explorer. To open the Model Explorer, in the Simulink Editor, select **View > Model Explorer**. For details, see “Model Explorer Overview”.

To generate a report that lists the suboptimal conditions or settings in a model, with suggestions for better model configuration settings where appropriate, use the Model Advisor. To open the Model Advisor, in the Simulink Editor, select **Analysis > Model Advisor**. For details, see “Consulting the Model Advisor”

To navigate a model hierarchically, open systems in a model, and determine the blocks contained in a model, you can use the Simulink Model Browser. To open the Model

Browser, select **View** > **Model Browser Options** > **Model Browser**. For details, see "Model Browser".

Simulink provides several tools for examining the results of a simulation. Simulation data includes signal, time, output, state, and data store logging data. You can set up your model to export simulation data to the MATLAB workspace during simulation for later retrieval, analysis, and postprocessing. You can use several approaches to explore the simulation data. For example, you can use the Simulation Data Inspector to generate a report with plots of simulation data. For details, see "Export Runtime Information" in the Simulink documentation.

Report Creation Workflow

Approaches for Creating Reports

You can create and generate reports :

- Interactively, using the Report Explorer
- Programmatically, using the DOM (Document Object Model) API

You can use the Report Explorer graphical interface to create reports without having to write code.

Using the programmatic approach, you can integrate report generation into analysis and testing applications. For more information, see “Programmatic Report Creation”.

Interactive Report Creation

1. Open the Report Explorer. In the menu bar on the Simulink model window, click **Tools** > **Report Generator**.
2. Create a new report setup file.

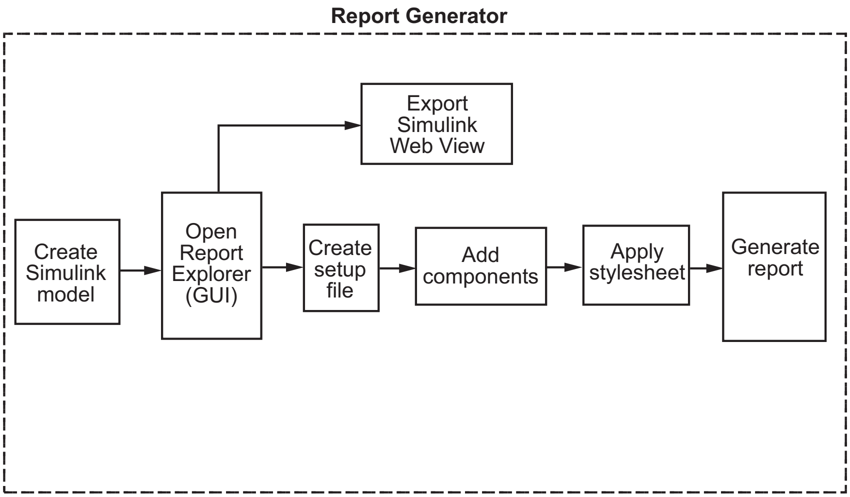
   For information about report setup files, see “Report Setup”.
3. Add existing Simulink or Stateflow components to the report setup file, or create your own custom components.

   For information on using components, see “Working with Components”.

   “Working with Components”
4. Choose a Microsoft Word, HTML, or PDF template or a Report Explorer stylesheet to apply styles the report setup file.

   For information about generating a report using a template, see “Report Generation Using Templates”. For information about Report Explorer style sheets, see “Layout Stylesheets”.
5. Generate the report.

The following figure illustrates a typical Simulink Report Generator workflow.


Report Generator
Export
Simulink
Web View
Create
Simulink
model
Open
Report
Explorer
(GUI)
Create
setup
file
Add
components
Apply
stylesheet
Generate
report

Report Components

In this section...

About Report Components

Components are modules that you include in a report setup file to insert elements, such as tables, lists, and figures, into a report. You also can use components to control report generation processing.

Many components have parent-child relationships with other components. For example, a child components of a Paragraph component include components such as Text, Image, and Insert Variable.

Use a combination of the following types of components in your report setup file.

Component Type	Description
"Report Structure Components" on page 1-12	Include a title page, sections, and other components to organize the content of a report.
"System-Based Components" on page 1-12	Include components that automatically obtain information directly from a model to include in a report.
"User-Supplied Information Components" on page 1-14	Include text and graphics that you supply, independent of a model.
"Dynamic Reporting Components" on page 1-14	Set up dynamic control for when to include components and what information to report on for a component, based on data from

Component Type	Description
	a model or on other conditions that you specify.

Use the Report Explorer to add components to a report and to specify component properties.

Report Structure Components

To add a title page, use a `Title Page` component. You can include an abstract and legal notice information. For an example, see “Add a Title Page”.

To organize a report into sections, use `Chapter/Subsection` components. For an example, see “Create a Section for Each Iteration”.

System-Based Components

The Simulink Report Generator includes many components that obtain information directly from a model and include that information in a report. Using system-based components allows your report to reflect the current state of a model. You can generate the report whenever you want to capture the latest version of a model, without changing the report setup file.

Property table components display property name/property value pairs for objects in tables. Summary table components insert tables that include specified properties for objects into generated reports. The tables contain one object per row, with each object property appearing in a column, as shown in the following summary table.

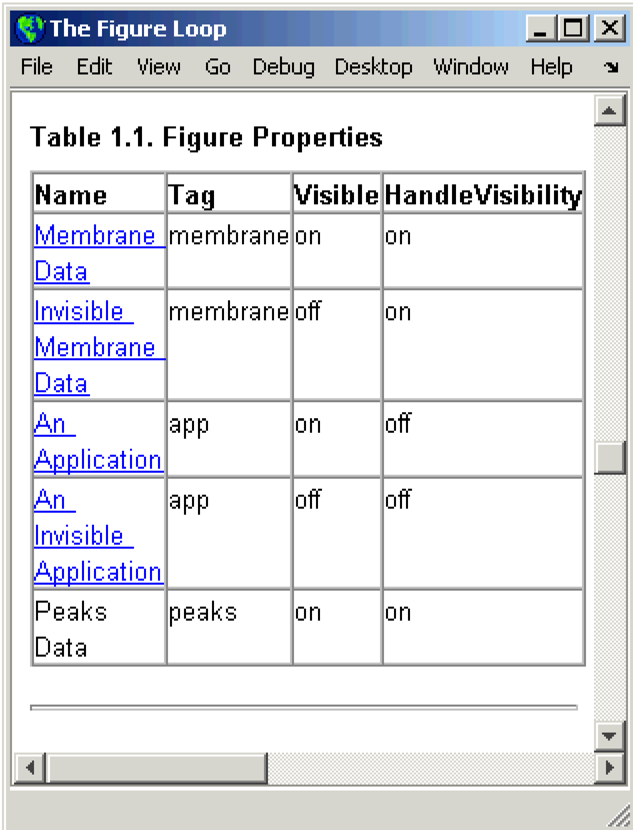


Table 1.1. Figure Properties

Name	Tag	Visible	HandleVisibility
Membrane Data	membrane	on	on
Invisible Membrane Data	membrane	off	on
An Application	app	on	off
An Invisible Application	app	off	off
Peaks Data	peaks	on	on

To use descriptive information from DocBlock blocks, use the `Documentation` component.

A few examples of system-based components include:

- `MATLAB Property Table`
- `Simulink Workspace Variable`
- `System Hierarchy`
- `Simulink Summary Table`
- `Simulink Dialog Snapshot`
- `Block Execution Order List`
- `Model Loop`

- `Model Configuration Set`
- `Scope Snapshot`

For examples of using system-based components, see:

- "Property Table Components"
- "Summary Table Components"
- "Create the Body of the Report"

The Simulink Report Generator also includes system-based components that contain model elements from the following Simulink products:

- Stateflow
- Fixed-Point Designer™
- Simulink Coder™
- Simulink Verification and Validation

User-Supplied Information Components

In addition to using system-based components to extract data from a system and insert that information into a report, you can also add content that you, or others, supply. For example, to include text, use the `Paragraph` and `Text` components.

To insert a graphic from a file, use an `Image` component. To insert ASCII text, use an `Import File` component.

To include notes about the report source files, use a `Comment` component.

For an example, see "Add Introductory Text to the First Chapter".

Dynamic Reporting Components

Dynamic reporting components execute conditionally, enabling you to decide when a child component executes or how many times a child component executes. To control the report generation flow, use logical and flow components such as `Logical If`, `Logical Then`, `While Loop`, or `For Loop`.

A looping component runs its child components a specified number of times. There are several looping components, including logical loops, Handle Graphics® loops, and model and chart loops. For model and chart loops, you can control aspects such as the order in which the report sorts blocks.

For examples, see:

- “Logical and Looping Components”
- “Add Logical Then and Logical Else Components”
- “Create the Body of the Report”
- “Filter with Loop Context Functions”

Format Control at the Component Level

The output format and stylesheet that you select for a report determines most aspects of the generated report formatting. For details, see “Report Output Format”.

In addition to stylesheets that control the format and layout of the report, for some components you can set properties to specify formatting details for that specific instance of a component. For example, for the `Simulink Property Table`, you can specify whether to display table borders or specify the alignment of text in table cells.

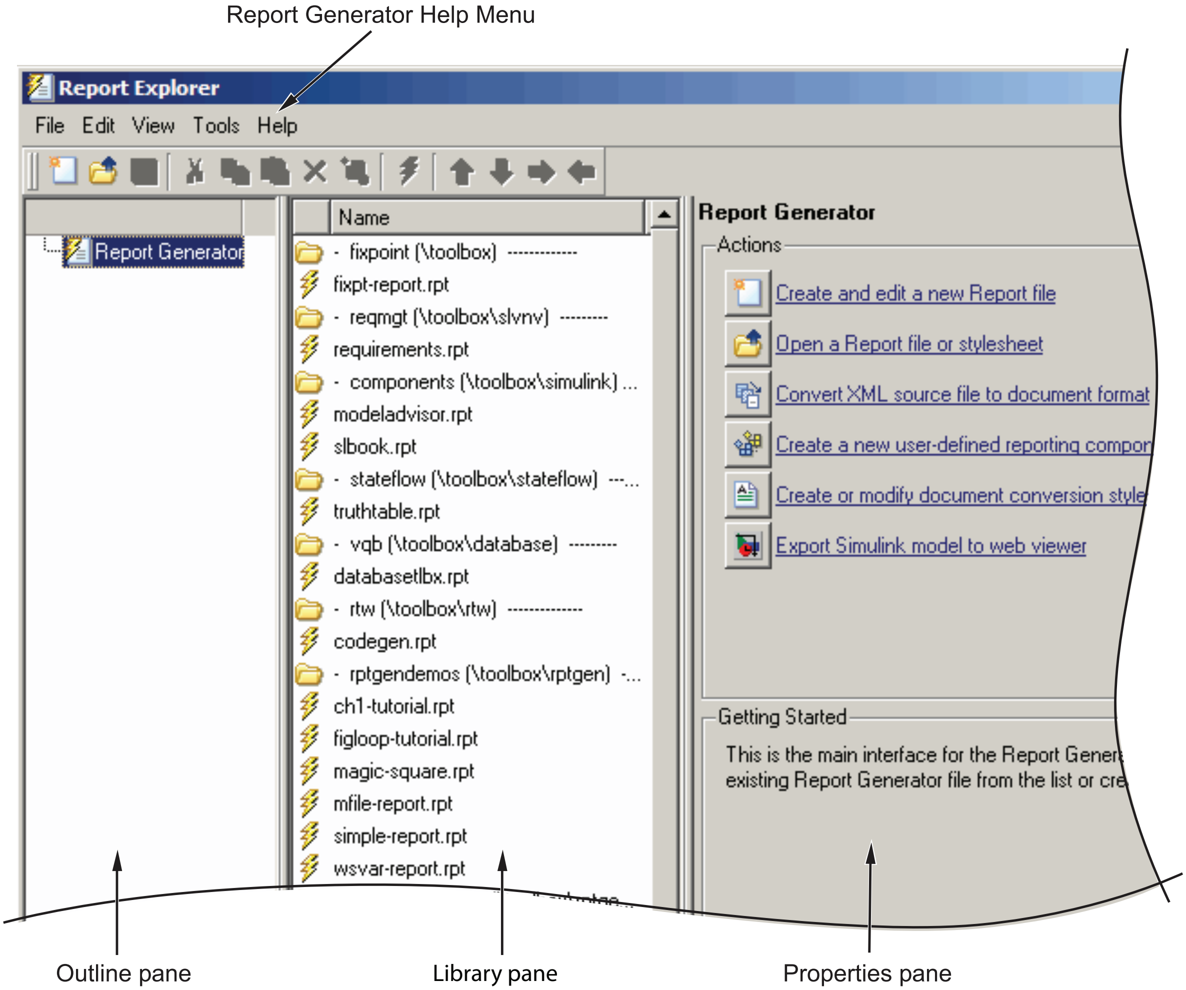
Report Explorer

About the Report Explorer

The *Report Explorer* is the MATLAB Report Generator and Simulink Report Generator graphical interface. It allows you to:

- Create and modify report setup files.
- Apply stylesheets to format the generated report.
- Specify the report file format.
- Generate reports.

To open the Report Explorer, enter `report` in the MATLAB Command Window.



The Report Explorer has three panes:

- The *Outline pane* on the left shows the hierarchy of components in currently opened report setup files. Report components can reside within other report components, creating parent, child, and sibling relationships.
- The *Library pane* in the middle lists the objects available in the context of the Outline pane.

Outline Pane Context	Library Pane Contents
No report setup file is open.	Reports
Report setup file is open.	Components
Stylesheet is open.	Stylesheet attributes

- The *Properties pane* contents depend on the Outline pane context. If no report setup file is open, on the right displays tasks the Report Explorer can perform. If a report setup file is open, the Properties pane displays the properties for the item that is currently selected in the Library pane.

Outline Pane Context	Properties Pane Contents
No report setup file is open.	Tasks that the Report Explorer can perform
Report setup file is open.	Properties for the item that is currently selected After you create a report setup file, the Properties pane initially displays properties for the report setup file as a whole.

Tip If the Report Explorer window opens with only two panes, one of the panes is hidden. You can move the vertical boundaries between the panes to reveal any hidden pane, or to make visible panes wider or narrower.

How Simulink Report Generator and Simulink Software Interact

The Simulink and Simulink Report Generator software interact to create reports and Web views. The following table describes these interactions.

User Interface	Interaction with Simulink Report Generator software	Description
Report Explorer	The Report Explorer is the Simulink Report Generator graphical interface. For more information, see "Report Explorer" on page 1-16.	Use the Report Explorer to edit existing report setup files, components, stylesheets, and attributes, or to customize your own.
Simulink model window	Use Simulink model window to: • Export the model to a Web view • Generate a System Design Description report • Start the Report Explorer to export the model to a Web view or create and generate a report	Use the Simulink model window to: • Create reports that incorporate the current Simulink model • Export the model to a Web view For more information, see "Model Web Views" and "Generate a System Design Description".

Simulink Report Generator software also interacts with Simulink Coder and Stateflow software.

Use report generation capabilities with the Simulink Coder software to:

- Create Adobe® Acrobat® PDF and Microsoft Word documents for generated code.

Use report generation capabilities with the Stateflow software to:

- Take snapshots of charts.
- Describe truth tables.
- Document chart hierarchy.
- Document object properties.

Supported Report Formats

When the report generation process first creates a report, it generates a DocBook XML source file. You can customize this XML as needed. For more information, see the OASIS™ DocBook TC Web page at `http://www.oasis-open.org/committees/docbook` and `http://www.docbook.org/tdg/en/html/docbook.html`.

Next, the report generation process converts the XML source to one of these user-specified report formats:

- Adobe Acrobat PDF

  Note: PDF reports only support bitmap (`.bmp`), jpeg (`.jpg`), and Scalable Vector Graphics (`.svg`) images. The SVG format is only supported for Simulink models and Stateflow charts. For example, MATLAB figures do not display in SVG when you select the SVG format for PDF reports.

- Hypertext Markup Language (HTML)
- Microsoft Word (`.doc`)
- Rich Text Format (RTF)

  Note: RTF reports use placeholders (field codes) for dynamically generated content, such as page numbers or images.

  On Windows® platforms, to display that content, press **Ctrl-A**, and then press **F9**.

  On Linux® and Mac platforms, use the field code update interface for the program that you are using to view the RTF document.

2

Generate System Design Description Reports

System Design Description

In this section...

Predefined Standard Reports

From the Simulink Editor, you can access two predefined, standard Simulink Report Generator reports called:

- System Design Description
- System Requirements Traceability

The System Design Description report provides summary or detailed information about a system design represented by a model. You can choose report options using the report dialog, or you can create a customized version using the Report Explorer. For details, see "Generate a System Design Description".

You can use the System Design Description report setup file as a starting point for creating a setup file for your own report.

You can also generate an HTML model report for Stateflow charts. For details, see "Generate a Model Report".

The System Requirements Traceability report requires that you have the Simulink Verification and Validation product installed. The System Requirements Traceability report includes information about all the requirements associated with the model and its objects. For details, see "Customize Requirements Traceability Report for Model" in the Simulink Verification and Validation documentation.

What Is the System Design Description?

The System Design Description is a prebuilt Simulink Report Generator report that describes the system design represented by a Simulink model.

By default, the Simulink Report Generator generates the report for the model from which you invoke the System Design Description report option.

What You Can Do with the Report

You can use the System Design Description to

- Review a system design without having the model open
- Generate summary and detailed descriptions of the design
- Assess compliance with design requirements
- Archive the system design in a format independent of the modeling environment
- Build a customized version of the report, using the Report Explorer

---

Note: To view step-by-step tutorials for creating and generating a report, see `the Introduction to System Design Description Reports` example.

Report Contents

You can specify what kinds of information to include in the report, in terms of:

- What elements of a model to include in the report (for example, whether to include subsystems from custom libraries)
- Whether to generate a summary version or a detailed version of the System Design Description report.

For details, see "Generate a System Design Description" on page 2-6.

Summary Version

Section	Information
Report Overview	Model version
Root System	• Block diagram representing the algorithms that compute root system outputs • Description (if available from model) • Interface: name, data type, and other properties of the system input and output signals

Section	Information
	• Subsystems: the path and a block diagram for each subsystem • State charts • Requirements (optional)
Subsystems	• Path • Block diagram
System Design Variables	• Design variables • Functions in design variable expressions

Detailed Version

The detailed version of the report includes all the information that is in the summary form of the report, as well as more information about the system components. The atomic subsystem information is more detailed than virtual subsystem information.

Section	Information
Report Overview	Model version
Root system	• Block diagram representing the algorithms that compute root system outputs • Description (if available from model) • Interface: name, data type, and other properties of the root system input and output signals • Block parameters • Includes detailed information about MATLAB Function blocks • Block execution order for root system and atomic subsystems • Look-up tables • Simulink workspace variables • Model configuration sets • State charts • Requirements (optional)

Section	Information
Subsystems	The same type of information as the information for the root system, as well as: • Path of the subsystem in the model • (For atomic subsystems) Checksum that indicates whether the version of an atomic subsystem that generates the report differs from other versions of the subsystem • Referenced models (optional) • Subsystems from custom libraries (optional)
State Charts	• State chart • States • Transitions between the states • Junctions • Events that trigger state transitions • Data types • Targets • Truth tables

Report Captures Documentation Included in a Model

The System Design Description reports documentation included in a model, including:

- The model description (from the model properties)
- The block property `Description`
- DocBlock model documentation blocks

To enrich the generated System Design Description, consider adding descriptive information in a model in these ways listed above.

Generate a System Design Description

In this section...

Open the System Design Description Dialog Box

To open the System Design Description dialog box:

1. Open the model or subsystem for which you want to generate a report.

Note: The System Design Description fails for models that simulate with an error.

---

2. From the Simulink Editor **File** menu, select **Reports** > **System Design Description**. The System Design Description dialog box opens
3. Specify layout and content options for the report. For details, see "Choose System Design Description Options" on page 2-6.
4. Click the **Generate**

Choose System Design Description Options

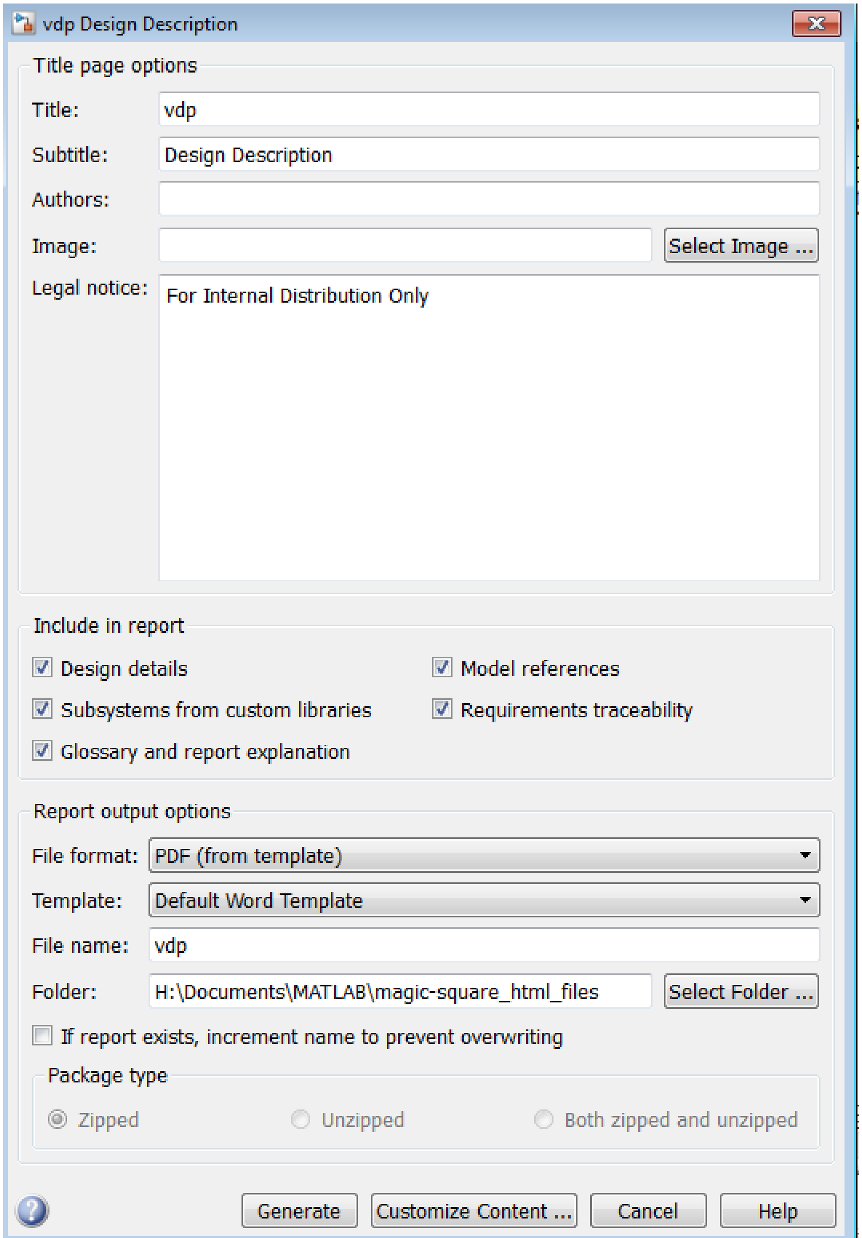
You can specify options for layout and content, for the following items:

- Title page contents
- Report content
- Report file format and storage location

Tip For faster report generation, set File format to use a template. Select one of these formats:

- `PDF (from template)`, which is the default
- `Word (from template)`

- `HTML (from template)`


vdp Design Description
Title page options
Title:
vdp
Subtitle:
Design Description
Authors:
Image:
Select Image ...
Legal notice:
For Internal Distribution Only
Include in report
Design details
Model references
Subsystems from custom libraries
Requirements traceability
Glossary and report explanation
Report output options
File format:
PDF (from template)
Template:
Default Word Template
File name:
vdp
Folder:
H:\Documents\MATLAB\magic-square_html_files
Select Folder ...
If report exists, increment name to prevent overwriting
Package type
Zipped
Unzipped
Both zipped and unzipped
Generate
Customize Content ...
Cancel
Help

In the dialog box, to display detailed information about each option, right-click on the option prompt and select the **What's This** context-sensitive help.

To create a customized version of the report, click the **Customize Content** button. The customize option creates a copy of the report setup file and opens the copy in the Report Explorer. See "Customize the System Design Description" on page 2-10.

Customize the System Design Description

In this section...
"Using the Report Explorer to Customize the Report" on page 2-10
"Building a Dialog Box for a Custom Report Setup File" on page 2-11

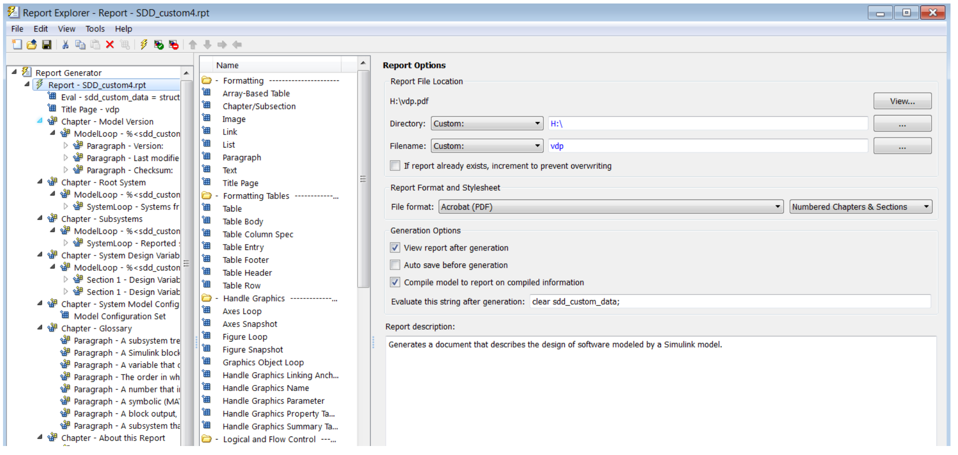
You can create customized versions of the System Design Description report by using the Report Explorer and, optionally the MATLAB tools for building graphical user interfaces.

By default, when you open a customized version of the report, the System Design Description dialog box does not open.

Using the Report Explorer to Customize the Report

To customize the System Design Description setup file in the Simulink Report Generator using the Report Explorer:

1 In the System Design Description dialog box, click the **Customize Content** button to open the Report Explorer.



The Report Explorer reflects any changes (for example, a different report name) that you made in the System Design Description dialog box.

2 In the Report Explorer, add or modify components. See "Working with Components" and "Information Components".

- Do not remove the sdd_custom_data structure, which is defined as:

  ```
  sdd_custom_data = struct('model',bdroot,'rootSystem',gcs);
  ```

  You can modify the model argument, which is the model for which you generated the report and the rootSystem argument, which is the system level in the model at which, and below which, you want to use to generate the report.
- Do not remove or modify functions that begin with StdRpt, such as %StdRpt.getChecksum

3. Optionally modify a style sheet (see "Layout Stylesheets").
4. Save the customized report with a name other than SDD_custom.rpt.

Building a Dialog Box for a Custom Report Setup File

To provide options for your custom report, you can create a dialog box, like the System Design Description dialog box. The dialog box that you create for your custom report can allow others to adapt the report to meet their needs, without their having to use the Report Explorer.

3

Creating Simulink Reports

Create a Simulink Report Generator Report

This example shows how to use the Report Explorer to design a report setup file and generate a report that does the following:

- Opens a Simulink model for the van der Pol equation, called the `vdp model`.
- Sets the `Gain` parameter for the `Mu` block to five different values.
- Simulates the model each time the `Gain` parameter is set.
- Collects the results. Results that fall within a specified range appear in a table in the generated report.

You do not need to know MATLAB or Simulink software to create and run this example report. However, knowledge of these products might be helpful for understanding the MATLAB code and model simulation that executes.

To create this report, you perform these main tasks:

- "Specify Report Options in the Setup File" on page 3-3
- "Add Report Content with Components" on page 3-5

This example includes separate sections for different kinds of report creation and generation task. Each section builds on the previous sections. However, if you want to see the report setup components for a later section without doing the previous sections, in MATLAB you can view the completed report setup file by opening `Dynamic Simulink Report`. The report is for the `vdp` model.

Note: For another set of step-by-step examples for creating and generating a report, see the Introduction to System Design Description Reports example.

Specify Report Options in the Setup File

To create and configure the report setup file:

1. Start a Simulink software session.
2. Open the Report Explorer. From the MATLAB Toolstrip, in the **Apps** tab, in the **Database Connectivity and Reporting** section, click **Report Generator**.
3. Select **File** > **New** to create a report setup file.
4. Save the report setup file.

   In the Properties pane:

   a. Specify where to save the report setup file. To save it in the current working folder, select `Present Working Directory` from the **Directory** selection list.
   b. Specify the report format. In the **File format** selection list, select `Acrobat (PDF)`.
   c. Enter a description for the report. In the **Report description** text box, replace the existing contents with the following text.

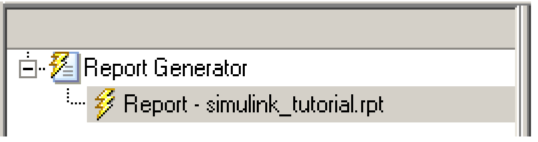
   Tip Copy and paste this code from the HTML documentation into the Report Explorer.

   ```
   Simulink Dynamic Report

   This report opens up a model, sets a block parameter
   several times, simulates the model, and collects the
   results. Results that fall within a specified range are
   displayed in a table after the test is complete.

   The report is configured to test the vdp model only.
   By selecting the Eval String component immediately
   below the Report component, you can modify
   * model
   * block
   * parameter
   * tested values
   ```
5. Click **File** > **Save As** to save the report setup file as `simulink_tutorial.rpt`.

The Outline pane on the left displays the new file name.



To create the content for the report, see "Add Report Content with Components" on page 3-5.

Add Report Content with Components

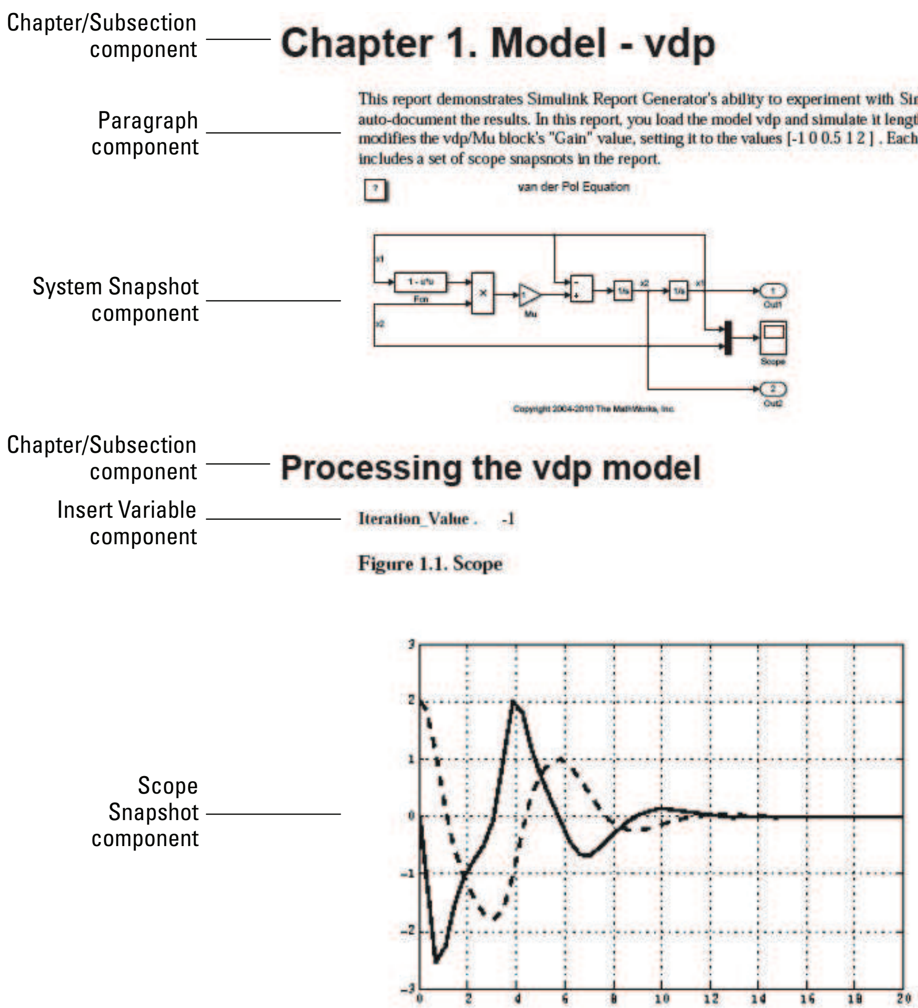
Report Components

Report *components* specify what information to include in the report. Components are self-contained, modular MATLAB objects that control the report-generation process and insert elements, such as tables, lists, and figures, into a report setup file. Use components to customize the appearance and output of reports.

For more information, see "Report Components".

The following figure shows a sample page from the report you create in this example, and which components you use to produce this output.

Note: Do not deactivate report components that you add to the report setup file.



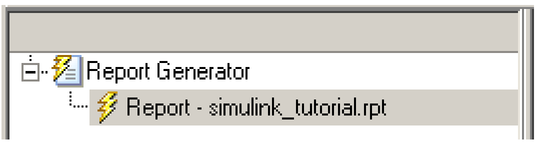
Add MATLAB Code

Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

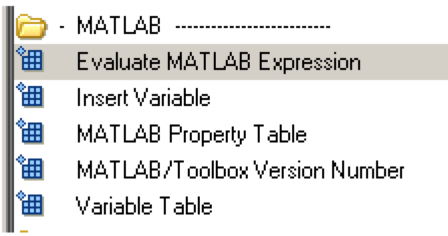
To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

The first component to add is the `Evaluate MATLAB Expression` component, which evaluates MATLAB commands in the workspace. The code in this component assigns initial values to variables used in this example.

1 In the Outline pane on the left, select `simulink_tutorial.rpt`.



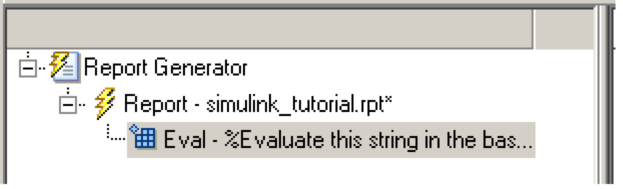
2 In the Library pane in the middle, under the `MATLAB` category, select `Evaluate MATLAB Expression`.



3 In the Properties pane on the right, click the icon next to **Add component to current report** to insert the component into the report.

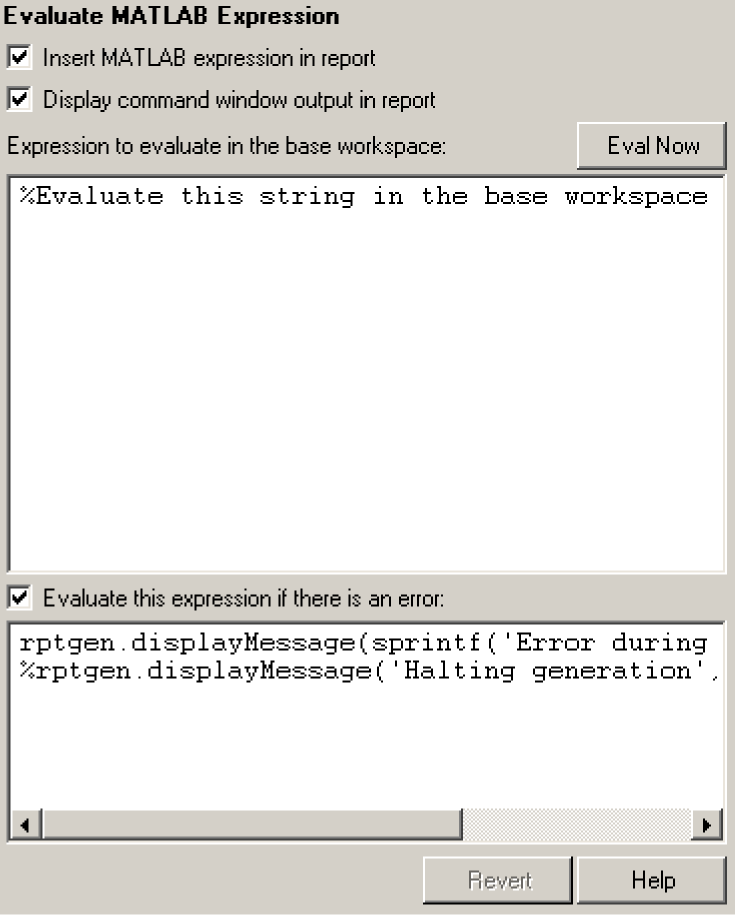
Note: You cannot edit the component information in the Properties pane on the right until you add the component to the report.

In the Outline pane on the left, the `Evaluate MATLAB Expression` component appears under the `simulink_tutorial` report setup file. The Simulink Report Generator software abbreviates the component name to `Eval`.



The icon in the upper-left corner of the `Eval` component's icon indicates that this component cannot have child components. By default, any components you add while the `Eval` component is selected are siblings of this component.

The options for the `Evaluate MATLAB Expression` component appear in the Properties pane on the right.



4 Clear the **Insert MATLAB expression in report** and the **Display command window output in report** check boxes so you do not include MATLAB code or output in this report.

5 Add MATLAB code to the **Expression to evaluate in the base workspace** text box to specify the following values:

- The model name
- The block name
- The block parameter

- Parameter values
- Other initial values required for processing the vdp model

Replace the existing text with the following MATLAB code.

```
%The name of the model
%that will be changed
expModel='vdp';

%The name of the block in the model
%that will be changed
expBlock='vdp/Mu';

%The name of the block parameter
%that will be changed
expParam='Gain';

%The values that will be set
%during experimentation
expValue=[-1 0 .5 1 2];

%expValue can be either a vector
%or a cell array

testMin=2.1;
testMax=3;

%---- do not change code below line ---

try
   open_system(expModel);
end

expOkValues=cell(0,2);
```

Note: When you change a field in the Properties pane on the right, the field background changes color (the default is a cream color), indicating that there are unapplied changes to that field. As soon as you perform operations on another component, the Simulink Report Generator software applies the changes, and the background color becomes white again.

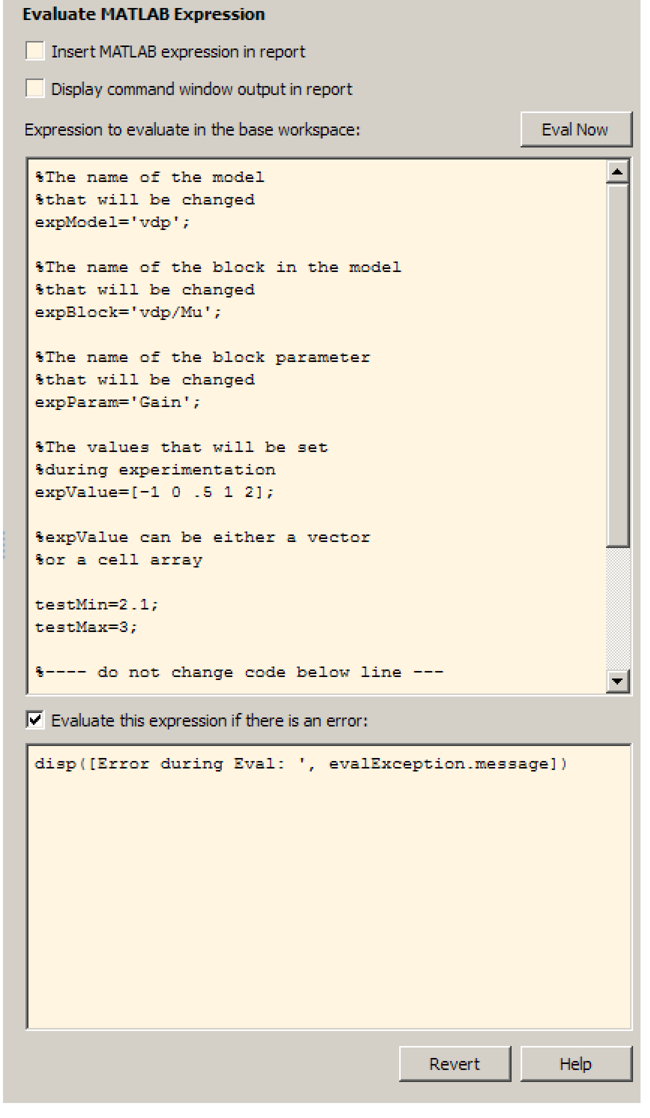
---

6. Select the **Evaluate this expression if there is an error** check box.

7 In the field under the check box, replace the existing text with the following text:

```
disp(['Error during eval: ', evalException.message])
```

The Report Explorer window now looks as follows.



Tip To run the commands that you specified in your MATLAB expression, click the **Eval Now** button. This button is located at the upper-right corner of the Report Explorer. This is an easy way to ensure that your commands are correct and will not cause report generation problems.

8 Click **File** > **Save** to save the report setup file.

For information about handling error conditions, see “Error Handling for MATLAB Code” on page 3-51.

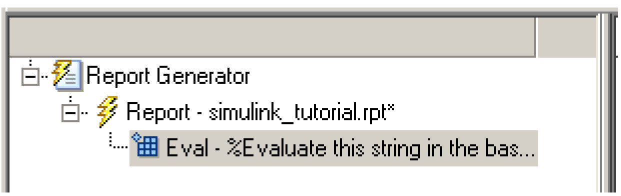
Add a Title Page

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

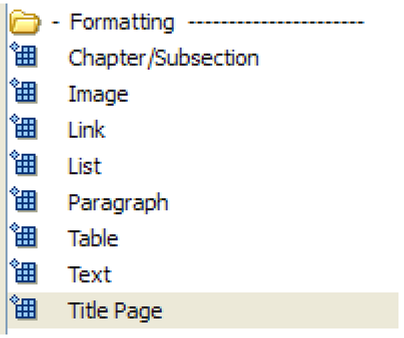
To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Create a custom title page for your report using the `Title Page` component.

1 In the Outline pane on the left, select the `Eval` component.

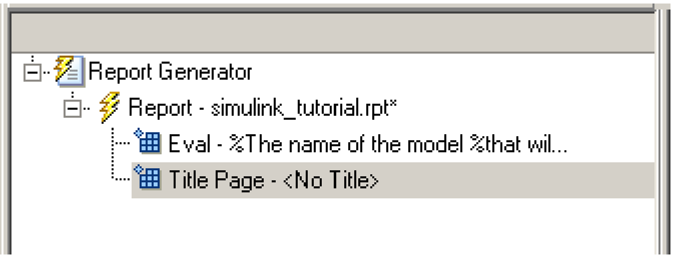


2 In the Library pane in the middle, under the `Formatting` category, click `Title Page`.



3 Click the icon next to **Add component to current report**.

The `Title Page` component appears in the Outline pane.



Note: To use the Title Page component, you need to have a Chapter component in your report . You have not yet added a Chapter component, so the Properties pane displays a message indicating that chapters are required for the Title Page component to appear correctly. Because later in this example you add Chapter components to this report, you can ignore that message.

---

4 In the Properties pane on the right:

a In the **Title** text box, enter:

```
Dynamic Simulink Report
```

b In the **Subtitle** text box, enter:

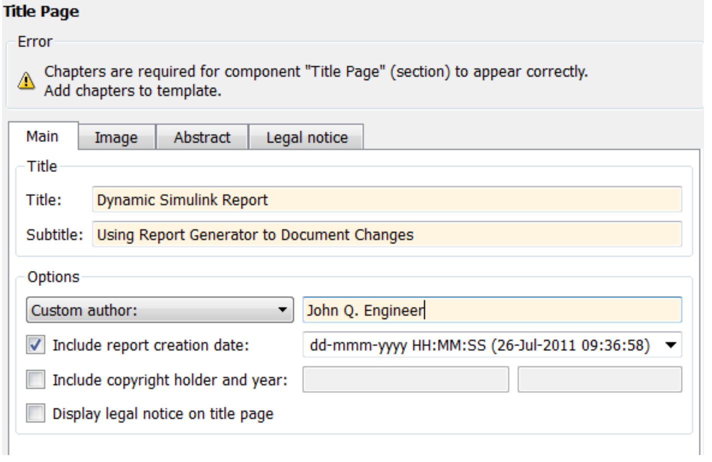
```
Using Simulink Report Generator to Document Changes
```

c In the **Options** section, choose `Custom Author` from the selection list.

d Enter your name in the text box.

e Select the **Include report creation date** check box.

f Select the default date and time format from the selection list. The Properties pane on the right looks as follows.



5 Save the report setup file.

Open the Simulink Model

Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

The following statement in the `Evaluate MATLAB Expression` component that you created in "Add MATLAB Code" on page 3-7 tries to open the vdp model:

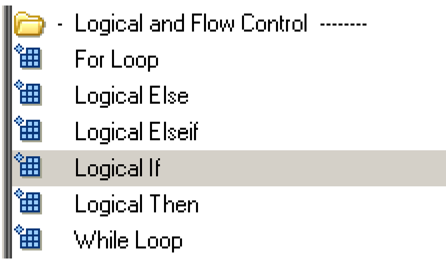
```
try
   open_system(expModel);
end
```

Tip Select the `Eval` component in the Outline pane on the left to look at this code again.

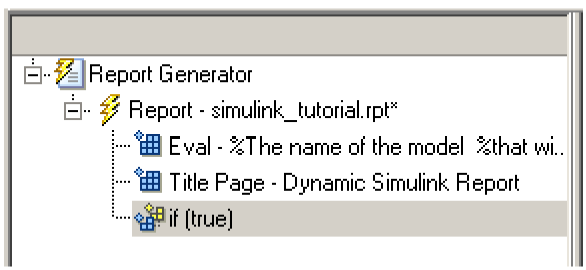
To see if the `vdp` model was successfully opened, test the result of the `open_system` command using a `Logical If` component.

1 In the Outline pane on the left, select the `Title Page` component.

2 In the Library pane in the middle, under the `Logical and Flow Control` category, select `Logical If`. This component checks to see if a given condition is true or false; in this case, if the model opened successfully.



3 In the Properties pane on the right, click the icon next to **Add component to current report**. The `Logical If` component appears as `if` in the Outline pane.



These components are child components of the report and siblings of one another. Components can have parent, child, and sibling relationships.

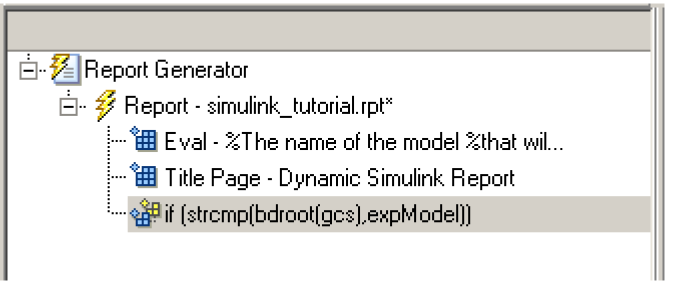
This component can have child components. "Add Logical Then and Logical Else Components" on page 3-16 explains how to add two child components to the `if` component.

4 In the Properties pane on the right, in the **Test expression** text box, replace the default text, `true`, with the following text:

`strcmp(bdroot(gcs),expModel)`
The `strcmp` function compares the name of the open Simulink model and the value of `expModel`, which was set to `'vdp'`. It tests to see if the vdp model opened successfully. `strcmp` returns `1` (`true`) if the two strings match, and `0` (`false`) if not.

5 Save the report setup file.

The `if` component name in the Outline pane changes to include the expression that you added.



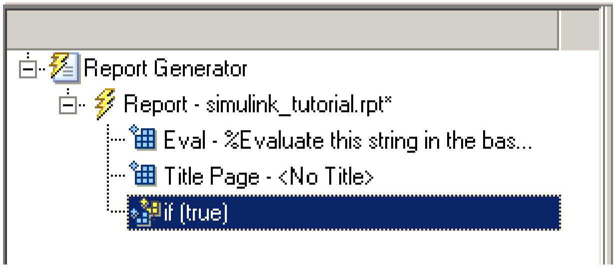
Add Logical Then and Logical Else Components

Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

The `if strcmp(bdroot(gcs), expModel))` component has two possible results. Add two child components to the report setup file to process these cases.

1 In the Outline pane on the left, select the `if` component.

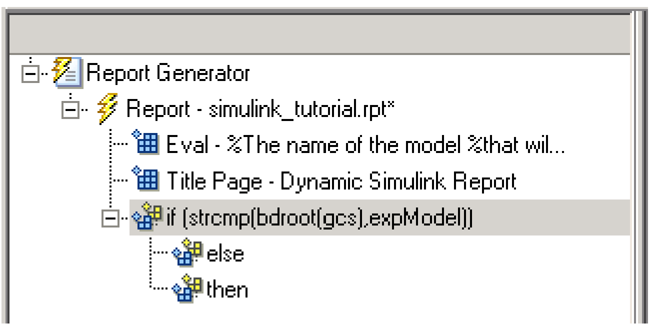


2 In the Library pane in the middle, under the `Logical and Flow Control` category, double-click `Logical Then`.

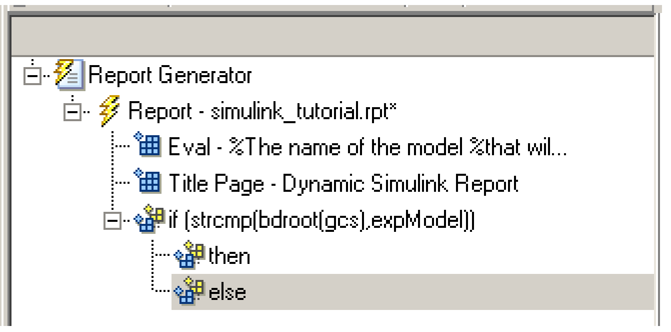
3 In the Outline pane on the left, select the `if` component again.

4 In the Library pane in the middle, under the `Logical and Flow Control` category, double-click `Logical Else`.

   Both elements are added as child components to the if component, as shown in the Outline pane.



5 To move the else component under the `then` component, select the else component and click the **down** arrow on the toolbar once. The Outline pane on the left looks as follows.



6 Save the report setup file.

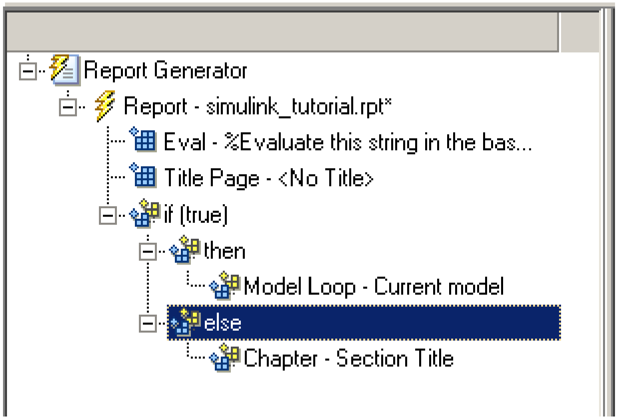
Error If Model Cannot Be Opened

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

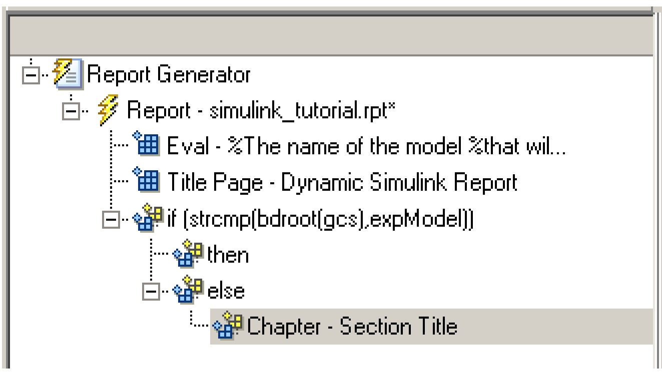
To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

If the `if strcmp(bdroot(gcs), expModel))` component fails (the vdp model cannot open), the `else` component executes. Display an error message in the report using the `Chapter/Subsection` component.

1 In the Outline pane on the left, select the `else` component.



2 In the Library pane in the middle, under the `Formatting` category, double-click `Chapter/Subsection` to add it as a child of the `else` component. This component displays an error message if an error occurs when opening the vdp model.



Note: When you add a component to a report, it is added by default as a child component unless the selected component cannot have child components.

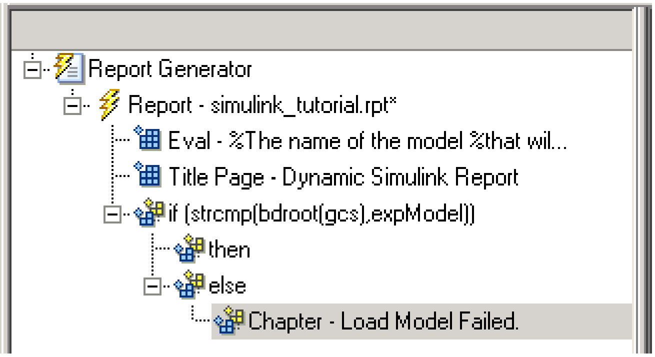
---

3 In the Properties pane on the right, choose `Custom` from the **Title** selection list, and then enter the following text in the text box:

```
Load Model Failed.
```

Save the report file.

The Outline pane looks as follows.



4 In the Outline pane on the left, select the `Chapter` component.

5 In the Library pane in the middle, under `Formatting`, double-click `Paragraph`.

6 In the Properties pane on the right, enter the following text in the **Paragraph Text** text box to display the following error message:

```
Error: Model %<expModel> could not be opened.
```

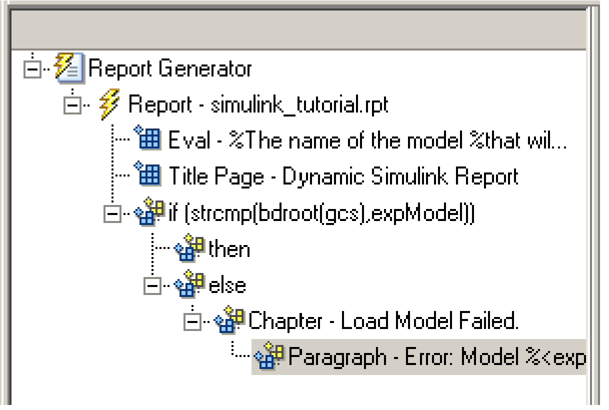
The expression `%<expModel>` indicates that the value of the workspace variable `expModel` is inserted into the text, as in the following example.

```
Error: Model vdp could not be opened.
```

7 In the Outline pane on the left, select the `Chapter/Section` component.

8 Save the report setup file.

The Outline pane looks as follows.



Create the Body of the Report

Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Creating the body of the report involves setting up components and code for dynamic execution of report components. In this example, you perform the following tasks:

- "Process with a Model Loop Component" on page 3-22
- "Add a Paragraph for Each Model" on page 3-24
- "Insert a Snapshot of the Model" on page 3-25
- "Add a Loop for Processing the Model" on page 3-26
- "Block Parameter Value from a MATLAB Expression" on page 3-28
- "Create a Section for Each Iteration" on page 3-29
- "Insert the Block Value" on page 3-31
- "Set a Parameter Value" on page 3-32
- "Check Value Using a Logical If Component" on page 3-34

- “Simulate the Model Using a Model Simulation Component” on page 3-37
- “Create a Post-Test Analysis Section” on page 3-43

Each action requires a separate component under the `then` component. For information about the then component in this report, see “Add Logical Then and Logical Else Components” on page 3-16.

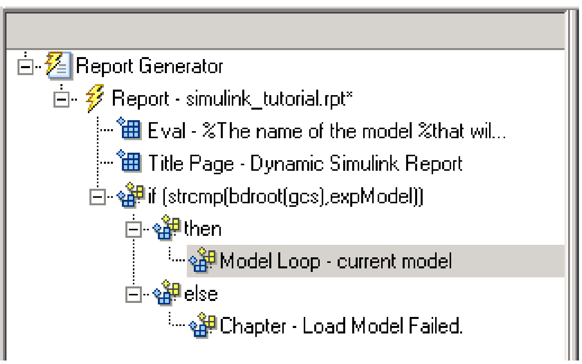
Process with a Model Loop Component

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

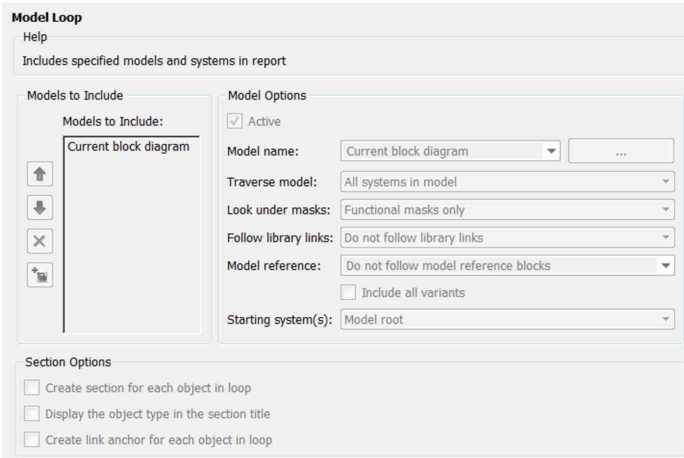
To see the completed report setup file, open `Simulink Dynamic Report`.

The report changes the `Gain` parameter for the `Mu` block in the vdp model several times. This task requires a `Model Loop` component.

1. In the Outline pane on the left, select the `then` component.
2. In the Library pane in the middle, scroll down to the `Simulink` category, and then double-click `Model Loop`. It is added as a child of the `then` component.



The Properties pane on the right looks as follows.



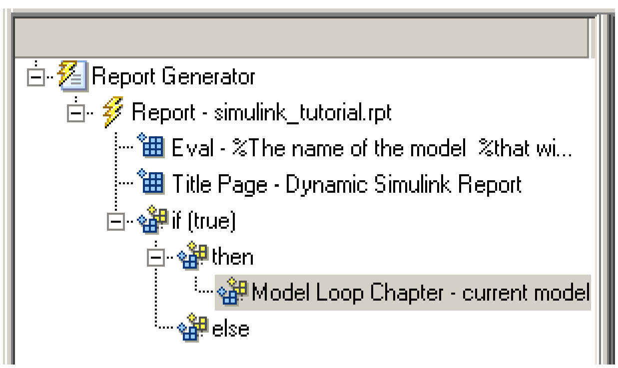
3 In the Properties pane on the right:

a Select the **Active** check box to process the vdp model.

b In the **Traverse model** selection list, select `Selected system(s) only` to traverse only the vdp model.

c Select `Model root` from the **Starting system(s)** selection list.

d At the bottom of the Properties pane on the left, select the **Create section for each object in loop** check box to create a chapter or section for each model. When you select this check box, the component name in the Outline pane on the left changes to `Model Loop Chapter`.



e Select the **Display the object type in the section title** check box to include the object type (in this example, model) in the title name.

f Clear the **Create link anchor for each object in loop** check box.

4 Save the report setup file.

Add a Paragraph for Each Model

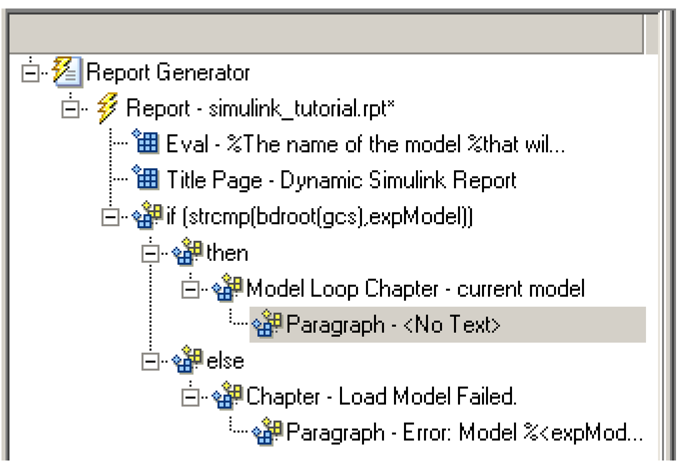
Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

In each Model Loop Chapter, add an explanation using the `Paragraph` component.

1 In the Outline pane on the left, select the `Model Loop Chapter` component.

2 In the Library pane in the middle, scroll up to the `Formatting` category, and then double-click `Paragraph`. The `Paragraph` component is added as a child of the `Model Loop Chapter` component.



3 In the Properties pane on the right, in the **Paragraph Text** text box, enter the following text:

```
This report demonstrates Simulink Report Generator's ability
to experiment with Simulink systems and auto-document
the results. In this report, you load the model %<expModel>
and simulate it %<length> times. This report modifies the
%<expBlock> block's "%<expParam>" value, setting it to the
values %<expValue>. Each iteration of the test includes
a set of scope snapsnots in the report.
```

When this report is generated, the variable names preceded by percent signs (%) and enclosed in brackets (<>) are replaced with the values of those variables in the MATLAB workspace.

4 Save the report setup file.

Insert a Snapshot of the Model

Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

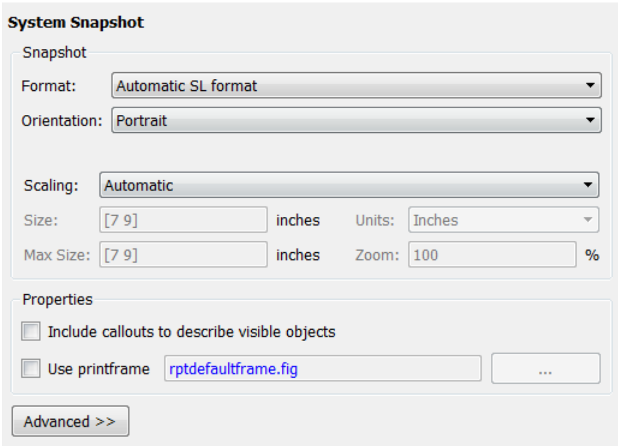
To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Inside each `Model Loop Chapter` component, include a snapshot of the current model using the `System Snapshot` component.

1 In the Outline pane on the left, select the `Model Loop Chapter` component.

2 In the Library pane in the middle, scroll down to the `Simulink` category, and then double-click the `System Snapshot` component.

   This component inserts an image of the current model into your report. The Properties pane on the right looks as follows.



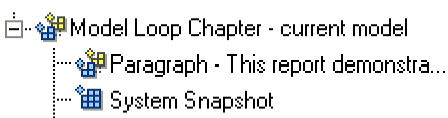
3 In the Properties pane on the right:

   a Select `Zoom` from the **Scaling** selection list.

   b Enter `70` as the % value.

4 In the Outline pane on the left, select the `System Snapshot` component.

5 Click the **down** arrow on the toolbar once to move it under the `Paragraph` component.



6 Save the report setup file.

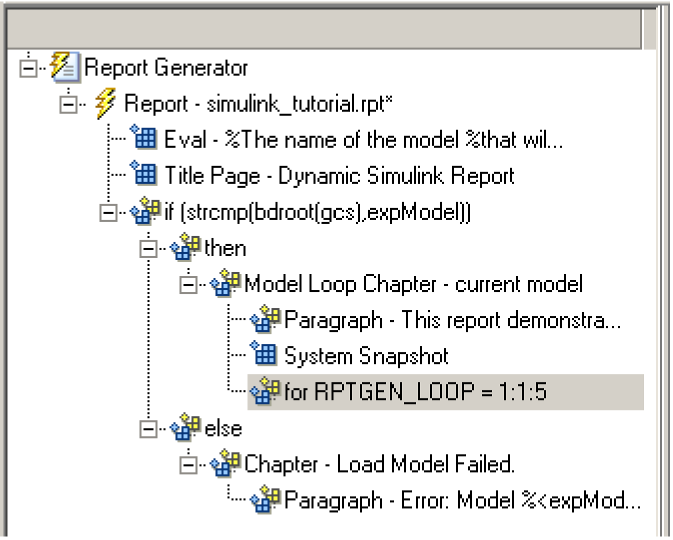
Add a Loop for Processing the Model

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Create a loop to process the model `%length` times using the `For Loop` component.

1. In the Outline pane on the left, select the `System Snapshot` component.
2. In the Library pane in the middle, under the `Logical and Flow Control` category, double-click `For Loop`. The `For Loop` component is added as a sibling of the `System Snapshot` component.



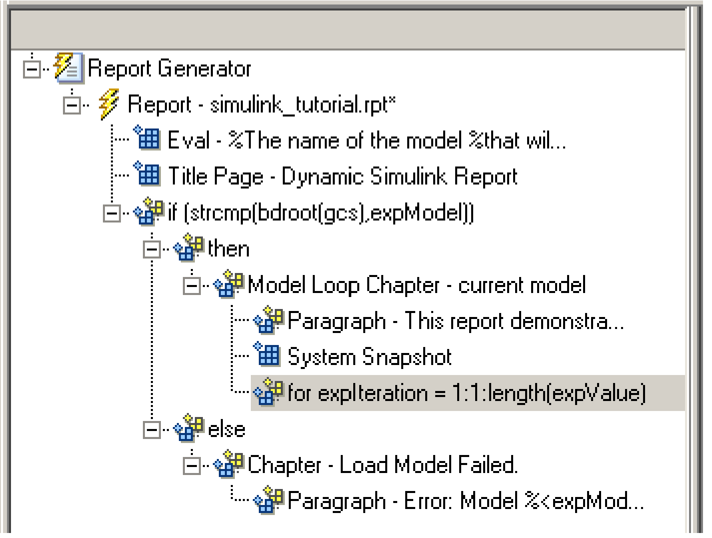
3. In the Properties pane on the right:

   a. In the **End** text box, replace the existing text with the following text:

      ```
      length(expValue)
      ```
      `expValue` is the array of `Gain` parameter values assigned in the `Eval` component with the command `expValue=[-1 0 0.5 1 2];`. The expression `length(expValue)` evaluates to `5` in this example.

   b. In the **Variable name** text box, replace the existing text with the name of the `for loop` variable. Enter the following text:

      ```
      expIteration
      ```
      The name of the `For` component in the Outline pane on the left changes to reflect the loop variable and the termination value.



4 Save the report setup file.

Block Parameter Value from a MATLAB Expression

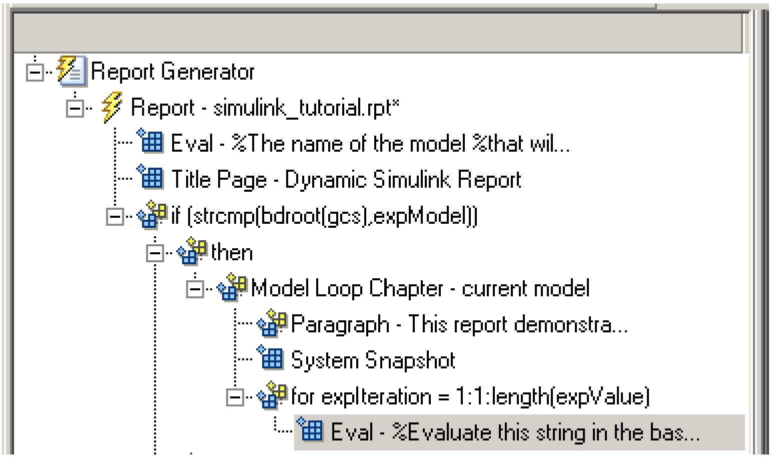
Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

For each iteration, get a value from the `expValue` array to use as the `Gain` parameter value. This task requires an `Evaluate MATLAB Expression` component.

1 In the Outline pane on the left, select the `for` component.

2 In the Library pane in the middle, under the `MATLAB` category, double-click `Evaluate MATLAB Expression`. In the Outline pane, the component name is shortened to `Eval`.



3 On the Properties pane on the right:

a Clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes.

b Enter the following text in the **Expression to evaluate in the base workspace** text box:

```
%Evaluate this string in the base workspace

if iscell(expValue)
   Iteration_Value=expValue{expIteration};
else
  Iteration_Value=...
     num2str(expValue(expIteration));
end
```

The `Iteration_Value` variable represents the designated array element.

c Clear the **Evaluate expression if there is an error** check box.

4 Save the report setup file.

Create a Section for Each Iteration

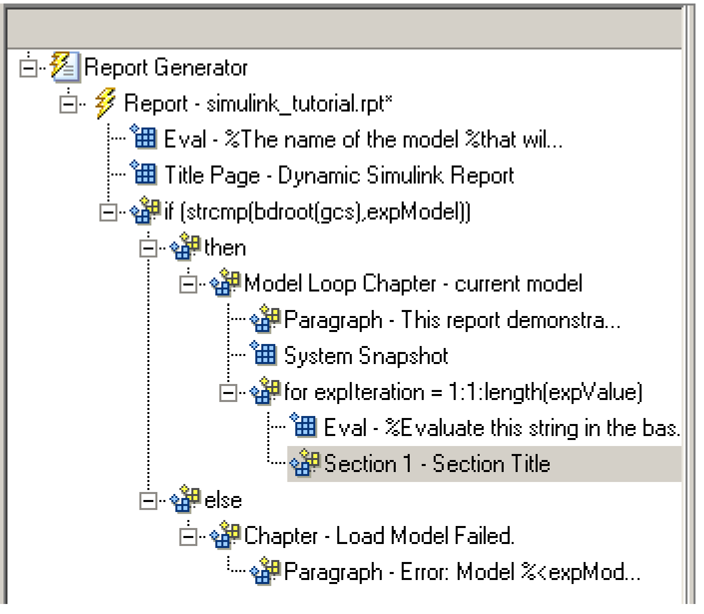
Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Create a separate section for each iteration of the loop that includes the data using the `Chapter/Subsection` component.

1 In the Outline pane on the left, under the `for` component, select the `Eval` component.

2 In the Library pane in the middle, under the `Formatting` category, double-click the `Chapter/Subsection` component to add it as a sibling. This component is automatically added as `Section 1` because it is inside a `Chapter` component (the `Model Loop Chapter` component).



3 In the Properties pane on the right:

a In the **Title** selection list, select `Custom`.

b In the text box, enter the following title:

```
Processing the vdp model
```

This indicates that the section title comes from the first child component. Do not change any other properties.

4 Save the report setup file.

Insert the Block Value

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Insert the `Gain` value that is used for each simulation.

1 In the Outline pane on the left, select the `Section 1` component.

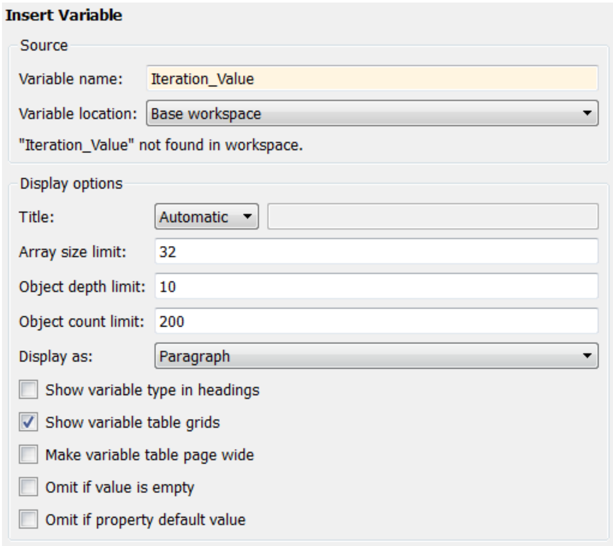
2 In the Library pane in the middle, under the MATLAB category, double-click `Insert Variable`.

3 In the Properties pane on the right:

   a In the **Variable name** text box, enter `Iteration_Value`.

   b In the **Display as** selection list, select `Paragraph`.

   The Properties pane on the right looks as follows.



4 Save the report setup file.

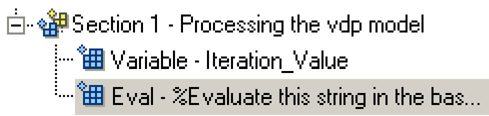
Set a Parameter Value

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

For each iteration, set the `Gain` parameter to the value that you extracted from the `expValue` array.

1 In the Outline pane on the left, select the `Variable` component.
2 In the Library pane in the middle, under the `MATLAB` category, double-click `Evaluate MATLAB Expression`. This component is added as a sibling of the `Variable` component.



3 In the Properties pane on the right, clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes.

4 In the **Expression to evaluate in the base workspace** text box, replace the existing text with the following text.

```
set_param(expBlock,expParam,Iteration_Value);
okSetValue=(1);
```

The `set_param` command sets the value of the `Gain` parameter for the `Mu` block in the vdp model to the value of `Iteration_Value`.

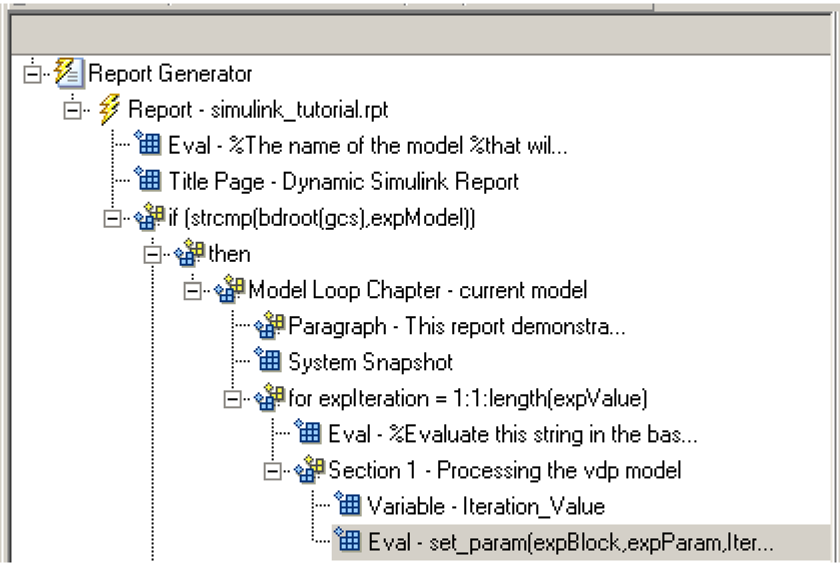
5 Make sure you select **Evaluate expression if there is an error**. Enter the following text into the text box:

```
okSetValue=logical(0);
```

If the `set_param` command works, `okSetValue` is set to `1`. If an error occurs, `okSetValue` is set to `0`. The next component then reports the error and terminates processing.

6 Save the report setup file.

The Outline pane on the left looks as follows.



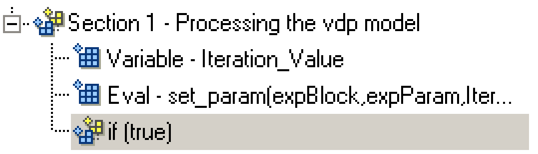
Check Value Using a Logical If Component

Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Check the value of `okSetValue` using a `Logical If` component. If the value is `0`, the simulation cannot proceed because the `Gain` parameter could not be set.

1. In the Outline pane on the left, select the `Eval` component for the `set_param` command.
2. In the Library pane in the middle, under the `Logical and Flow Control` category, double-click `Logical If`. The component is added as a sibling of `Eval`.



3 In the Properties pane on the right, in the **Test expression** text box, replace `true` with `okSetValue`.

`okSetValue` can be `1` (`true`) or `0` (`false`), so insert two components — `Logical Then` and `Logical Else` — to process those conditions:

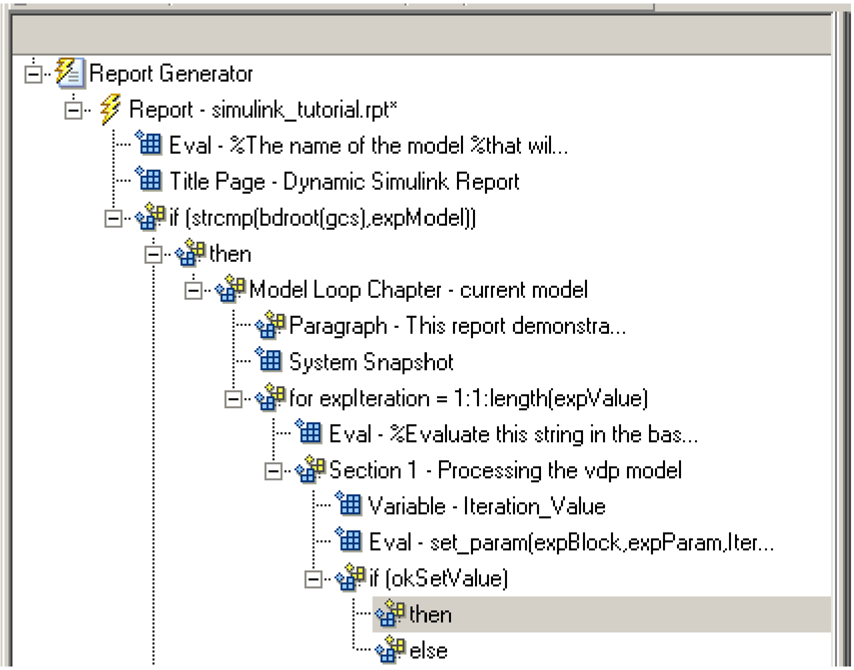
1 In the Outline pane on the left, select the `if(okSetValue)` component.

2 To insert `Logical Then` and `Logical Else` in the correct order:

   a In the Library pane in the middle, double-click the `Logical Else` component.

   b Select the `if(okSetValue)` component again.

   c Double-click the `Logical Then` component. The Outline pane on the left looks as follows.



3 In the Outline pane on the right, select the `else` component.

4 In the Library pane in the middle, double-click `Paragraph`.

   If `okSetValue` = `0`, the `Gain` parameter value is not set and the report displays an error.

5 In the Properties pane on the right:

   a Choose `Custom title` from the **Title Options** selection list.

   b Enter `Error` in the text box next to the selection list.

   c Enter the following text into the **Paragraph Text** text box:

```
Could not set %<expBlock> "%<expParam>" to value
%<Iteration_Value>.
```

6 Save the report.

Simulate the Model Using a Model Simulation Component

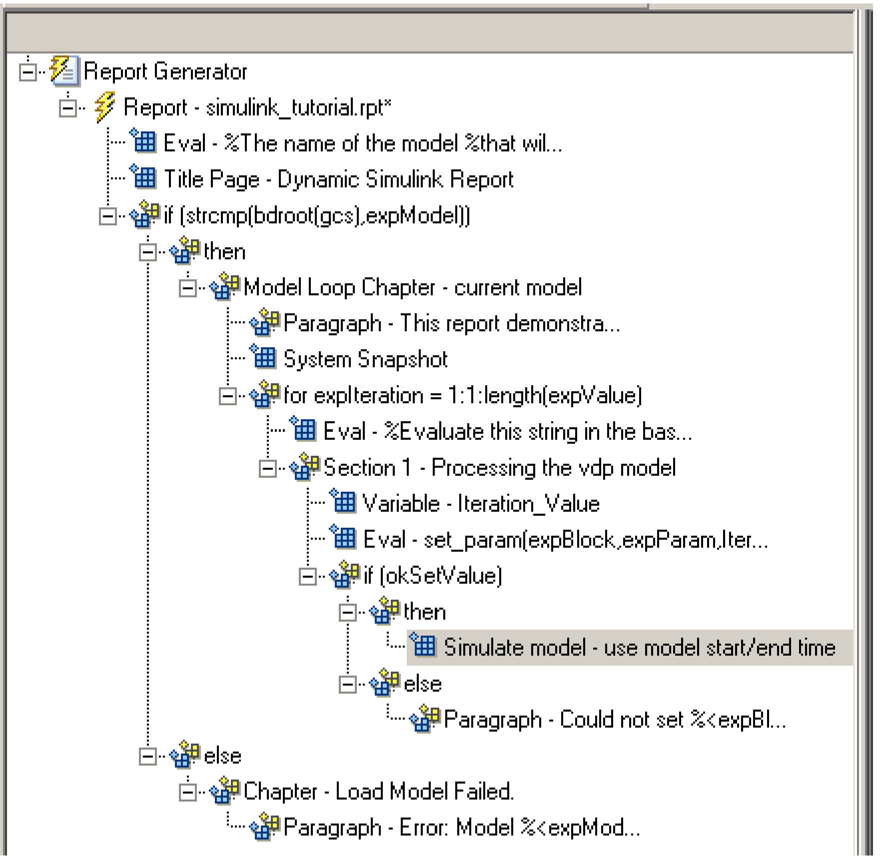
Note: This section builds on the previous tasks described in the step-by-step example summarized in "Create a Simulink Report Generator Report".

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

Now that the model is open and the `Gain` parameter is set, use the `Model Simulation` component to simulate the vdp model.

1 In the Outline pane on the left, select the `then` component under the `if (okSetValue)` component.

2 In the Library pane, under the `Simulink` category, double-click `Model Simulation`. In the Outline pane on the left, this component is renamed `Simulate model`.



3 In the Properties pane on the right:

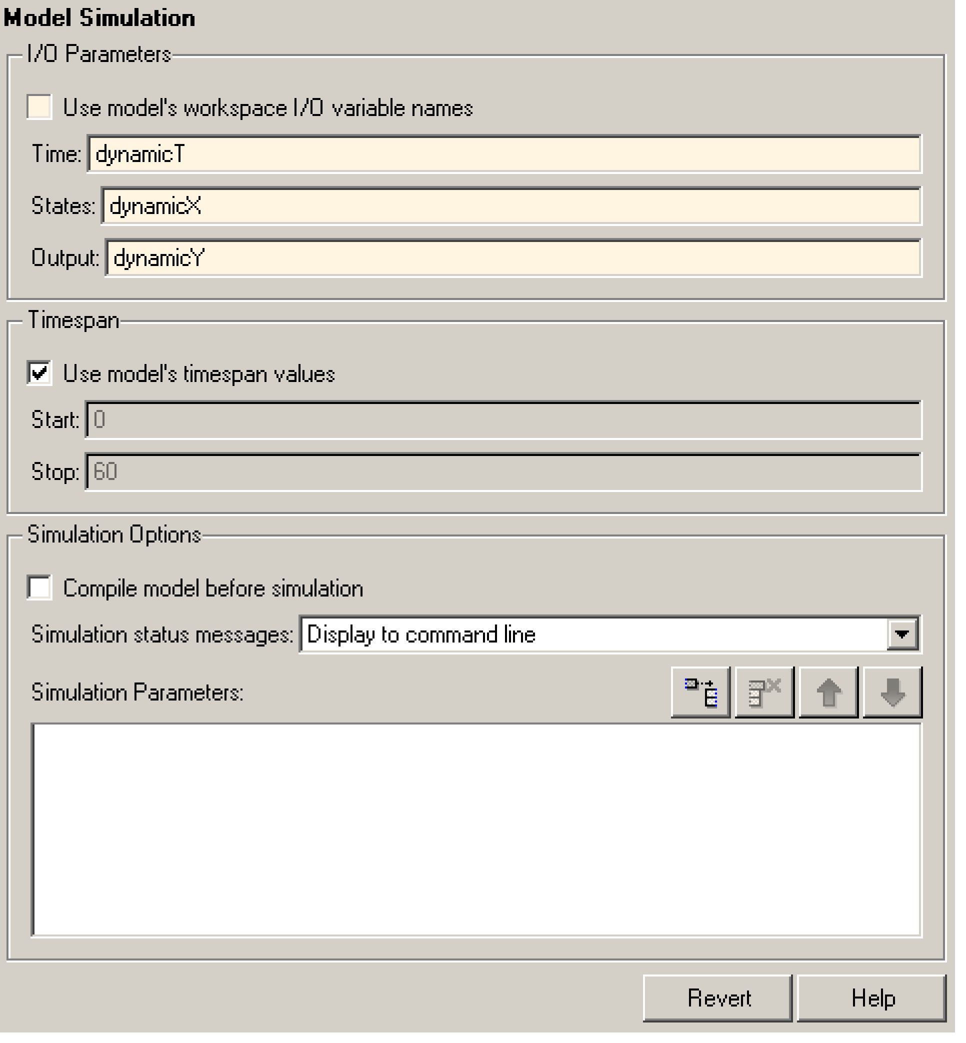
a Clear the **Use model's workspace I/O variable names** check box.

b In the **Time** text box, enter `dynamicT`.

c In the **States** text box, enter `dynamicX`.

d In the **Output** text box, enter `dynamicY`.

The Properties pane on the right looks as follows.



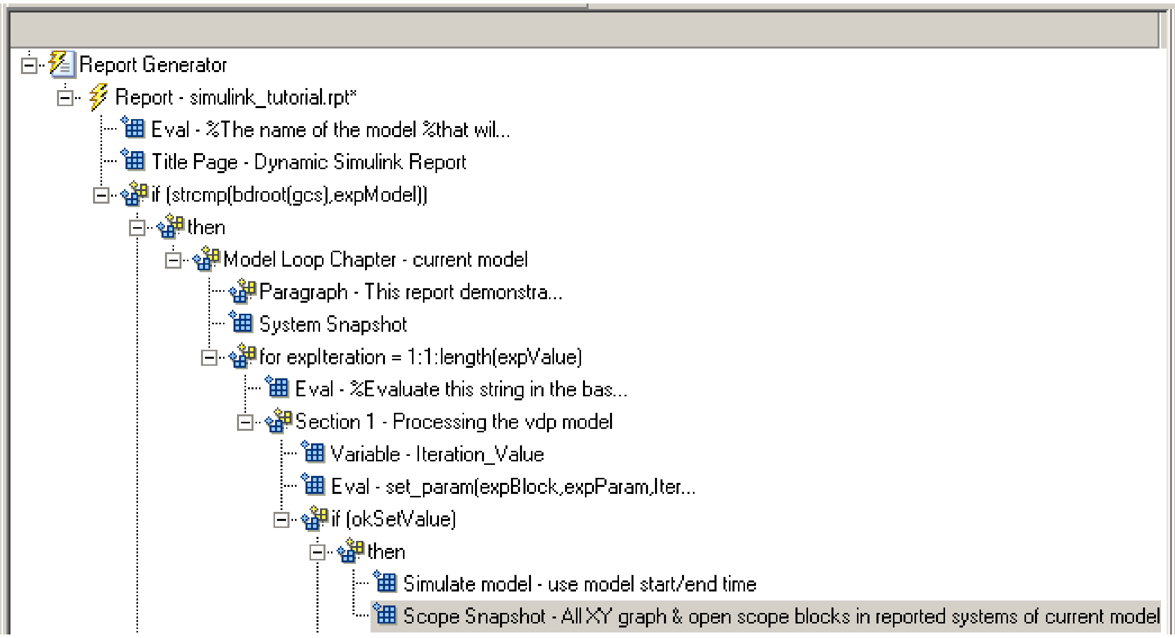
4 In the Outline pane on the left, select the `Simulate model` component.

5 In the Library pane in the middle:

  a Scroll down to the `Simulink Blocks` category.

  b Double-click `Scope Snapshot` to add it as a sibling of the `Simulink Model` component.

This component captures the scope for each iteration.



6 In the Properties pane on the right:

   a In the **Paper orientation** selection list, select `Portrait`.

   b For the **Image size**, enter `[5 4]`.

   c In the **Scaling** selection list, select `Zoom`.

   d Enter `75` for the % value.

7 Save the report setup file.

8 To test to see if the signal data falls within a specified range, add another `Logical If` component:

   a In the Outline pane on the left, select the `Scope Snapshot` component.

   b In the Library pane in the middle, scroll up to the `Logical and Flow Control` category.

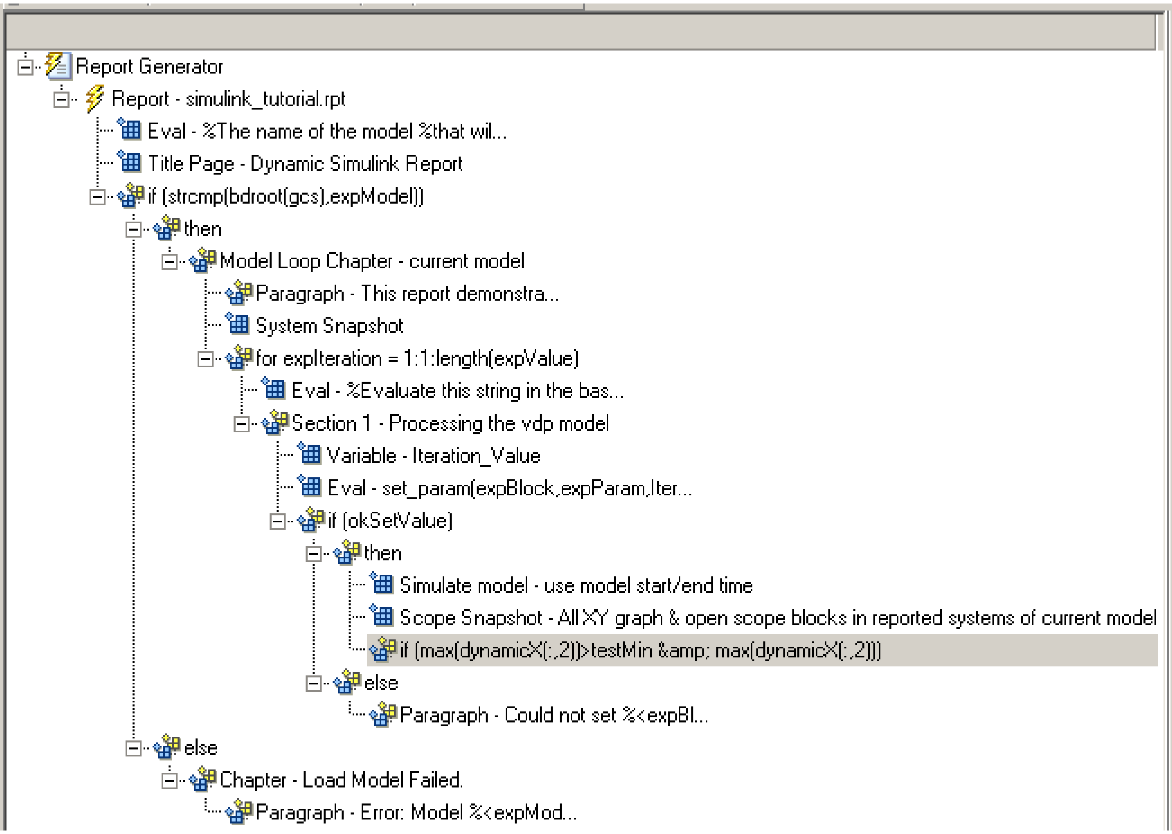
   c Double-click the `Logical If` component.

9 To test the signal data, replace `true` in the **Test expression** text box with the following in the Properties pane on the right:

```
max(dynamicX(:,2))>testMin & max(dynamicX(:,2))
```

10 Save the report.

The Outline pane looks as follows:



11 If this condition is true, the signal data falls within the desired range. Add a `Paragraph` component to print information about the signal data in the report.

a In the Outline pane on the left, select the `if` component you just added.

b In the Library pane in the middle, under the `Formatting` category, double-click `Paragraph` so that it becomes a child of the `if` component.

c In the Properties pane on the right:

i From the **Title Options** selection list, select `Custom title`.

ii Type `Success` in the text box.

iii Enter the following text in the **Paragraph text** text box.

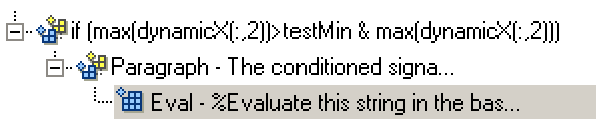
```
The conditioned signal has a maximum value
of %<max(dynamicX(:,2))>, which lies in the
desired range of greater than %<testMin> and
less than %<testMax>.
```

12 To save the success values to insert into a table at the end of the iterations, use an `Evaluate MATLAB Expression` component.

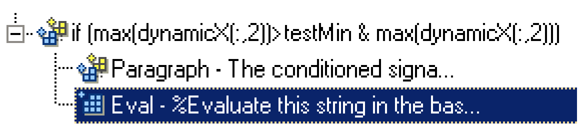
a In the Outline pane on the left, select the `Paragraph` component.

b In the Library pane in the middle, under the `MATLAB` category, double-click `Evaluate MATLAB Expression`.

An unintended result occurs: the new component is a child of the `Paragraph` component.



c To make the new component a *sibling* of the `Paragraph` component, in the Outline pane on the left, select the `Eval` component, and then Click the left arrow on the toolbar. The `Eval` component becomes a sibling of the `Paragraph` component.



13 In the Properties pane on the right, for the `Eval` component:

a. Clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes.

b. In the **Expression to evaluate in the base workspace** text box, enter the following to save the desired signal values in the `expOkValues` array:

```
expOkValues=[expOkValues;...
     {Iteration_Value,max(dynamicX(:,2))}];
```

c. Make sure you select **Evaluate this expression if there is an error**. Insert the following text in the text box:

```
disp(['Error during eval: ', evalException.message])
```

14. Save the report setup file.

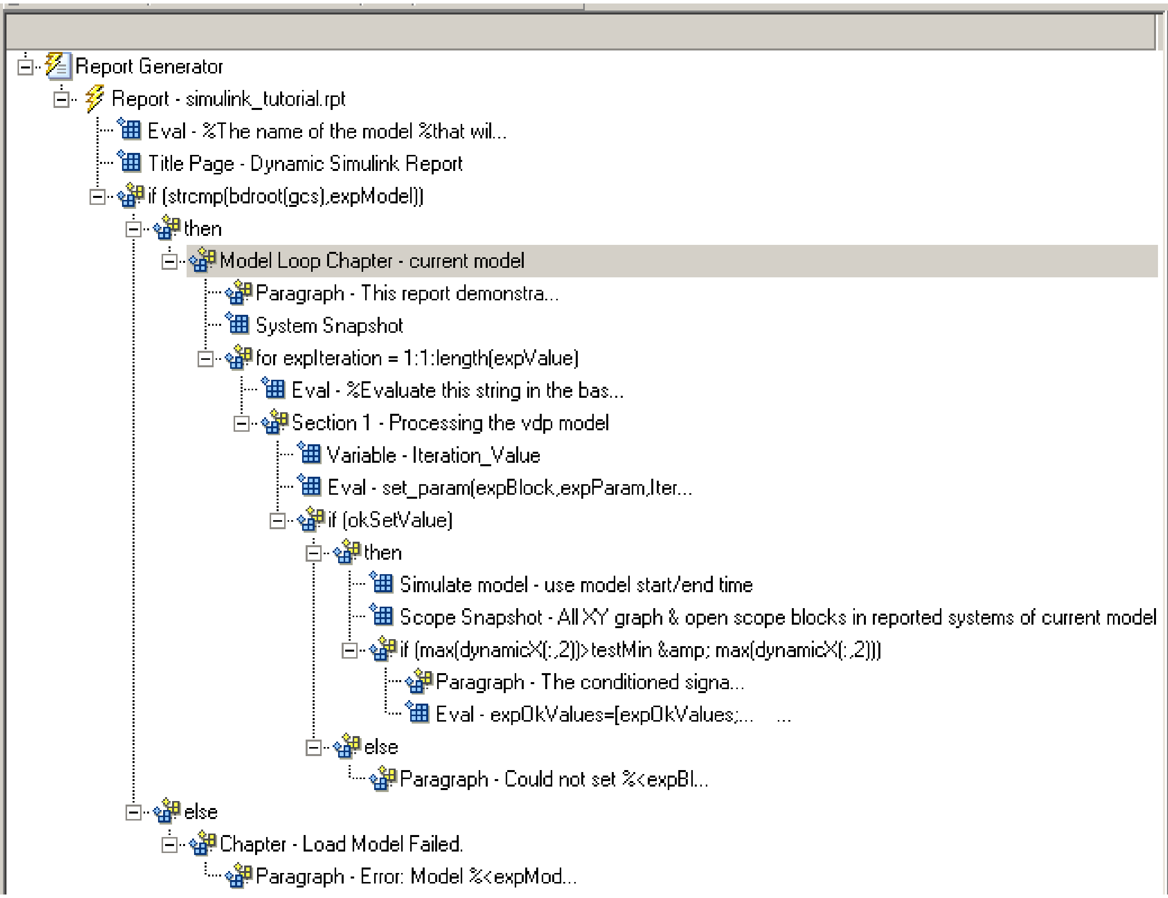
Create a Post-Test Analysis Section

Note: This section builds on the previous tasks described in the step-by-step example summarized in “Create a Simulink Report Generator Report”.

To see the completed report setup file, open `Simulink Dynamic Report`. The report is for the `vdp` model.

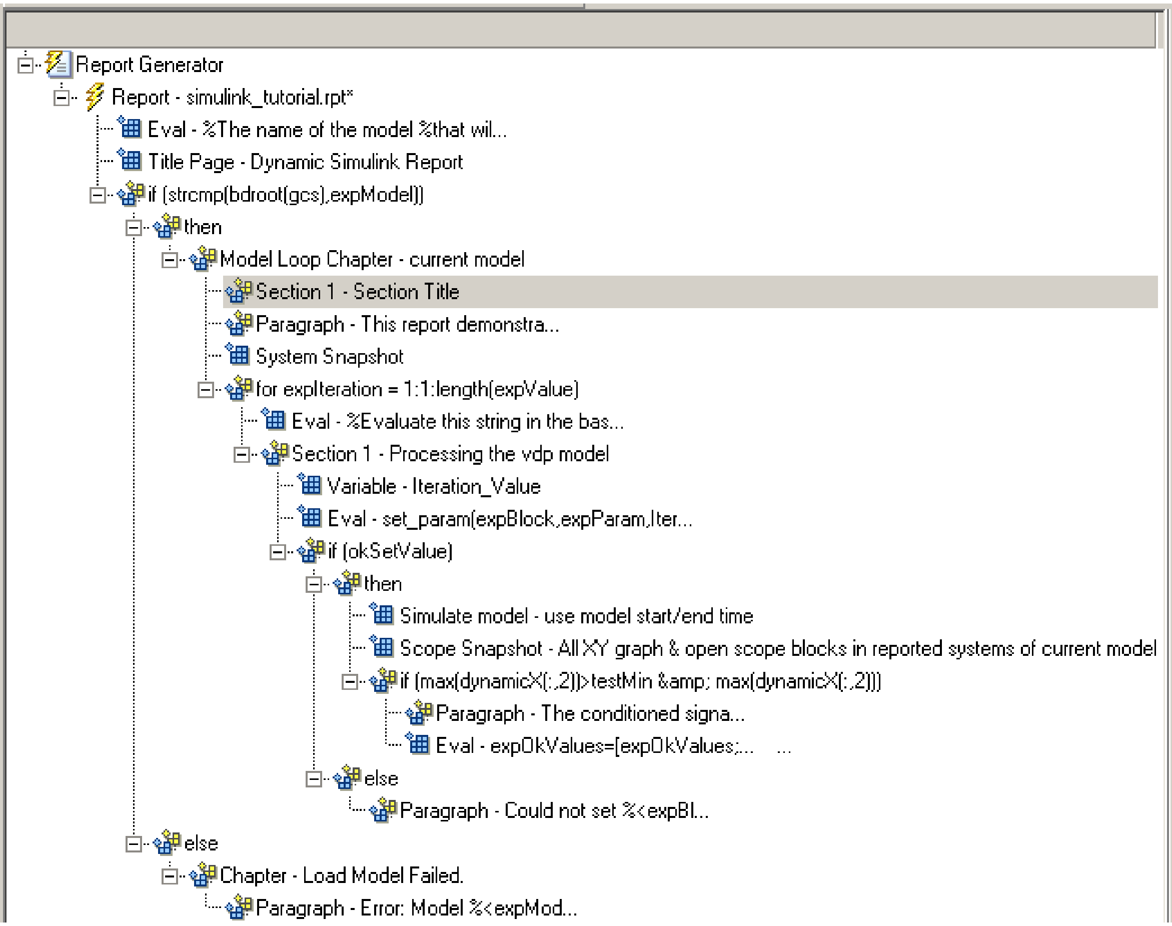
Now that you have collected all the desired values, create the post-test analysis section by creating a table and inserting it into your report at the end of this chapter.

1. In the Outline pane on the left, select the `Model Loop Chapter` component.

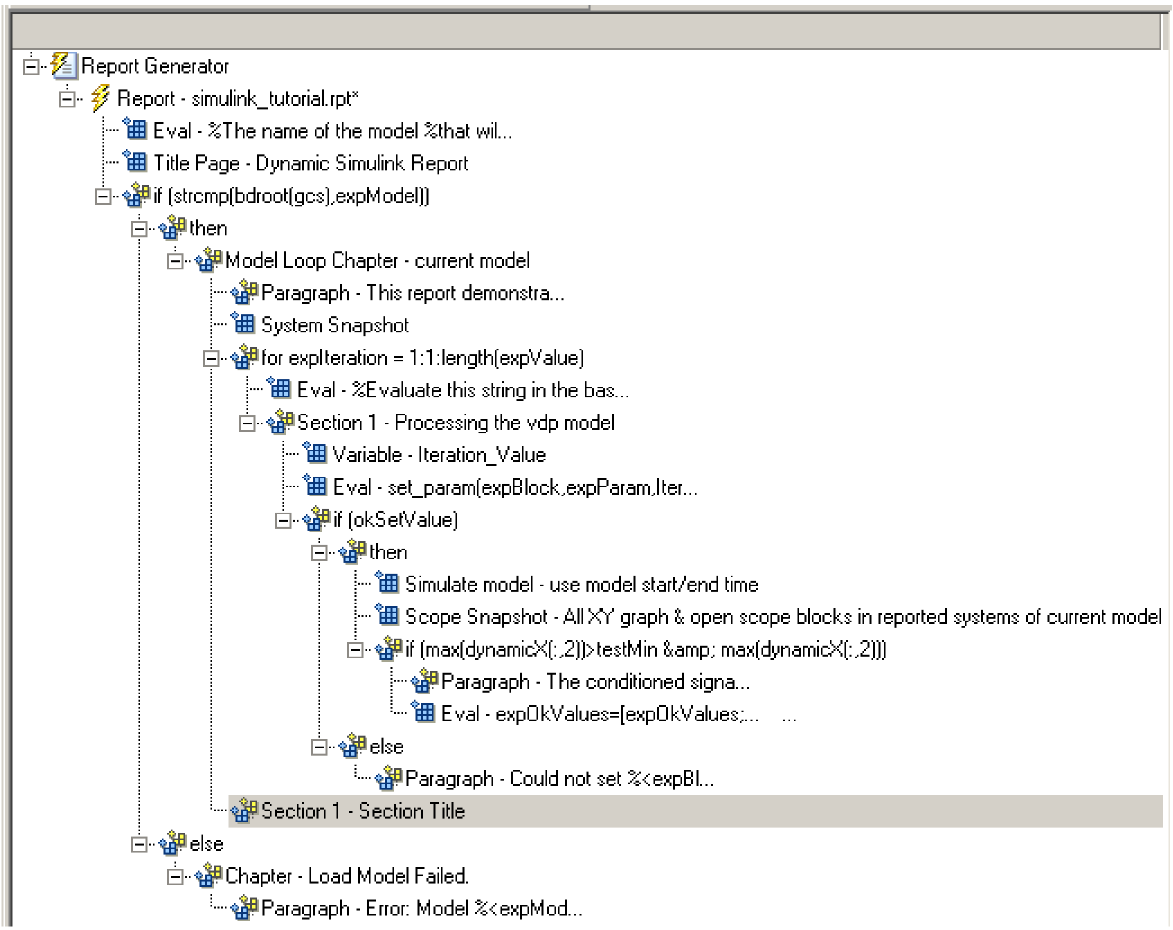


2 In the Library pane in the middle, under the `Formatting` category, double-click `Chapter/Subsection`.

The new section appears at the beginning of the chapter.



Click the **down** arrow three times so `Section 1` moves to the end of the `Model Loop Chapter` component.



3 In the Properties pane on the right:

  a Select `Custom` in the **Title** selection list.

  b Enter `Post-Test Analysis` in the text box.

4 In the Outline pane on the left, select the new `Section 1` component.

5 In the Library pane in the middle, under the `Formatting` category, double-click `Paragraph`. Do not change its properties.

6 To check whether there are any signal values within the desired range, check the array `expOkValues` with a `Logical If` component. If `expOkValues` is empty, there are no signal values in the desired range. Report the result of this check.

  a In the Outline pane on the left, select the `Paragraph` component and add a `Logical If` child component.

  b In the Properties pane on the right, enter the expression to evaluate in the **Test expression** text box:

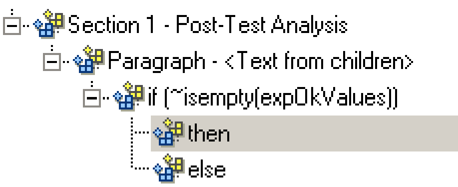
```
~isempty(expOkValues)
```

  This expression evaluates to `0` (`false`) if `expOkValues` is empty; otherwise, it evaluates to `1` (`true`).

  c In the Outline pane on the left, select the `if(~isempty(expOkValue))` component and add the `Logical Else` component as a child.

  d Select the `if(~isempty(expOkValue))` component again and add the `Logical Then` component as a child.

  The two components are siblings in the Outline pane on the left.



7 Save the report setup file.

8 Now, insert report components to handle the case where `expOkValues` is empty; that is, where no signal values fall within the designated range.

  a In the Outline pane on the left, select the `else` component.

  b In the Library pane in the middle, double-click the `Text` component to add it as a child of the `else` component.

  c In the Properties pane on the right, in the **Text to include in report** text box, enter the following:

```
None of the selected iteration values had
a maximum signal value between %<testMin> and %<testMax>.
```

9 Now handle the case where `expOkValues` is not empty and you want to insert a table of the acceptable signal values.

a In the Outline pane on the left, select the `then` component.

b Add a `Text` component as a child to the `then` component.

c In the Properties pane on the right, in the **Text to include in report** text box, enter the following text.

```
%<size(expOkValues, 1)> values for %<expBlock> were
found that resulted in a maximum signal value greater
than %<testMin> but less than %<testMax>. The following
table shows those values and their resulting signal maximum.
```

d In the Outline pane on the left, select the `Text` component under the `then` component of the `if(~isempty(expOkValues)` component.

10 To create an array for use when formatting the table, use the `Evaluate MATLAB Expression` component.

a In the Library pane in the middle, double-click `Evaluate MATLAB Expression`.

b In the Properties pane on the right:

i Clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes.

ii The next component of the report uses the strings `Mu Value` and `Signal Maximum` as table header values. Add the strings to the front of the `expOkValues` cell array by entering the following text into the **Expression to evaluate in the base workspace** text box:

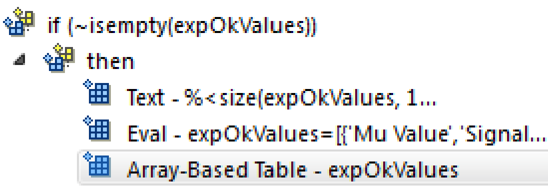
```
expOkValues=[{'Mu Value','Signal Maximum'} expOkValues];
```

iii Make sure you select the **Evaluate this expression if there is an error** check box. Enter the following text into the text box:

```
disp(['Error during eval: ', evalExpression.message])
```

11 In the Outline pane on the left, select the `Eval` component.

12 In the Library pane in the middle, under the `Formatting` category, double-click the `Table` component so it becomes a sibling of the `Text` and `Eval` components.



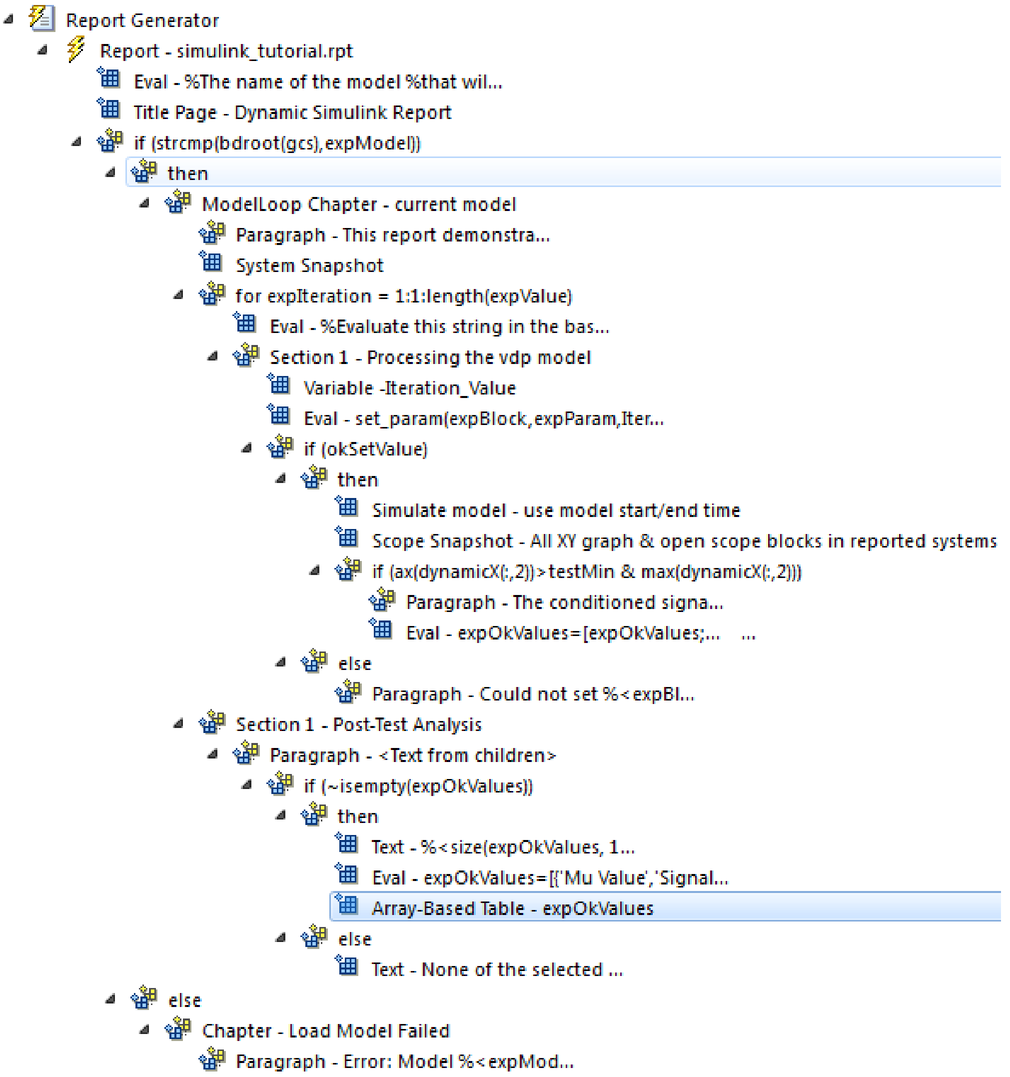
13 In the Properties pane on the right:

a In the **Workspace variable name** text box, enter `expOkValues`. The Simulink Report Generator software uses the contents of `expOkValues` to construct the table.

b In the **Table title** text box, enter `Valid Iteration Values`.

14 Save the report setup file.

The Outline pane on the left looks as follows.


Report Generator
Report - simulink_tutorial.rpt
Eval - %The name of the model %that wil...
Title Page - Dynamic Simulink Report
if (strcmp(bdroot(gcs),expModel))
then
ModelLoop Chapter - current model
Paragraph - This report demonstra...
System Snapshot
for expIteration = 1:1:length(expValue)
Eval - %Evaluate this string in the bas...
Section 1 - Processing the vdp model
Variable -Iteration_Value
Eval - set_param(expBlock,expParam,Iter...
if (okSetValue)
then
Simulate model - use model start/end time
Scope Snapshot - All XY graph & open scope blocks in reported systems
if (ax(dynamicX(:,2))>testMin & max(dynamicX(:,2)))
Paragraph - The conditioned signa...
Eval - expOkValues=[expOkValues;... ...
else
Paragraph - Could not set %<expBl...
Section 1 - Post-Test Analysis
Paragraph - <Text from children>
if (~isempty(expOkValues))
then
Text - %<size(expOkValues, 1...
Eval - expOkValues=[{'Mu Value','Signal...
Array-Based Table - expOkValues
else
Text - None of the selected ...
else
Chapter - Load Model Failed
Paragraph - Error: Model %<expMod...

Error Handling for MATLAB Code

You can add MATLAB code to a report, by using the `Evaluate MATLAB Expression` component (also called the Eval component). See "Add MATLAB Code" on page 3-7 for details.

The Evaluate MATLAB Expression component dialog box includes an **Evaluate this expression if there is an error** check box. The dialog box includes default error handling code that you can use, or you can create your own error handling code.

If you do not change the default error handling code, then when you generate the report, and there is an error in the MATLAB code that you added:

- If you clear **Evaluate this expression if there is an error** check box, then the complete report is generated, without displaying an error message at the MATLAB command line.
- If you select **Evaluate this expression if there is an error** check box, then the complete report is generated and an error message appears at the MATLAB command line.

To stop report generation when an error occurs in the MATLAB code that you added, change the second and third lines of the following default error handling code, as described below:

```
warningMessageLevel = 2;
displayWarningMessage = true;
failGenerationWithException = false;
failGenerationWithoutException = false;
```

To stop report generation and display an exception, change the default code to:

```
displayWarningMessage = false;
failGenerationWithException = true;
```

To stop report generation without displaying an exception, change the default code to:

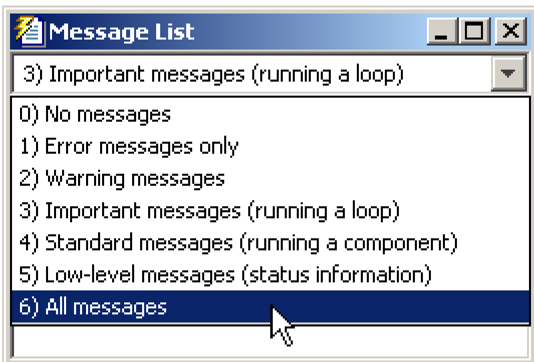
```
displayWarningMessage = false;
failGenerationWithoutException = true;
```

If you want to completely replace the default error handling code, use the `evalException.message` variable in your code to return information for the exception.

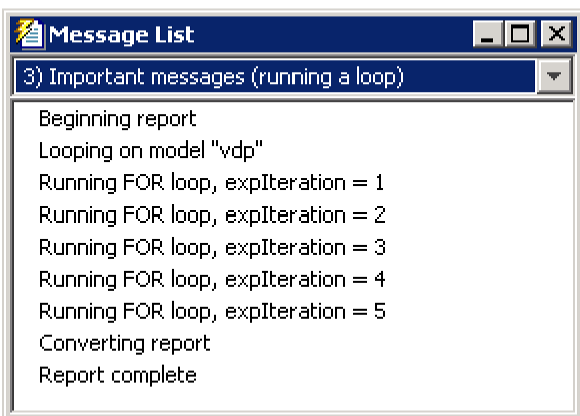
Generate the Report

Now the report includes all required components. To generate the report, click the Report icon on the toolbar. The following occurs:

1 A Message List window appears, displaying informational and error messages as the report is processed. Specify the level of detail you would like the Message List window to display while the report is being generated. Options range from `0` (least detail) to `6`(most detail). Click the selection list located under the title bar of the Message List window to choose an option, as shown in the following figure.

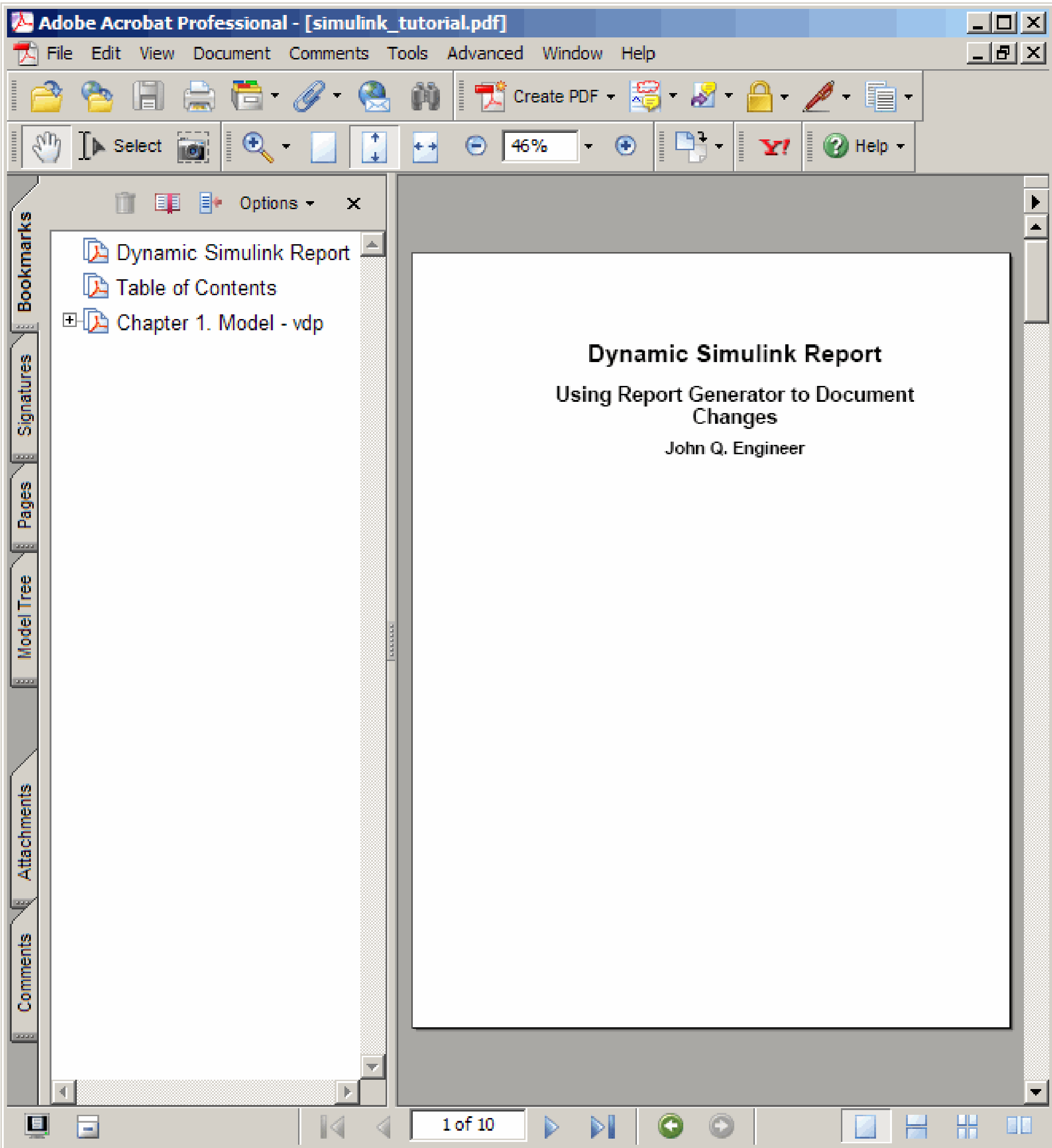


`Message level 3 (Important messages)` is used for the remainder of this example.



2 The vdp model appears. You can see each time it is simulated.
3 The scope window appears. The scope graph changes each time the parameter value changes.
4 Each component of the report is highlighted as it executes, in the Outline pane on the left in the Report Explorer window.

When the report is complete, Adobe Acrobat Reader opens your report in PDF format.


Adobe Acrobat Professional - [simulink_tutorial.pdf]
File Edit View Document Comments Tools Advanced Window Help
Create PDF
Select
46%
Help
Options
Bookmarks
Signatures
Pages
Model Tree
Attachments
Comments
Dynamic Simulink Report
Table of Contents
Chapter 1. Model - vdp
Dynamic Simulink Report
Using Report Generator to Document Changes
John Q. Engineer
1 of 10

Generate a Report

Generate a Report

Run a Report

You can generate a Simulink Report Generator report using one of these methods:

- In the Report Explorer Outline pane, select a report and do one of the following actions:
  - In the Report Explorer toolbar, click the Report button .
  - Press **CTL+R**.
  - Select **File** > **Report**.
- From the MATLAB command line, use the `report` command. For example, to print the `system1_description` report in PDF format, use:

```
report system1_description -fpdf
```

Report Output Options

Before you generate a report, you can set options to control aspects of report generation processing such as:

- Output file format (PDF, HTML, or Microsoft Word)
- Stylesheet for the selected output file format, to control the layout of the report (for example, whether to display a title page, font, and section numbering)
- Output file location
- Whether to view the generated report automatically

For details, see:

- "Report Output Format" on page 4-5
- "Location of Report Output File" on page 4-11

- "Create a Report Log File" on page 4-21
- "Report Description" on page 4-12
- "Change Report Locale" on page 4-17

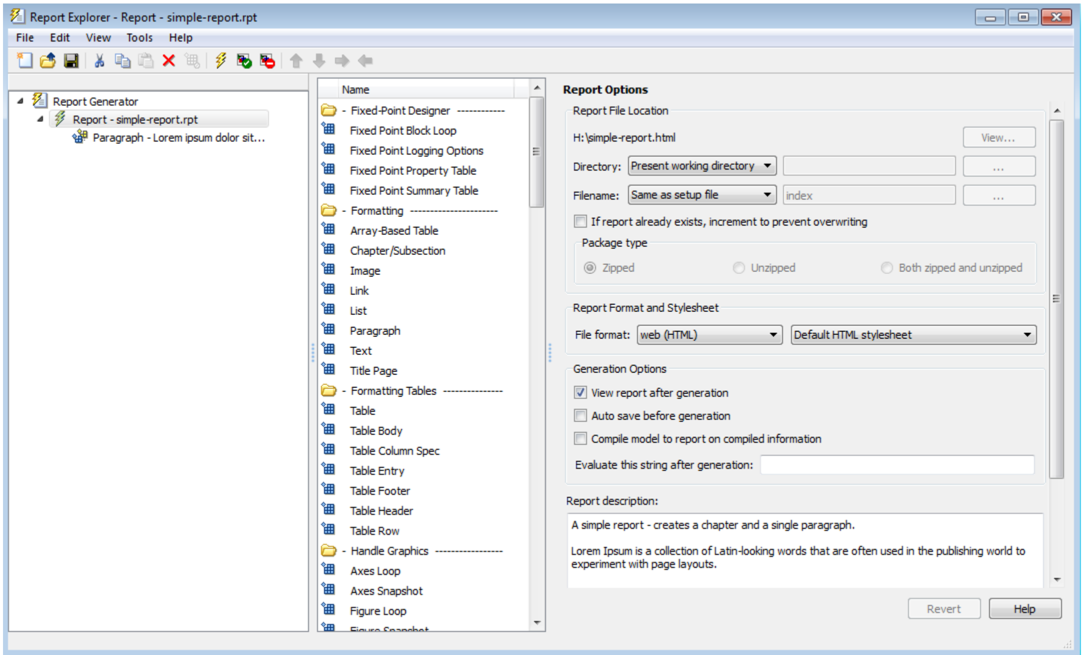
Select Report Generation Options

In this section...

Report Options Dialog Box

To specify report generation options for a specific report, in the Report Explorer, use the Report Options dialog box.

The Report Options dialog box in the Properties (right hand) pane of the Report Explorer.



To set defaults for report generation options that you can override with the Report Options dialog box or with individual components, use the Report Generator Preferences pane. For details, see “Report Generation Preferences” on page 4-13.

Report Output Format

In the Report Explorer, in the **File format** text box, choose the report output format.

Using a template when generating a report provides several benefits, compared to generating a report without using a template.

- Report generation is faster.
- Report generation does not use Java® memory. Generating reports without using a template can cause Java to run out of memory.
- You can customize report formatting using standard techniques for specifying Word and HTML styles.

File Format Using Report Templates

To use a template for report generation, select one of these options:

- `PDF (from template)`
- `HTML (from template)`
- `Word (from template)`

If you use a template, then from the list of templates, you can specify a template other than the default template. Each output format that does not use a template has a default stylesheet associated with it. Specify the stylesheet in the text box next to the **File format** text box.

For more information about using templates for report generation, see "Generate a Report Using a Template".

File Format Using Report Explorer Stylesheets

If you do not use a template, then you can specify in the **Stylesheet** field, you can specify a stylesheet.

Viewer	Format	Description	Stylesheets
Adobe Acrobat Reader	Adobe Acrobat (PDF)	Produce a PDF that you can view using Adobe Acrobat Reader software. See "PDF: Image Formats" on page 4-7.	PDF (see "PDF Stylesheets")
Word processor	`Word Document (RTF)` or `Rich Text Format`	Produce output that is compatible with most word-processing packages, including Microsoft Word software See "RTF: Display of Hidden Content" on page 4-7.	Print (see "RTF (DSSSL Print) and Word Stylesheets")

Viewer	Format	Description	Stylesheets
DocBook	`DocBook (no transform)`	Produce a report in DocBook format	N/A

Tip To create and use customized styles, see "Create a New Stylesheet".

PDF: Image Formats

PDF reports only support bitmap (`.bmp`), `.jpeg` (`.jpg`), and Scalable Vector Graphics (`.svg`).

The SVG format is only supported for Simulink models and Stateflow charts. For example, MATLAB figures do not display in SVG when you select the SVG format for PDF reports.

RTF: Display of Hidden Content

RTF reports use placeholders (field codes) for dynamically generated content, such as page numbers or images.

On Windows platforms, to display that content, press **Ctrl-A**, and then press **F9**.

On Linux and Mac platforms, use the field code update interface for the program that you are using to view the RTF document.

Change the Default Output Format

In the Report Generator Preferences pane, use the **Format ID** preference to specify the default output format for reports.

Stylesheets

For each output format, you can choose from several stylesheets for each report output format. For details, see:

- "PDF Stylesheets" on page 4-8
- "Web Stylesheets" on page 4-8
- "RTF (DSSSL Print) and Word Stylesheets" on page 4-9

Note: Some Web and Print stylesheets include an automatically generated list of titles, which includes table titles and figures with titles.

PDF Stylesheets

PDF Stylesheet	Description
`Default print stylesheet`	Displays title page, table of contents, list of titles
`Standard Print`	Displays title page, table of contents, list of titles
`Simple Print`	Suppresses title page, table of contents, list of titles
`Compact Simple Print`	Minimizes page count, suppresses title, table of contents, list of titles
`Large Type Print`	Uses 12-point font (slightly larger than Standard Print)
`Very Large Type Print`	Uses 24-point font and landscape paper orientation
`Compact Print`	Minimizes white space to reduce page count
`Unnumbered Chapters & Sections`	Uses unnumbered chapters and sections
`Numbered Chapters & Sections`	Numbers chapters and sections
`Paginated Sections`	Prints sections with page breaks
`Custom Header`	Lets you specify custom headers and footers
`Custom Titlepage`	Lets you specify custom title page content and presentation
`Verbose Print`	Lets you specify advanced print options

Web Stylesheets

Web Stylesheet	Description
`Default HTML stylesheet`	HTML on a single page
`Simulink book HTML stylesheet`	HTML on multiple pages; suppresses chapter headings and table of contents

Web Stylesheet	Description
Truth Table HTML stylesheet	HTML on multiple pages; suppresses chapter headings and table of contents
Multi-page Web	HTML, with each chapter on a separate page
Single-page Web	HTML on a single page
Single-page Unnumbered Chapters & Sections	HTML on a single page; chapters and sections are not numbered
Single-page Numbered Chapters & Sections	HTML on a single page; chapters and sections are numbered
Single-page Simple	HTML on a single page; suppresses title page and table of contents
Multi-page Simple	HTML on multiple pages; suppresses title page and table of contents
Multi-page Unnumbered Chapters & Sections	HTML on multiple pages; chapters and sections are not numbered
Multi-page Numbered Chapters & Sections	HTML on multiple pages; chapters and sections are numbered

RTF (DSSSL Print) and Word Stylesheets

RTF or Word Stylesheet	Description
Standard Print	Displays title page, table of contents, list of titles
Simple Print	Suppresses title page, table of contents, list of titles
Compact Simple Print	Minimizes page count, suppresses title, table of contents, list of titles
Large Type Print	Uses 12-point font (slightly larger than Standard Print)
Very Large Type Print	Uses 24-point font and landscape paper orientation
Compact Print	Minimizes white space to reduce page count
Unnumbered Chapters & Sections	Uses unnumbered chapters and sections
Numbered Chapters & Sections	Numbers chapters and sections

Report Generation Processing

The Report Options dialog box includes several options for controlling report processing.

Option	Purpose
View report after generation	View the report automatically. When report generation finishes, the viewer associated with the report output format displays the report. **Note:** On Linux and Macintosh platforms, the report output displays in Apache OpenOffice, which must be installed in `/Applications/OpenOffice.app`. To view the report manually, browse to the location specified in the **Report File Location** section in the Properties pane on the right, and open the file.
Auto save before generation	Automatically save the report setup file before you generate a report.
Compile model to report on compiled information	Ensure that a report reflects compiled values. By default, the Simulink Report Generator reports uncompiled values of Simulink parameters. The uncompiled values of some parameters, such as signal data types, can differ from the compiled values used during simulation. This option causes the report generator to compile a model before reporting on model parameters. After generating the report, the report generator returns the model to its uncompiled state. **Note:** When you select this option, whenever report generation requires simulating the model (for example, the report includes a “Model Simulation” component), the report generator uncompiles the model and then recompiles the model, if necessary, to report on model

Option	Purpose
	contents. If a report requires multiple compilations, the processing can be quite time-consuming. To minimize compilations, consider using separate reports to report on the contents of a model and on the results of simulating that model.
Evaluate this string after generation	Specify MATLAB code for processing to occur after the report is generated. For example, you could specify to close a model.

Location of Report Output File

Choose a folder to store the report file. You must have write privileges for that folder.

Folder

In the Report Explorer, in the Report Options dialog box, use the **Directory** field to specify the name of the folder in which to store the generated report file. Specify a folder to which you have write privileges.

The following table summarizes the report file location options.

Folder	Option
The same folder as the report setup file	`Same as setup file`
The current working folder	`Present working directory`
Temporary folder	`Temporary directory`
Another folder	`Custom`. Use the **Browse** button (...) to select from a list of directories.

You can use `%<VariableName>` notation to specify a folder in the **Custom** text box. For more information, see “%<VariableName> Notation” on the `Text` component reference page.

Report File Name

In the Report Explorer, in the Report Options dialog box, use the **Filename** field to specify a file name for the report file. Select one of the following options.

File Name	Option
The same file name as the report setup file	`Same as setup file` (default)
A file name different from the report setup file name	`Custom`. Enter the name of the report.

You can use %<VariableName> notation to specify a file name in the **Custom** text box. For more information, see “%<VariableName> Notation” on the `Text` component reference page.

Increment to Prevent Overwriting

To maintain the previous version of the setup file when you save updates to the setup file, select **If report already exists, increment to prevent overwriting**.

Image Output File Location

Images are placed in a folder with the same name as the report file. For example, `testreport.html` images are placed in a folder named `testreport_files`.

Report Description

To record notes and comments about your report setup, use the **Report Description** field. This text that you enter appears in the Properties pane when you select a report setup file in the Outline pane.

Report Generation Preferences

In this section...

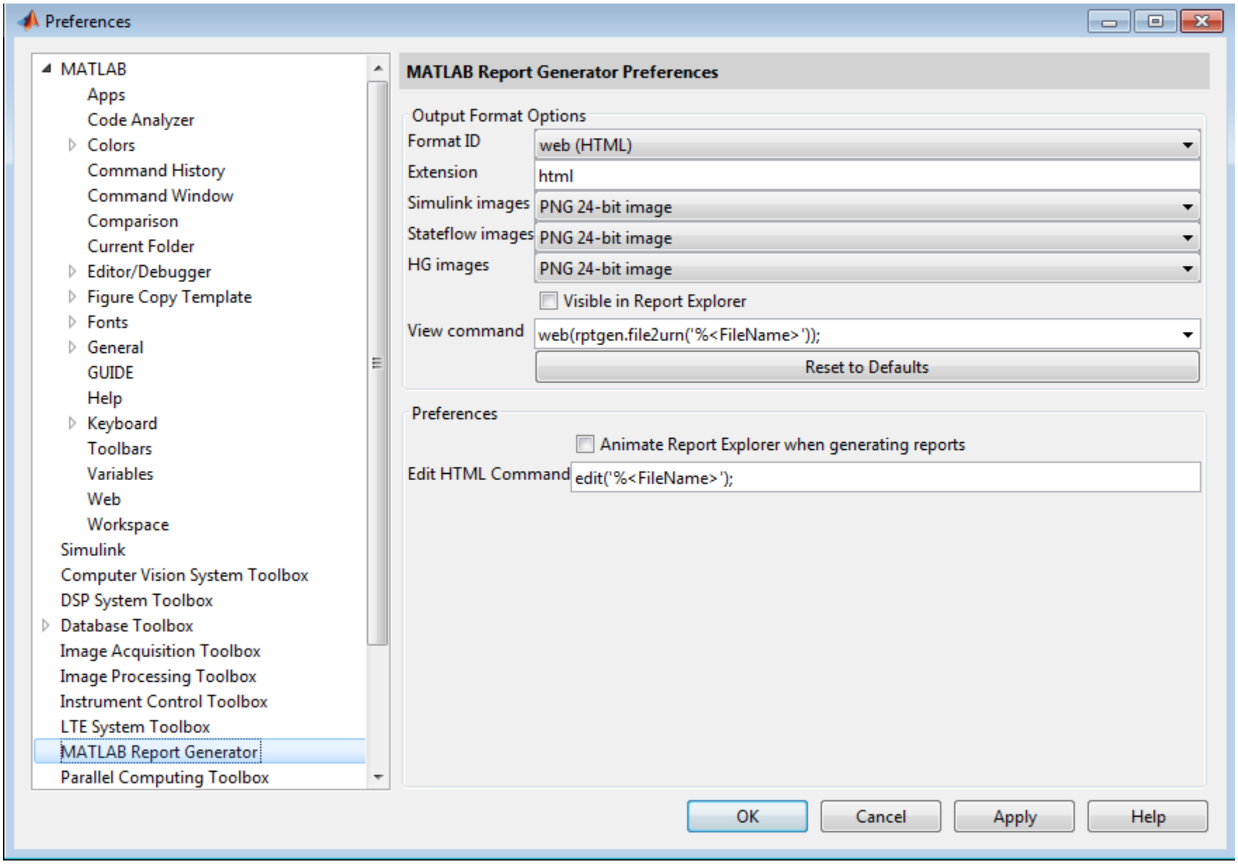
Report Generator Preferences Pane

To set defaults for report generation options, use the Report Generator Preferences pane. You can override these preferences with the Report Options dialog box or with individual components.

To specify report generation options for a specific report, in the Report Explorer, use the Report Options dialog box. For details, see "Select Report Generation Options" on page 4-4.

To open the Report Generator Preferences pane, use *one* of these approaches:

- In the Report Explorer, select **File** > **Preferences**.
- From the MATLAB Toolstrip, in the **Home** tab, in the **Environment** section, select **Preferences** > **Report Generator**.



File Format and Extension

To specify the default file format for reports, use the **Format ID** preference. The default preference is `web (HTML)`. You can select from a range of file formats, such as PDF, Microsoft Word, or LaTex.

Note: For reports that use the Word Document format, you must have Microsoft Word installed on the machine that you use to generate the report.

The **Extension** preference reflects the standard file extension for the file format specified with the **Format ID** preference. You can change the extension.

Image Formats

To set the default image formats associated with the output format for a report, use the following preferences.

Preference	Purpose
`Simulink Images`	Specify the format for Simulink images to include in the report.
`Stateflow Images`	Specify the format for Stateflow charts to include in the report.
`HG Images`	Specify the format for Handle Graphics images to include in the report.

Note: The default preferences for image formats should work in most viewing environments. However, some image formats do not display in some viewing environments.

Several components, such as the Figure Snapshot component, include an option for specifying the image format. The component setting overrides the image format preference.

Report Viewing

To control how you view a generated report, you can set the following preferences.

Preference	Purpose
`View command`	Specify the MATLAB command you want to use to view the report. Each file format has an associated default view command preference. You can modify the view command (for example, to support the use of a system browser).

Preference	Purpose
`Visible in Report Explorer`	Deselect this check box to make the current output format unavailable in the Report Explorer. For example, if your specified report format is `Word document` and you deselect this check box, then the Microsoft Word document format is no longer available for reports created using the Report Explorer.
`Animate Report Explorer when generating reports`	Select this check box if you want components in the Outline pane to be animated as the report generates. This box is selected by default. To speed up the report generation processing, consider clearing this preference.

Reset to Defaults

To reset all of the preferences in the Output Format Options section of the Report Generator Preferences pane, click **Reset to Defaults**.

The **Reset to Defaults** button does not change the **Animate Report Explorer when generating reports** preference.

Change Report Locale

Versions 2.0 and later of the MATLAB Report Generator and Simulink Report Generator software use the locale (system language settings) through the Oracle® Java interface; therefore, they should use the language specified on your system.

Alternatively, you can change the language directly in Java from the MATLAB command line. The following example sets the language to Italian:

```
java.util.Locale.setDefault(java.util.Locale.ITALY)
```

Alternatively, you can set the preferred language directly in your `.rpt` file:

1 Right-click the Report component and select **Send to Workspace**.

   This displays the properties of the report, which are stored in the variable *ans*. Access the report's `Language` property from the command line through this variable. By default, `Language` is `auto`, which indicates that the system's default language is in use.

2 Override the default value of `Language` by setting this property to your desired language; for example, `en` for English or `it` for Italian.

Convert XML Documents to Different File Formats

In this section...

Why Convert XML Documents?

You can generate a report in a different output file format without regenerating it by using either the Report Explorer File Converter or the `rptconvert` command. These utilities convert DocBook XML source files created by the report-generation process into formatted documents such as `HTML`, `RTF`, or `PDF`.

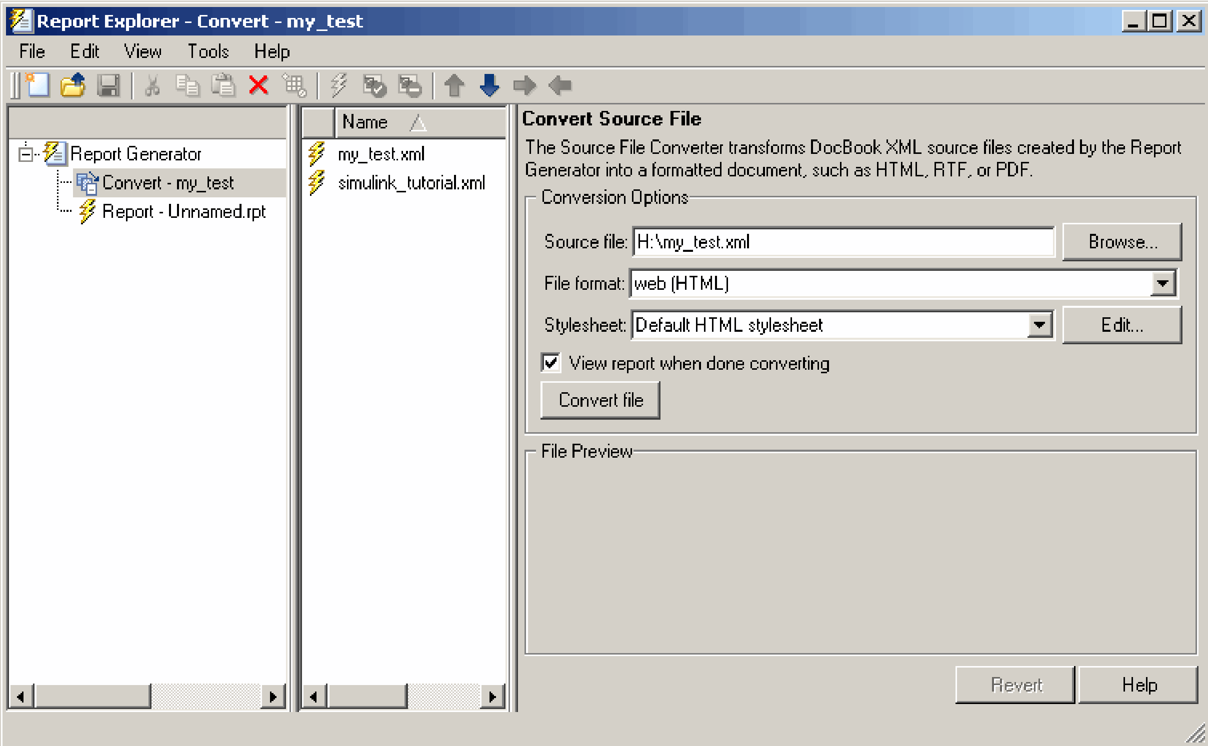
Note: The report-generation process can only convert XML source files created by the latest version of the software.

Convert XML Documents Using the Report Explorer

To open the **Convert** Properties pane:

1. In the Report Explorer, select **Tools** > **Convert source file**.

   The Convert Source File Properties pane appears. All XML files in your current folder appear in the Options pane in the middle.



2 Select your XML source file using one of the following options:

- Click **Browse** in the Properties pane on the right to browse to the location of your XML source.
- Double-click a file name in the Options pane in the middle to automatically enter it into the **Source file** field in the Properties pane.

3 Select your output format and stylesheet:

a In the **File format** text box, select an output format.

b In the **Stylesheet** text box, select a stylesheet. The stylesheet choice depends on the specified output format. You can use a predefined or customized stylesheet.

For more information about available formats and predefined stylesheets, see “Report Output Format”.

For more information about customizing stylesheets, see "Create a New Stylesheet".

4 Use the **View Report when done converting** check box to indicate whether you want to view the report after it has conversion.
5 To begin the conversion, click **Convert file**.

Convert XML Documents Using the Command Line

To convert files using the command line, use the "rptconvert"function.

Edit XML Source Files

Before you send a source file to the converter, edit it as text in the Report Explorer:

1 In the Outline pane on the left, open the File Converter.
2 Right-click **MATLAB Report Generator** and select **Convert source file**.
3 In the Options pane in the middle, select the source file to edit.
4 In the Properties pane on the right, click **Edit as text**.
5 Use the MATLAB Editor to edit and save the text.

Create a Report Log File

A log file describes the report setup file report-generation settings and components. A log file can be used for many purposes, including:

- As a debugger
- As a reference to a report setup file
- To share information about a report setup file through email

A log file includes the following information:

- Report setup file outline
- Components and their attributes
- Generation status messages currently displayed in the **Generation Status** tab

To generate a log file, click **File** > **Log File**. An HTML version of the log file with the name <report_template_file_name_log>.html is saved in the same folder as the report setup file.

Generate MATLAB Code from Report Setup File

You can generate MATLAB code versions of report setup files in the form of a MATLAB file (`*.m`). A MATLAB file of a report setup file is useful for various purposes, including generating reports and modifying report setup files programmatically.

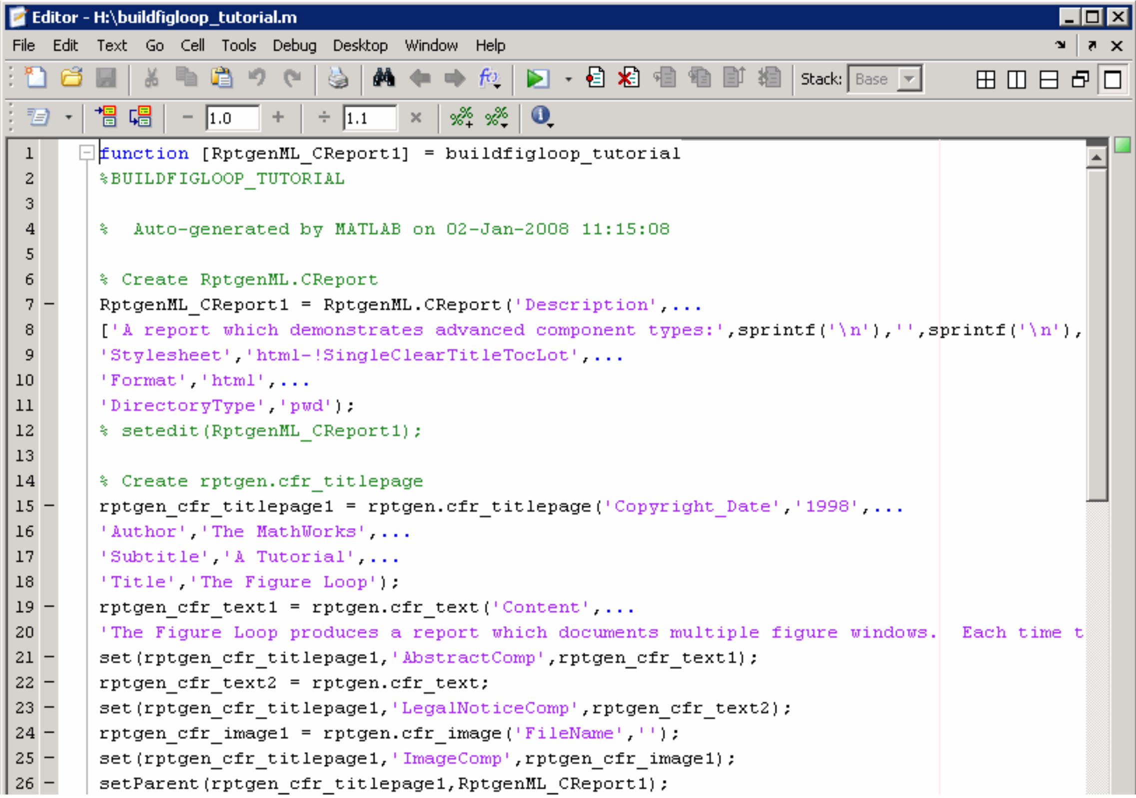
To generate a MATLAB file, load a report setup file into the Report Explorer and click **File** > **Generate MATLAB File**. After the MATLAB file generates, it opens in the MATLAB Editor. The filename for the generated file is the file name of the report setup file , preceded by "build."

Generate Reports from MATLAB Files

This example generates a MATLAB file from the `figloop_tutorial.rpt` report setup file, which is part of the MATLAB Report Generator software. The example then uses the `report` function to generate a report from the MATLAB file. For more information about this function, see the `report` reference page.

1. Start the Report Explorer by entering `report` in the MATLAB Command Window.
2. In the Options pane in the middle, double-click `figloop_tutorial.rpt` to open its report setup file.
3. In the Outline pane on the left, click `Report - figloop_tutorial.rpt` to select it.
4. In the Report Explorer menu bar, click **File** > **Generate MATLAB File**.

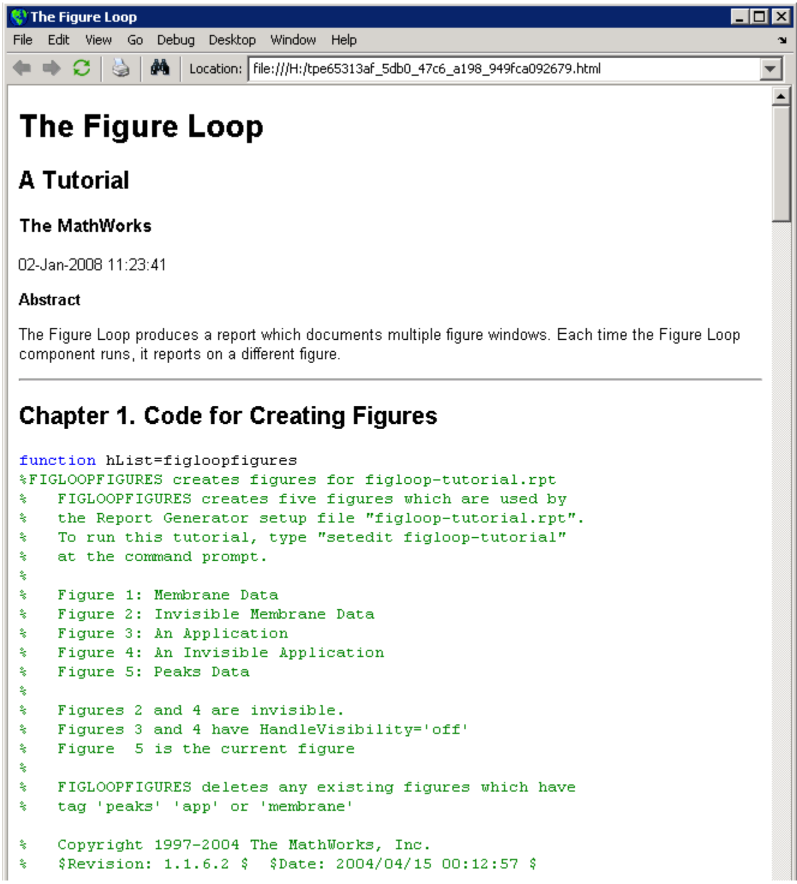
   The MATLAB Report Generator software generates MATLAB code for the `figloop_tutorial.rpt` report setup file. It saves this code in the `buildfigloop_tutorial.m` file in the folder you specify. Part of this file appears in the following figure.



5 To generate the `figloop_tutorial` report from this MATLAB file, run the following command in the MATLAB Command Window:

```
report(buildfigloop_tutorial);
```

The MATLAB Report Generator software runs and displays the report.



The Figure Loop

A Tutorial

The MathWorks

02-Jan-2008 11:23:41

Abstract

The Figure Loop produces a report which documents multiple figure windows. Each time the Figure Loop component runs, it reports on a different figure.

Chapter 1. Code for Creating Figures

```
function hList=figloopfigures
%FIGLOOPFIGURES creates figures for figloop-tutorial.rpt
%   FIGLOOPFIGURES creates five figures which are used by
%   the Report Generator setup file "figloop-tutorial.rpt".
%   To run this tutorial, type "setedit figloop-tutorial"
%   at the command prompt.
%
%   Figure 1: Membrane Data
%   Figure 2: Invisible Membrane Data
%   Figure 3: An Application
%   Figure 4: An Invisible Application
%   Figure 5: Peaks Data
%
%   Figures 2 and 4 are invisible.
%   Figures 3 and 4 have HandleVisibility='off'
%   Figure  5 is the current figure
%
%   FIGLOOPFIGURES deletes any existing figures which have
%   tag 'peaks' 'app' or 'membrane'

%   Copyright 1997-2004 The MathWorks, Inc.
%   $Revision: 1.1.6.2 $  $Date: 2004/04/15 00:12:57 $
```

Troubleshooting Report Generation Issues

In this section...
“Memory Usage” on page 4-25
“HTML Report Display on UNIX Systems” on page 4-25

Memory Usage

By default, the MATLAB software sets a limit of 100 MB on the amount of memory the Oracle Java Virtual Machine (JVM™) software can allocate. The memory that the report generation process uses to build a document must fit within this limit. If you are having trouble processing large reports, it might be helpful to increase the amount of memory that MATLAB Report Generator and Simulink Report Generator software can allocate. See the following sections for more information.

Run the MATLAB Software Without a Desktop

One way to increase the amount of JVM memory available to the MATLAB Report Generator and Simulink Report Generator software is to run the MATLAB software with `-nodesktop` mode enabled.

Increase the MATLAB JVM Memory Allocation Limit

To increase the amount of JVM memory available by increasing the MATLAB JVM memory allocation limit, from the MATLAB Toolstrip, in the **Home** tab, in the **Environment** section, select **Preferences**. Use the **General** > **Java Heap Memory** dialog box.

HTML Report Display on UNIX Systems

HTML reports may not automatically display on some UNIX® platforms. To work around this issue, configure the MATLAB Report Generator software to launch an external browser:

1. In the Report Explorer, click **File** > **Preferences**.
2. Enter the following text in the **View command** field:

   ```
   web(rptgen.file2urn('%file name'), '-browser')
   ```
   Where *file name* is the name of your report setup file.

5

Export Simulink Models to Web Views

Web Views

What Is a Web View?

A Web view is an interactive rendition of a model that you can view in a Web browser. You can use Web views to navigate hierarchically to specific subsystems and see properties of blocks and signals. Web views provide a simple way to interactively explore a model. For example, you can view block parameter values without opening a block parameter dialog box.

Use Web views to share models with people who do not have Simulink installed.

You can save Web views of a model over time, creating snapshots of the model as it changes during the development process.

System Requirements

Although you use Simulink Report Generator software to create Web views. you can display a Web view in a browser, even if you do not have Simulink Report Generator installed.

By default, when you export a Web view, that Web view automatically displays in your default Web browser. Web views require a Web browser that supports SVG natively.

Note: The MATLAB Web browser does not support Web views.

Web View Files

By default, exporting a Web view creates a zip file that includes the Web view HTML file, as well as files that support Web view display. Supporting files include files include `.svg` and `.png` files. Zip file packaging compresses the files and consolidates the Web view and supporting files into one zip file.

You can choose to export the Web view files as the Web view HTML file and the supporting files, in a folder, without being zipped. You can open the Web view HTML file directly, without having to open a non-zipped file. You can also choose to export the Web view files as both a zip file and as non-zipped files.

The default name of the zip file or folder that contains the non-zipped Web view files is the name of the model that contains the systems to export. You can specify a different file or folder name.

The default location for storing Web view files is the MATLAB current folder. You can choose a different folder.

If you send Web view files to someone else, consider whether you need to explain how to access the Web view file.

Related Examples

- "Export Models to Web View Files" on page 5-4
- "Display and Navigate a Web View" on page 5-6
- "Create and Use a Web View" on page 5-10

More About

- "Optional Web Views" on page 5-17

Export Models to Web View Files

In this section...

Open the Web View Dialog Box

To export a Web view, use the Web View dialog box. The way you access the dialog box differs, depending on whether you are using the Simulink Editor or the Report Explorer. You can also open the dialog box from the command line.

Interface	What You Do
Simulink Editor	Select **File** > **Export Model to** > **Web**.
Report Explorer	Select **Tools** > **Export Simulink to Web**.
Command line	Use the `slwebview` function without arguments.

Export a Model to a Web View

1. Open the model to export.
2. In the Simulink Editor, select **File** > **Export Model to** > **Web**.
3. In **Systems to Export**, select the levels of the model to export, in relationship to the system currently displayed or chart currently selected in the Simulink Editor.
4. For the systems in the levels that you are exporting, in **Include Options**, select any kinds of systems you want the Web viewer user to be able to navigate below the Subsystem or Model block, to the underlying blocks or models.

   If you select more than one kind of system, the criteria for exporting information for interacting with the contents of the systems are applied downward through the model hierarchy. For example, if you left **Referenced Models** unchecked when you exported the model, regardless of how you set the **Library Links** option, in the Web view you cannot interact with a library link block that is inside of a referenced model.
5. In the **Systems to Exclude** list of the systems you have selected to export, select any systems that you do not want to export.

6 To avoid overwriting existing exported Web view files, select **If file exists, increment name to prevent overwriting**.

7 In **Package Type**, specify whether you want to package the Web view as a zipped file (the default packaging). In **Package name**, you can specify a name for the zip file or for the folder for the Web view files.

8 Click **Export**.

Note: If you use the Web View export option in the Report Explorer Properties pane, then click **Export model**. If you change the visible system or chart while the Report Explorer **Web View** pane is visible, the pane does not automatically change to show information about the newly visible system. To update the **Web View** pane, click **Refresh**.

---

9 If you have Simulink Verification and Validation installed, you can use the **Optional Views** tab to select **Model Coverage** or **Requirements**, or both, to export the associated kind of information.

See Also

Functions

`slwebview`

Related Examples

- “Display and Navigate a Web View” on page 5-6
- “Create and Use a Web View” on page 5-10

More About

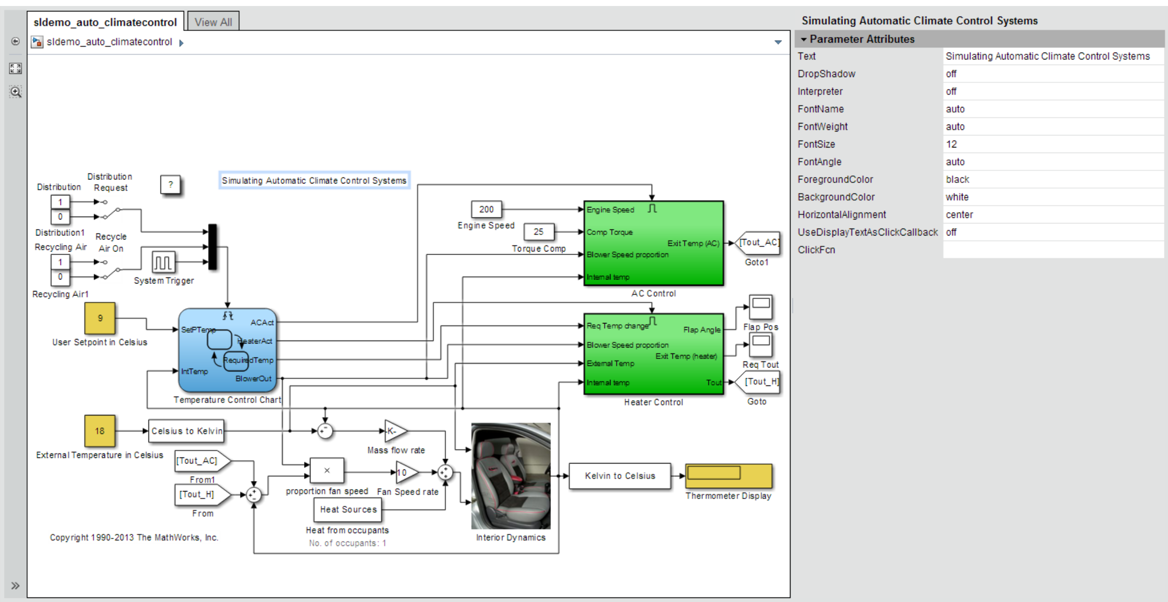
- “Web Views” on page 5-2
- “Web View Files” on page 5-2
- “Optional Web Views” on page 5-17

Display and Navigate a Web View

In this section...

Display a Web View When You Export It

When you export a Web view using the Web View dialog box or from the Report Explorer **Web View** pane, the Web view appears in your system Web browser.



Open a Web View File in a Web Browser

To open a Web view file to display in a Web browser, from the folder that contains the Web view files, select the HTML file.

Open the `webview.html` file to display the Web view. For details about file packaging and location, see “Web View Files” on page 5-2.

To use a Google® Chrome Browser, you need to do some setup.

Open a Web View in a Google Chrome Browser on a Windows Platform

1. If you do not already have a shortcut set up for Google Chrome, click the Windows **Start** button and search for `Chrome`.
2. Right-click and drag the Google Chrome icon to an open area on your desktop.
3. Right-click the icon and select **Create shortcut**.
4. Right-click the shortcut and select **Properties**.
5. In the **Target** edit box, append the following text: `--allow-file-access-from-files`. Be sure to use two hyphens at the beginning. Click **OK**.
6. Close all open Google Chrome browsers.
7. Use the shortcut to open a Google Chrome browser.
8. Open the Web view file.

Open a Web View in a Google Chrome Browser on a Macintosh Platform

1. Run **Terminal**. You can find it using **Spotlight**, in `Applications/Utilities`.
2. Enter the following text:

```
open/Applications Google\Chrome.app --allow-file-access-from-files
```

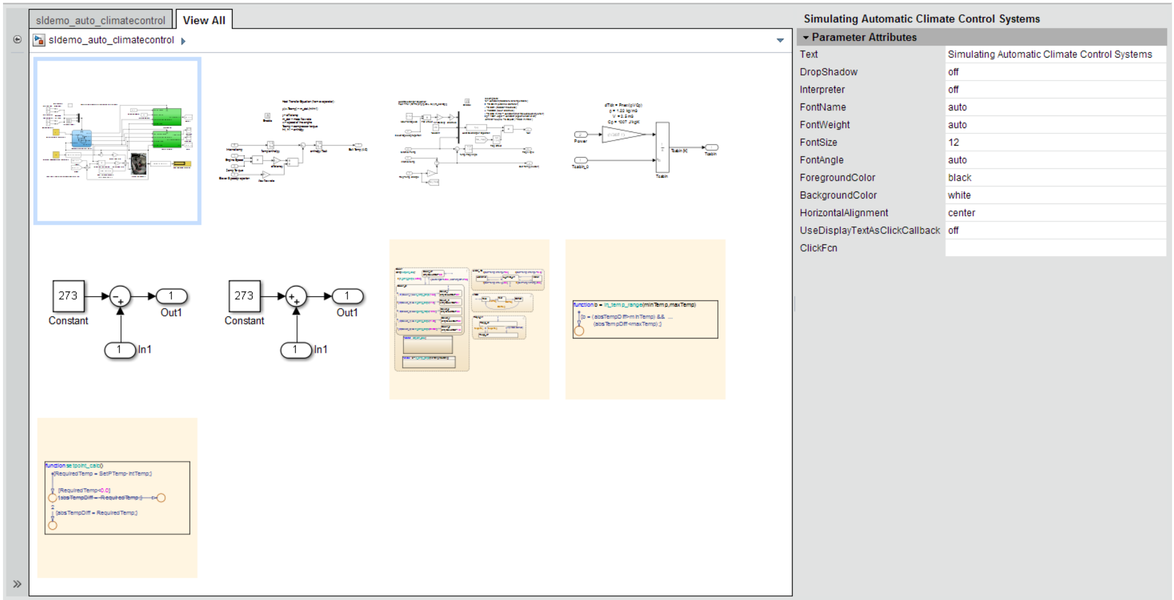
Open a Web View in a Google Chrome Browser on a Linux Platform

1. Run **terminal**.
2. Enter the following text:

```
./chromium-browser --allow-file-access-from-files
```

View Contents of a System

To see a thumbnail of the contents of all of systems in the Web view, click the **View All** tab.



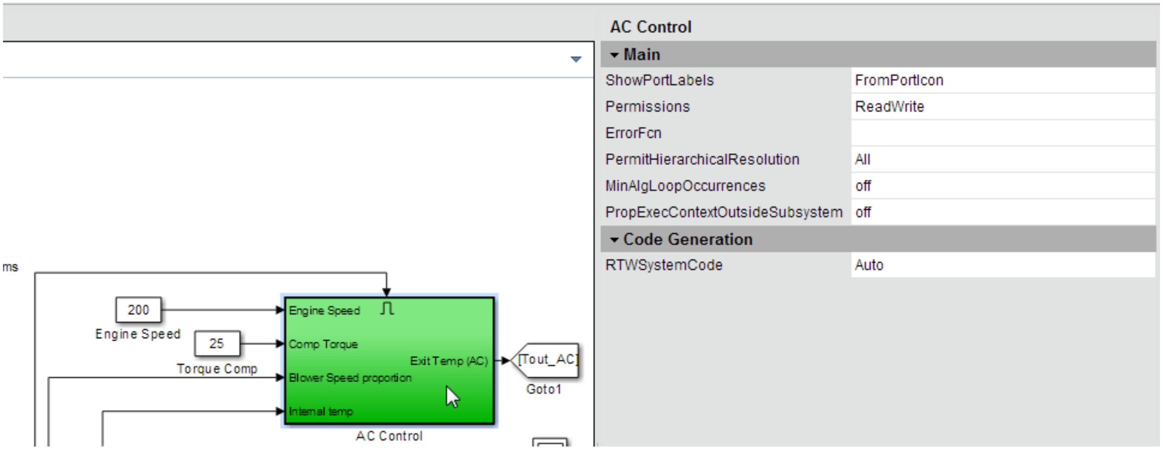
To view the contents of a specific system, use one of these approaches:

- In the model viewer, double-click the system.
- In the model browser, select a system. To expose this pane, click **Hide/Show Model Browser** ».
- Click the **View All** tab and click the thumbnail of a system.

To open a system in a separate tab, press **CTRL** and click the system.

View Block Parameters and Signal Properties

Click a block or signal in the model to see its parameters or properties in the **Object Inspector** pane.



Access Optional Web View Information

To view the model coverage or requirements optional Web view information in a Web view, you must have Simulink Verification and Validation installed. To access the information, click a highlighted block (for example, blocks with an orange border have requirements information). The information for that block appears in the **Informer** pane below the model.

Create and Use a Web View

In this section...

About This Tutorial

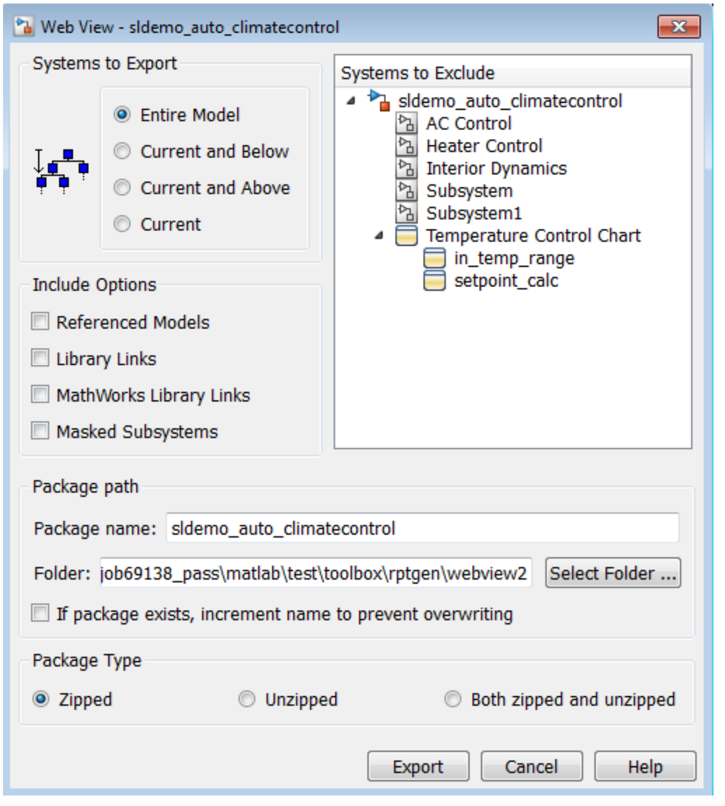
This tutorial takes you through the steps to export a Simulink model to a Web view.

You create a Web view from the Simulink model window using the `sldemo_auto_climatecontrol` model, which is provided with the Simulink software. This model simulates the working of an automatic climate control system in a car.

Export Specific Systems

When you create the Web view, you can specify export options.

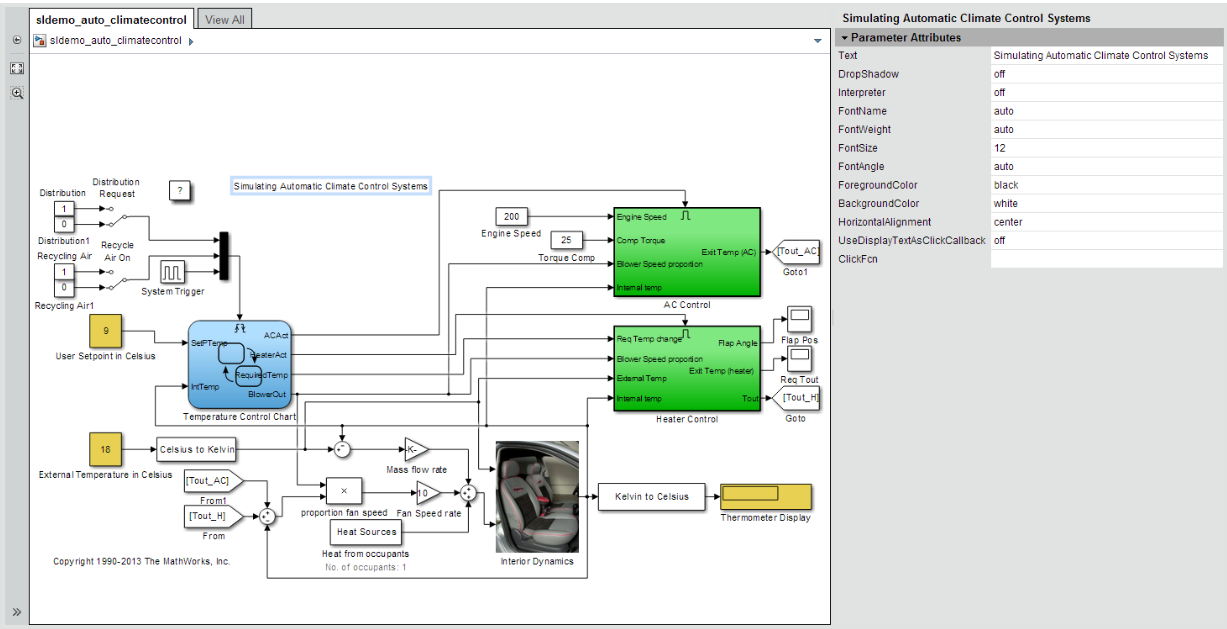
1. At the MATLAB command prompt, enter `sldemo_auto_climatecontrol` to open the Simulink model.
2. In the Simulink Editor, select **File** > **Export Model to** > **Web**.



3. In the **Include Options**, select **Masked Subsystems**. This enables users of the Web view to interact with masked blocks.
4. In **Systems to Exclude**, select `Temperature Control Chart`.
5. Use the **Folder** edit box to specify `climate_control_webview` as the name for the zip file for the exported Web view files.
6. Select the **If file exists, increment name to prevent overwriting** check box. Selecting this option prevents overwriting the Web view files if you export multiple Web views from the same model.
7. Click **Export**.

Exporting the selected systems to a Web view creates several support files, as well as an HTML file for displaying the systems. In this example, you change the defaults for the naming of the files.

The Web view files are exported, and the Web view appears in a Web browser.

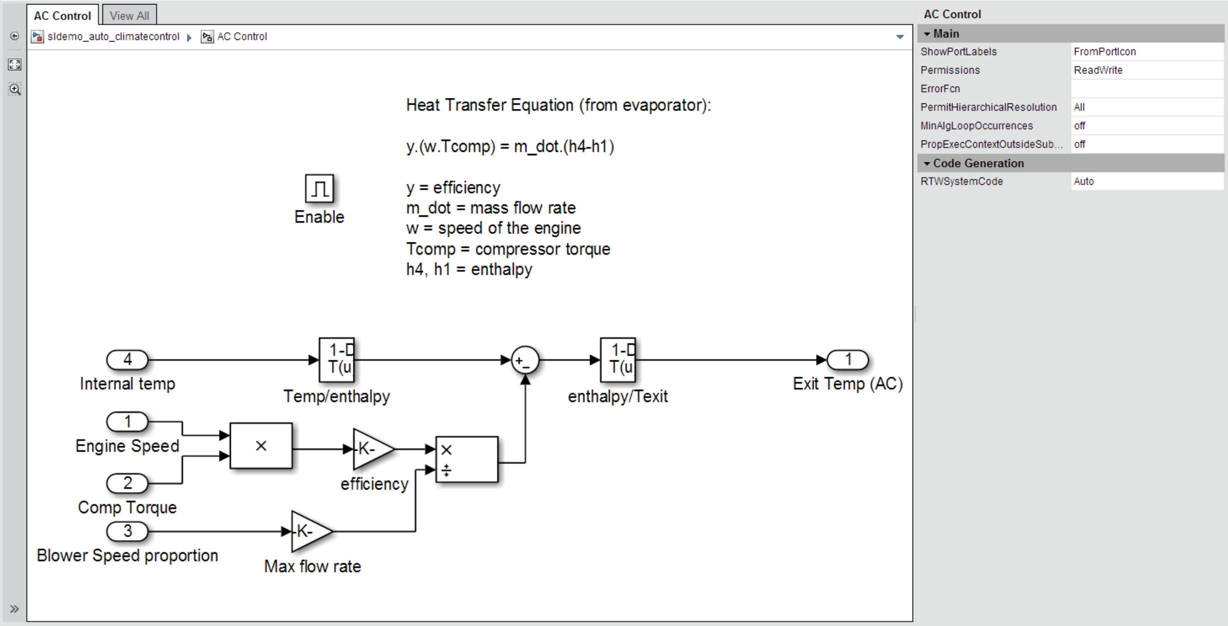


The Temperature Control Chart appears in the top level of the model, but you cannot open that chart in the Web view to see its contents.

Navigate the Web View

By default, if you export a whole model to a Web view, the **Model Viewer** pane shows the whole model. You can display specific systems in the Web view. For example:

1 In the model viewer, double-click the AC Control subsystem.

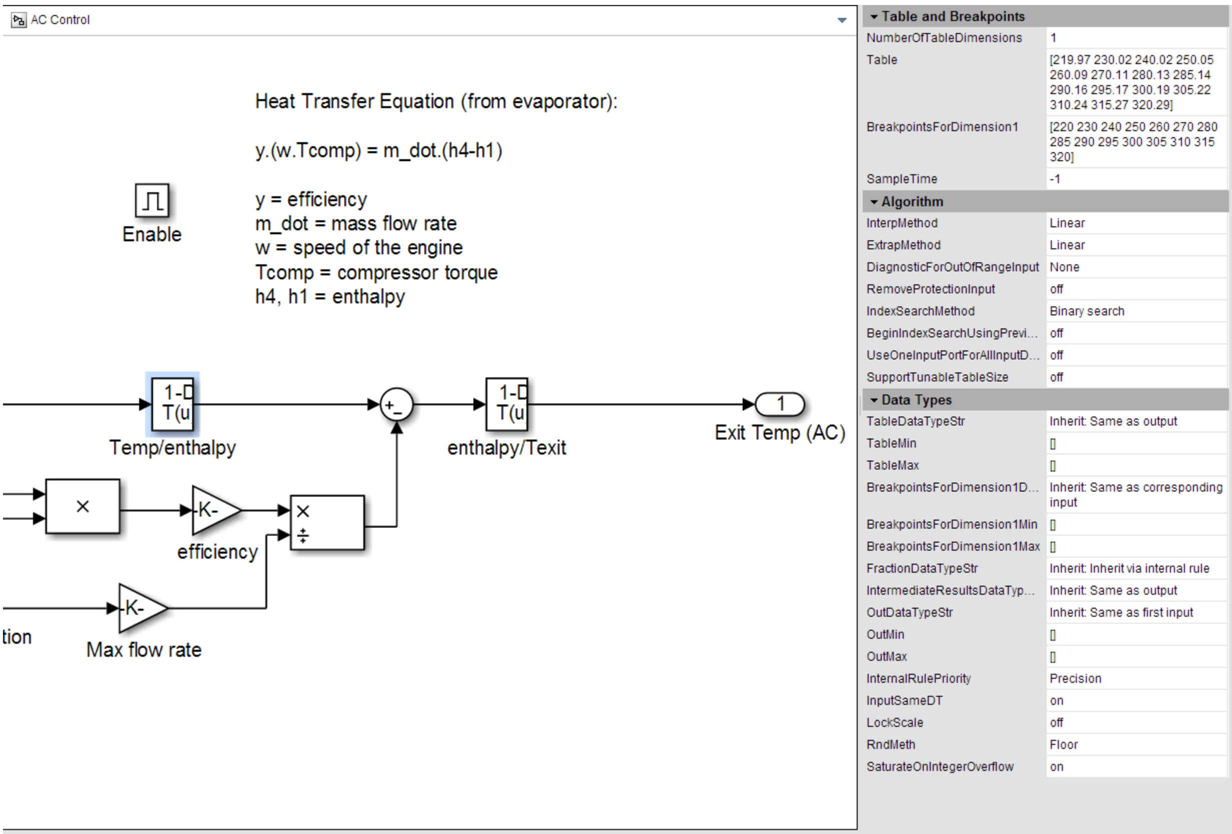


The AC Control subsystem appears in the model viewer. The tab label reflects the name of the currently displayed subsystem.

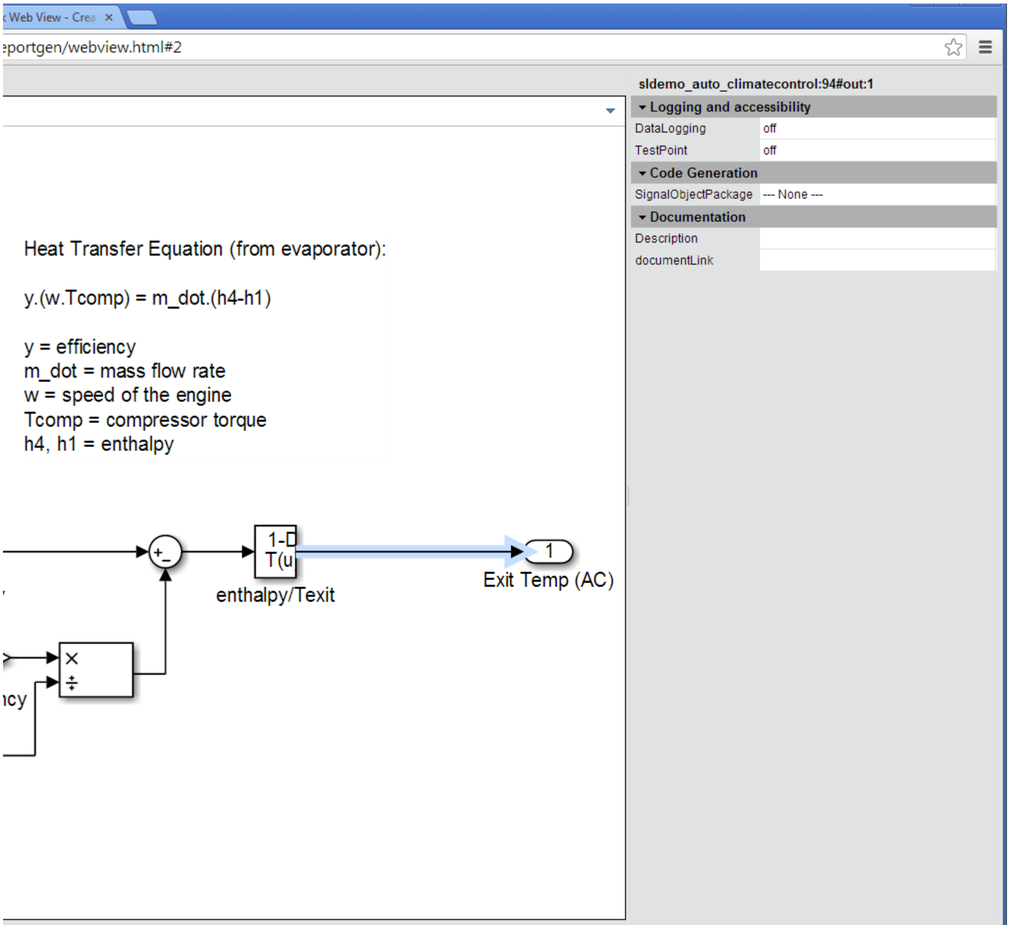
2 Open the model browser (hidden by default). Click **Hide/Show Model Browser** .
3 Open another system, in a separate tab. In the model browser, **CTRL+click** select the Heater Control system.
4 Drag the AC Control system to the top of model viewer. Place the cursor in the display area, hold down the mouse scroll wheel, and drag.
5 Zoom the display with the mouse scroll wheel.

Display Parameters and Properties of Blocks and Signals

1 In the model browser, select `sldemo_auto_climatecontrol`.
2 Double-click the AC Control subsystem.
3 Click the Temp/enthalpy block to view the block parameter values. The **Object Inspector** pane groups the block parameters by the block parameter dialog box tabs.



4 Click the input signal for the Exit Temp (AC) block to display the signal properties.



Try navigating to other parts of the Web view.

5 Close the Web view.

Open the Web View

In the MATLAB current folder (or wherever you saved the zip file when you performed the steps in “Export Specific Systems” on page 5-10), extract the `climate_control_webview` zip file contents and open the `webview.html` file.

Related Examples

- "Export Models to Web View Files" on page 5-4
- "Display and Navigate a Web View" on page 5-6

More About

- "Web Views" on page 5-2

Optional Web Views

Optional Web views provide information about a model in addition to the standard Web view information about blocks and signals in a model.

The Simulink Report Generator includes the following optional Web views that you can capture and view if you have Simulink Verification and Validation installed:

- Requirements
- Model coverage

These optional views display requirements or model coverage information associated with the current Web view.

To include model coverage information in a Web view, set up a coverage report for the model and simulate the model before selecting the model coverage optional view option.

Related Examples

- "Capture and View Optional Web View Information" on page 5-18
- "Export Models to Web View Files" on page 5-4

More About

- "Optional Web Views" on page 5-17

Capture and View Optional Web View Information

In this section...

Capture Optional Web View Information for a Model

When you create a Web view, you can include optional Web view information.

To add the model coverage or requirements optional view information to a Web view for a model, you must have Simulink Verification and Validation installed.

1 In the Web View dialog box, open the **Optional Views** tab.
2 Select each optional view (for example, **Model Coverage** or **Requirements**) that you want to capture the associated information for.

Tip The **Optional Views** tab appears only if you can access an optional Web view.

Alternatively, in the Simulink Editor, click **Analysis** > **Requirements Traceability** > **Generate Web View**.

View Optional Web View Information

To view the model coverage or requirements optional Web view information in a Web view, you must have Simulink Verification and Validation installed.

In a Web view, click a highlighted block (for example, blocks with an orange border have requirements information). The information for that block appears in the **Informer** pane below the model.

Related Examples

- “Export Models to Web View Files” on page 5-4

More About

- “Optional Web Views” on page 5-17

Add Content with Components

Components

Components are MATLAB objects that specify the content of a report. Add components to specify the types of content that commonly occur in reports. The MATLAB Report Generator provides a set of components for specifying the types of content that commonly occur in MATLAB-based reports. The Simulink Report Generator provides additional components to facilitate generation of reports from Simulink models. You can also create custom components to handle content specific to your application.

Using the Report Explorer, you can interactively combine components to create a report setup that specifies the content of a particular report or type of report. For general information about working with components, see:

- "Insert Components"
- "Set Component Properties"

Use a combination of the following types of components in your report setup file, based on your goals for the report.

Type of Component	Description
"Report Structure Components" on page 6-4	Include a title page, chapters, sections, paragraphs, lists, tables, and other standard document structure elements.
"Table Formatting Components" on page 6-5	Organize generated content into tables.
"Property Table Components" on page 6-6	Display tables with property name/ property value pairs for objects.
"Summary Table Components" on page 6-17	Display tables with specified properties for objects.
"Logical and Looping Components" on page 6-21	Run child components a specified number of times. There are several looping components, including logical loops and Handle Graphics loops.

Component Formatting

When you generate a report, in the Report Options dialog box, in the File format field you specify the type of report output you want. For example, you can generate a report in PDF, HTML, or Microsoft Word format.

For each format, you can choose to apply styles from either of these style definition sources:

- An HTML or Word report template
- A Model Explorer stylesheet for HTML, Word, or PDF.

The output format and the associated template or stylesheet that you select for a report determines most aspects of the formatting of the generated report. For example, a report template or stylesheet typically uses different font sizes for chapter titles and section titles. For details, see “Report Output Format”.

Several components include properties that you can set to specify formatting details for that specific instance of a component. For example, for the `MATLAB Property Table`, you can specify formatting such as whether to display table borders or the alignment of text in table cells.

Report Structure Components

Use report structure components to organize the content of your report into chapters, sections, paragraphs, lists, tables, and other standard document structure elements. The following table summarizes the report structure components.

Component	Usage
"Title Page"	Generate a title page for a report.
"Chapter/Subsection"	Parent components that generate the content of a chapter or chapter subsection.
"Paragraph"	Specify the content and text format of a paragraph of text. Can serve as the parent of one or more text components, enabling use of multiple text formats (for example, bold, regular, or italic) in the same paragraph.
"Text"	Format strings of generated text.
"List"	Generate a list from a cell array of numbers or strings or parent components (for example, Paragraph components) that specify the items in a list. You can create multilevel lists by embedding list components within list components.
"Link"	Generate a hyperlink from one location in a report to another or to an external location on the user's file system or the Worldwide Web.
"Image"	Insert an image into a report.
"Array-Based Table"	Generate a table from a cell array of numbers or strings.
"Table"	Parent a table body component. See "Table Formatting Components".

Table Formatting Components

Use table formatting components to organize generated content into tables. The following table summarizes the table formatting components.

Component	Usage
"Table"	Parent a table body component. Can also parent column specification components and a table header and a table footer component. Specifies properties of the table as a whole (for example, its title, number of columns, and border).
"Table Body"	Parent the rows that make up the table body. Specifies the default vertical alignment of entries in a table body.
"Table Column Specification"	Specify attributes of a table column, such as its width and borders and the default horizontal alignment of column entries.
"Table Entry"	Parent a component that determines a table entry's content, such as a paragraph, image, list, or another table component. Specifies attributes of a table entry, such as the number of rows and columns that it spans.
"Table Footer"	Parent the row components that generate the content of a table footer.
"Table Header"	Parent the row components that generate the content of a table header.
"Table Row"	Parent the table entry components that generate the content of a table row.

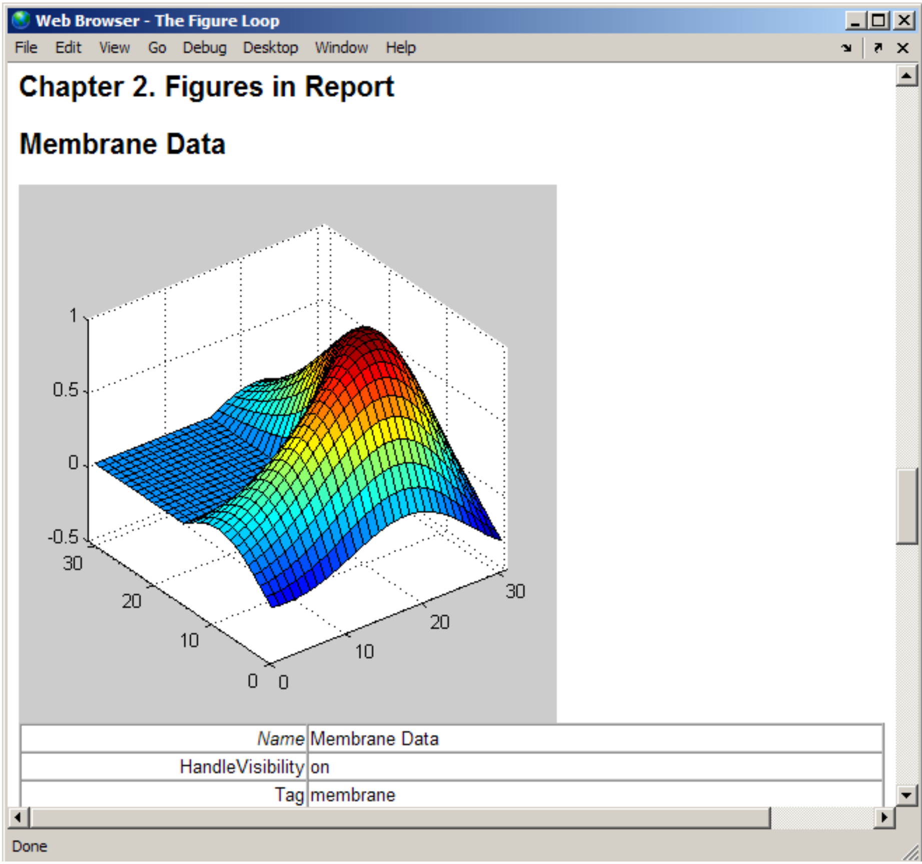
Tip Inserting a Table component into a setup also inserts all the descendant components required to generate a 2x2 table, creating a table template. Edit this template to create a table that suits your needs.

Property Table Components

In this section...

About Property Table Components

Property Table components display property name/property value pairs for objects in tables. The following example shows a property table from the `figloop-tutorial` report.



Many types of property table components are available, including:

- MATLAB Property Table
- Simulink Property Table (requires Simulink Report Generator)
- Stateflow Property Table (requires Simulink Report Generator)

The component used in this example represents MATLAB Report Generator property table components, all of which exhibit similar behavior.

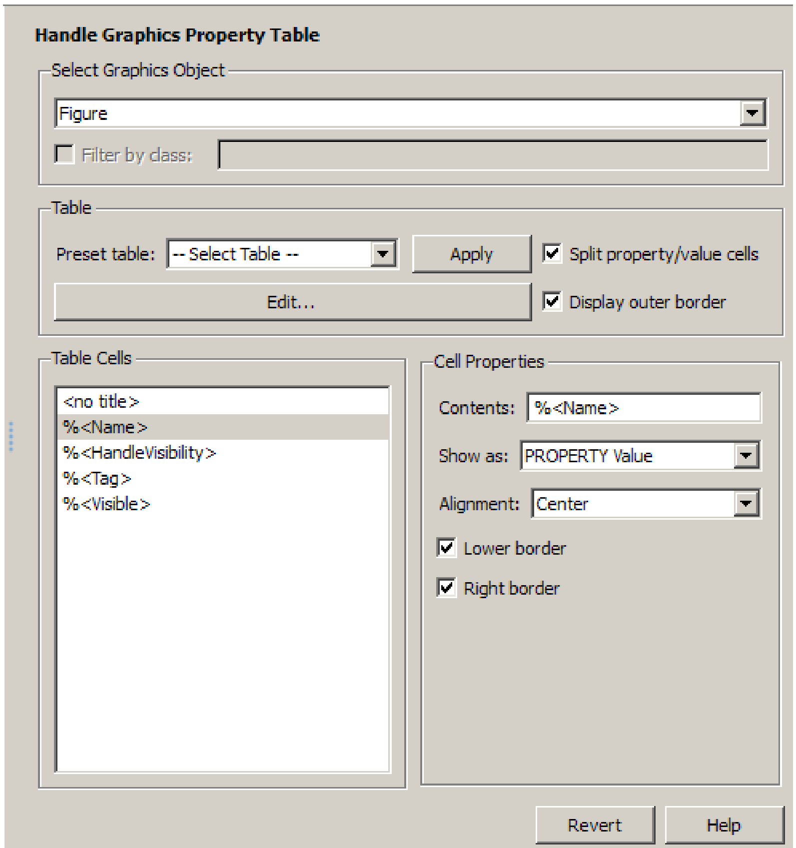
Open the Example Report Template

This example uses the `figloop-tutorial` report template. To open the figure loop tutorial report template, at the MATLAB command line enter:

```
setedit figloop-tutorial
```

Examine the Property Table Output

Property pages for all property table components are similar in form. In the Outline pane, select the `Figure Prop Table` component. To modify table settings, in the Handle Graphics Property Table dialog box, click the **Edit...** button.



Select Object Types

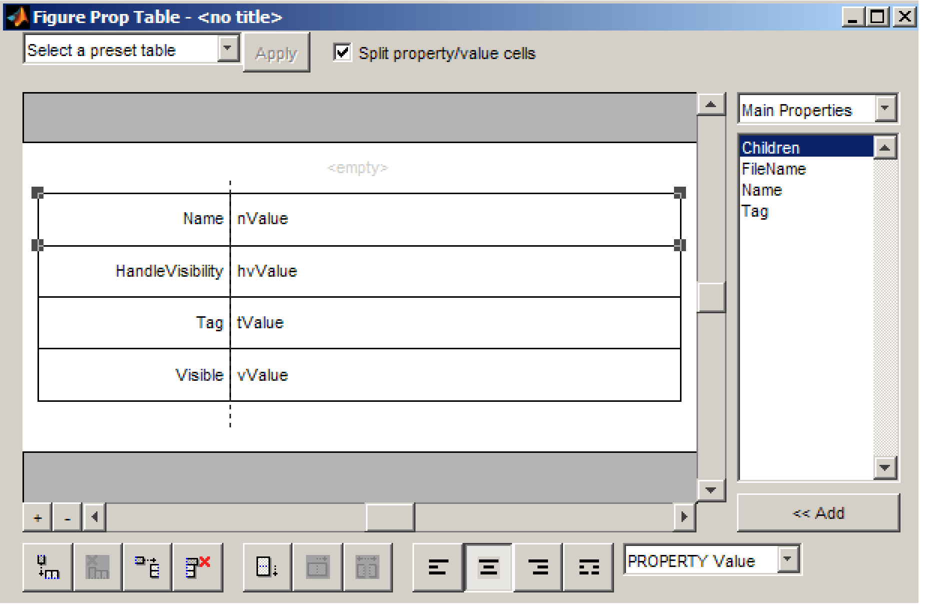
Property table components offer multiple object types on which to report. For example, the Handle Graphics Property Table lets you report on a figure, an axes object, or a Handle Graphics object.

You can select a different object type on which to report in the **Object type** list in the Properties pane for the component.

Display Property Name/Property Value Pairs

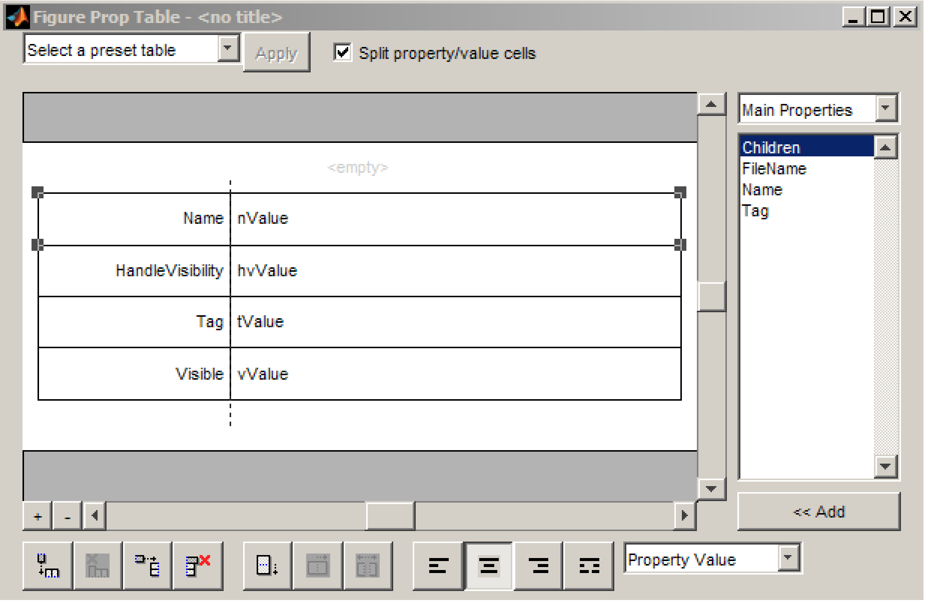
Split Property/Value Cells

1. In the Properties pane for the Handle Graphics Property Table component, clear the **Split property/value cells** check box.
2. Click **Edit**. The table is now in *nonsplit mode*. Nonsplit mode supports more than one property name/property value pair per cell and text.



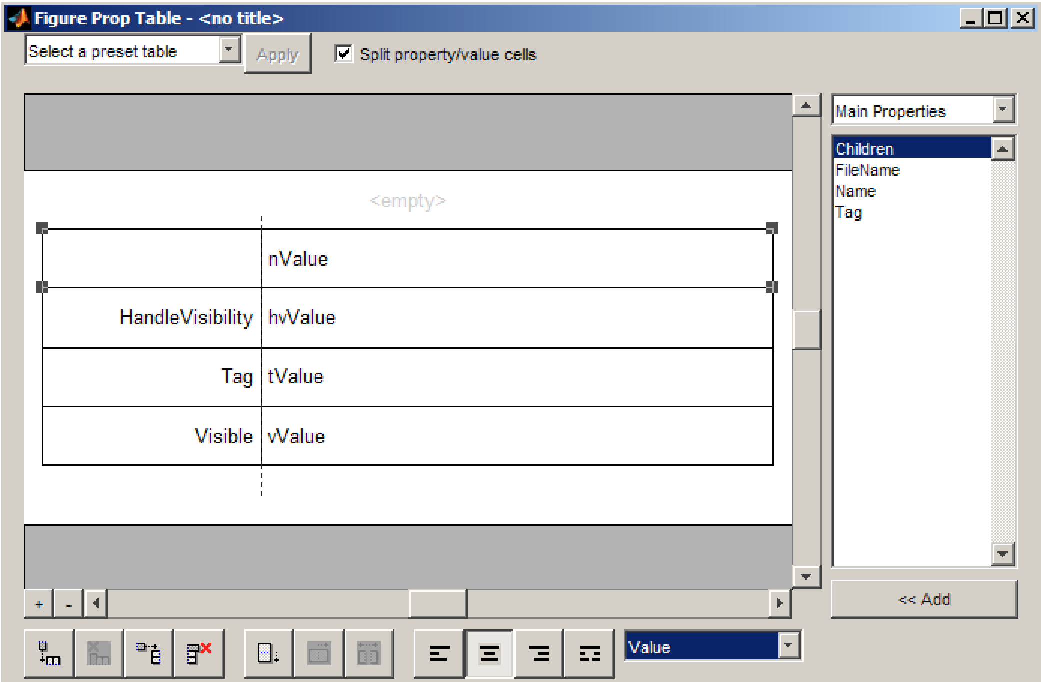
3. For the property name and property value to appear in adjacent horizontal cells in the table, select the **Split property/value cells** check box. The table is now in

split mode. Split mode supports only one property name/property value pair per cell. If more than one property pair appears in a cell, only the first pair appears in the report; all subsequent pairs are ignored.



Display Options

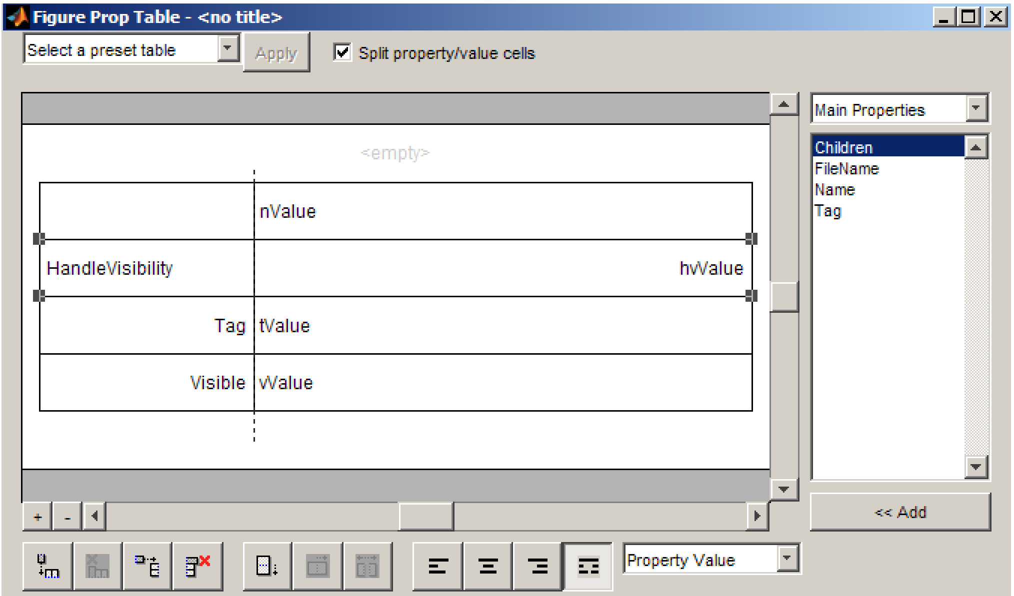
Property name/property value pairs can appear in cells in several ways. To specify how a given property name/property value pair appears in a cell, select that field in the table (for this tutorial, select the `Name` property). Choose `Value` from the display options drop-down list at the bottom of the dialog box. In the selected table row, only the value appears.



Format Options

To specify alignment for text in a given cell, in the toolbar at the bottom of the dialog box use the four justification buttons.

Select the `HandleVisibility` table row. Then select the double-justify button (the last button to the right).



Edit Table Titles

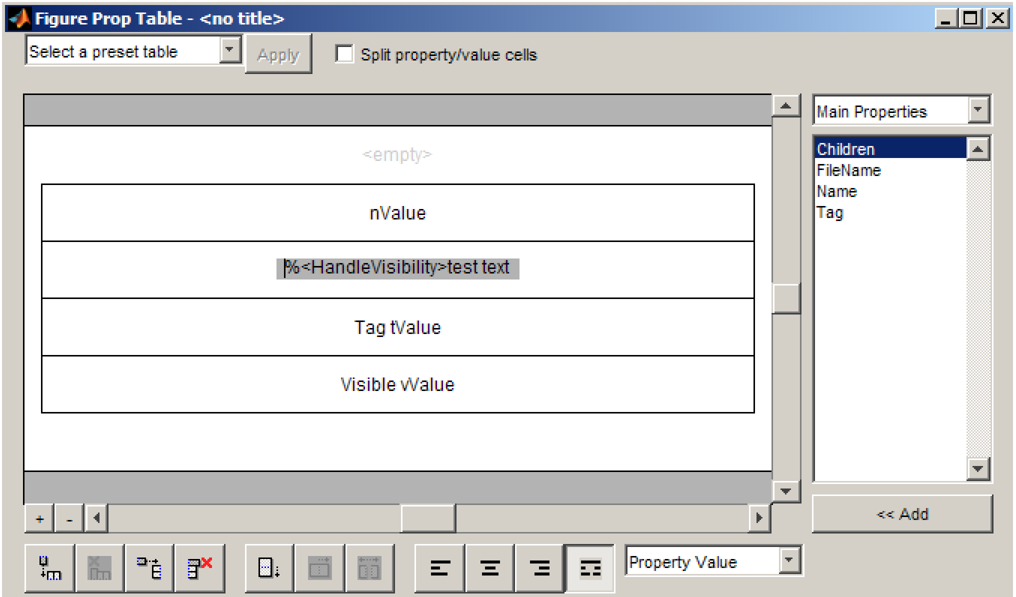
Table titles can contain properties and text. By default, the title of a table is the same as the value of the `%<Name>` property. You can modify this property to modify the table title.

Note: Table titles are always in nonsplit mode.

Enter Text into Table Cells

For the text to be visible, the table must be in nonsplit mode. Clear **Split property/ value cells**.

To enter text into the `HandleVisibility` table cell, double-click the cell. A gray box appears with the label for the cell property.



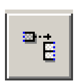
If you type text outside the angle brackets, the text appears as is in the report. Text inside the table brackets must specify a valid property name. If you enter an invalid property name, the property name appears in the report without a property value.

Add, Replace, and Delete Properties in Tables

Adding Table Properties

To add a Handle Graphics property to a table, use the following steps.

1. In the Figure Property Table window, select a table row above which you want add a new property.

2. Click the Add Row Above Current Cell  button

   A new row appears above the current row.

3. Add the property to the new table row.

   a. Select the new table row.

b In the Properties Type drop-down list at the upper-right of the dialog box, select a property type.

c In the **Properties** list, select the property you want to add.

d Click the **<< Add** button, or double-click the property name. The property appears in the table row.

Alternatively, if you know the name of the property you want to add, enter the property name directly into the cell as described in “Enter Text into Table Cells”. For information about adding new table rows, see “Add and Delete Columns and Rows”.

Replace Table Properties

To replace a property in a cell of a table in split mode, follow the instructions in “Adding Table Properties” on page 6-13.

Note: You cannot use these steps to delete a property in a cell when the table is in nonsplit mode.

Delete Table Properties

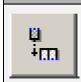
Delete a property by backspacing over it or using the **Delete** key.

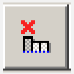
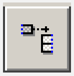
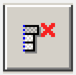
Format Table Columns, Rows, and Cells

Add and Delete Columns and Rows

To add or delete a column or row, select a cell and then click one of the buttons described in the following table.

Note: You cannot delete a row or column when it is the only row or column in the table.

Button	Action
	Add column (added to the left of the selected column)

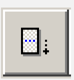
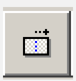
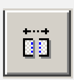
Button	Action
	Delete selected column
	Add row (added above the selected row)
	Delete selected row

Resize Columns

To resize the width of a column, click and drag its vertical borders as needed.

Merge and Split Cells

To merge or split table cells, select a row and then click one of the buttons described in the following table.

Button	Action
	Merge cells downward
	Merge cells to the right
	Split cells

Display or Hide Cell Borders

To toggle cell borders on and off:

1. Place your cursor in a cell and right-click to invoke its context menu.
2. Choose **Cell borders** > **Top**, **Bottom**, **Right**, or **Left** to toggle the specified border on or off.

Zoom and Scroll

You can zoom in and out of the table with the zoom buttons, which are located to the left of the horizontal scroll bar.

Button	Action
	Zoom in
	Zoom out

You can scroll vertically and horizontally using the table scroll bars.

Select a Table

To display property name/property value pairs, you can select a preset table or use a custom table.

- A preset table is built-in and formatted. You can select a preset table in the preset table selection list in the upper-left of the Figure Prop Table window. To apply a preset table, select the table and click **Apply**.
- To create a custom table, select a preset table and modify it to fit your needs by adding and/or deleting rows and properties. You may want to start with the **Blank 4x4** preset table.

Note: You cannot save a custom table as a preset table. If you do so, you lose all changes to the custom table.

Summary Table Components

In this section...

About Summary Table Components

Summary table components insert tables that include specified properties for objects into generated reports. Summary tables contain one object per row, with each object property appearing in a column, as shown in the following summary table in the `figloop-tutorial` report.

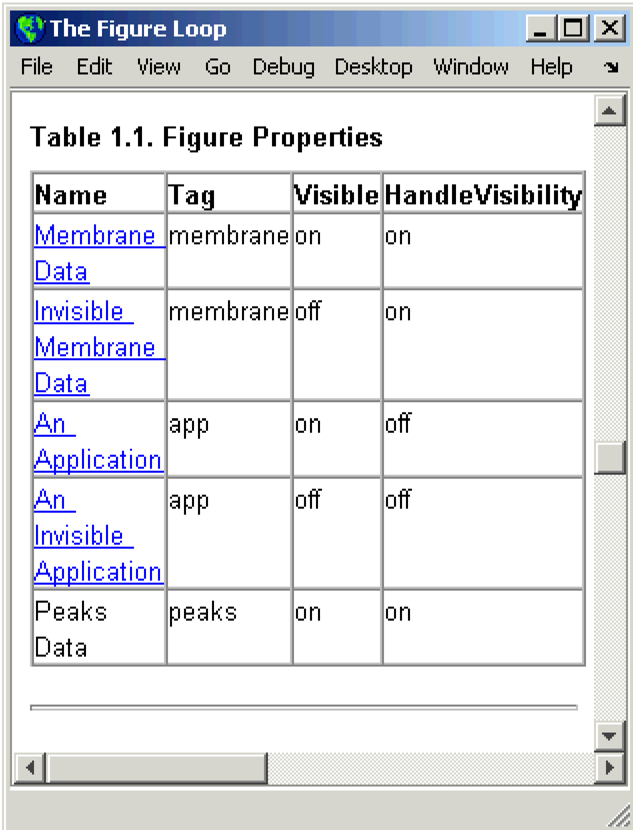


Table 1.1. Figure Properties

Name	Tag	Visible	HandleVisibility
Membrane Data	membrane	on	on
Invisible Membrane Data	membrane	off	on
An Application	app	on	off
An Invisible Application	app	off	off
Peaks Data	peaks	on	on

Many types of summary table components are available, including:

- Handle Graphics Summary Table
- Simulink Summary Table (requires Simulink Report Generator)
- Stateflow Summary Table (requires Simulink Report Generator)

The component used in this example represents MATLAB Report Generator summary table components, all of which exhibit similar behavior

Open the Example Report Template

This example uses the `figloop-tutorial` report template. To open the figure loop tutorial report template, enter the following at the MATLAB command line:

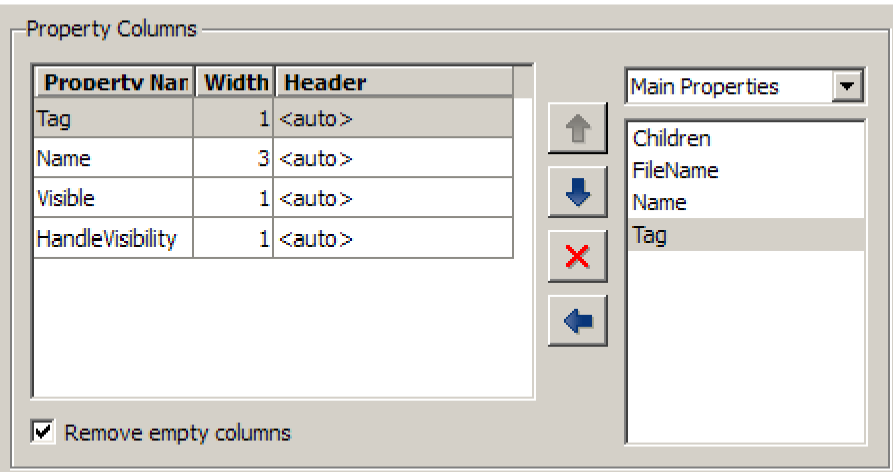
```
setedit figloop-tutorial
```

Select Object Types

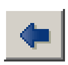
You can use the **Object type** selection list to choose Handle Graphics object types for the summary table, including blocks, signals, systems, and models. The `figloop-tutorial` reports on figure objects.

Add and Remove Properties

You can select object properties to appear in the summary table from the Property Columns pane. To add a property to the summary table, select the property category from the property category drop-down box to the right of the **Property columns** table. Each property category has its own list of properties, which appear in the field under the box. The following figure shows `Main Properties` as the selected category.



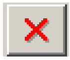
To add a property:

1. Select the category from the property category drop-down box.
2. Select a property in the properties list.
3. Click the Add property  button.

   The property appears in the **Property columns** table.

To remove a property from the table:

1 Select the property in the **Property columns** table.

2 Click the Delete property  button.

The property name is removed from the **Property columns** table.

Note: After making changes in the Report Explorer, click **Apply** to make the changes take effect.

You can define your own properties by entering their names into the **Property columns** table using valid variable notation. For more information, see "%<VariableName> Notation" on the `Text` component reference documentation.

Set Relative Column Widths

To apply a relative column width to the summary table columns in the generated report, double-click on the `Width` column of a row in the **Property columns** table . If you do not specify a value for this field, column widths automatically set.

Set Object Row Options

You can use the Object Rows pane to set options for table rows, including anchor, filtering, and sorting options. Select **Insert anchor for each row** to place an anchor in each table row in the report. Use the **Include figures** list to specify what objects to include in the summary table.

Summary table components in `figloop-tutorial` report on figure objects. For more information on options for these figure objects, see the following sections:

- "Loop on the Current Figure"
- "Loop on Visible Figures"
- "Loop on Figures with Tags"

Logical and Looping Components

Logical and looping components execute conditionally, determining when a child component executes or how many times a child component executes.

A *looping component* runs its child components a specified number of times. There are several looping components, such as logical loops, Handle Graphics loops, and model and chart loops. For model and chart loops, you can control aspects such as the order in which the report sorts blocks.

For an example that uses loop components, see “Edit Figure Loop Components”.

You can use loop context functions with loop components. For details, see:

- “Filter with Loop Context Functions”
- “Loop Context Functions”

Filter with Loop Context Functions

In this section...

Use loop context functions to filter the modeling elements to report on and to perform special reporting on specific elements.

In the following example, in a Block Loop component, you use `RptgenSL.getReportedBlock` in a Logical If component to report on targeted blocks within a Block Loop component.

For a summary of loop context functions, see "Loop Context Functions" on page 6-24.

Create and Save the Setup File

1. Open the f14 model.
2. At the MATLAB command prompt, enter:

   ```
   report
   ```
3. In the Report Explorer, select **File** > **New**.
4. In the Properties pane, set **Directory** to `Present working directory`.
5. Save the setup file as `inport_outport.rpt`.

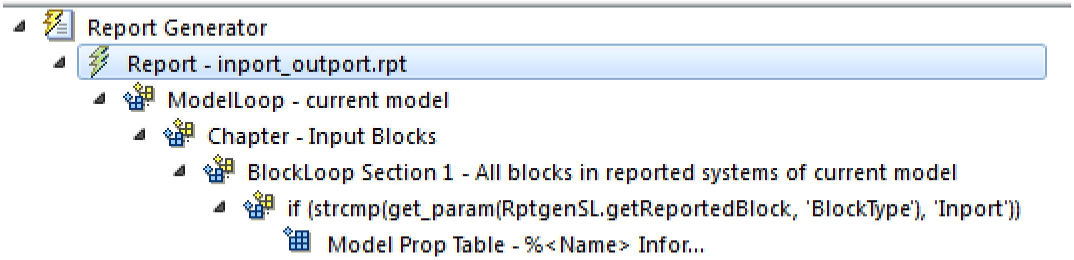
Add Components

Add these components to the report, in order.

From this Library Folder	Add this Component	Set this Property
`Simulink`	Model Loop	N/A
`Formatting`	Chapter	**Title** to `Inport Blocks`
`Simulink`	Block Loop	N/A
`Logical and Flow Control`	Logical If	**Test Expression** to `strcmp(get_param...`

From this Library Folder	Add this Component	Set this Property
		`(RptgenSL.getReportedBlock,'BlockType'),...` `'Inport')`
`Simulink`	Simulink Property Table	N/A

The report setup file looks like this:

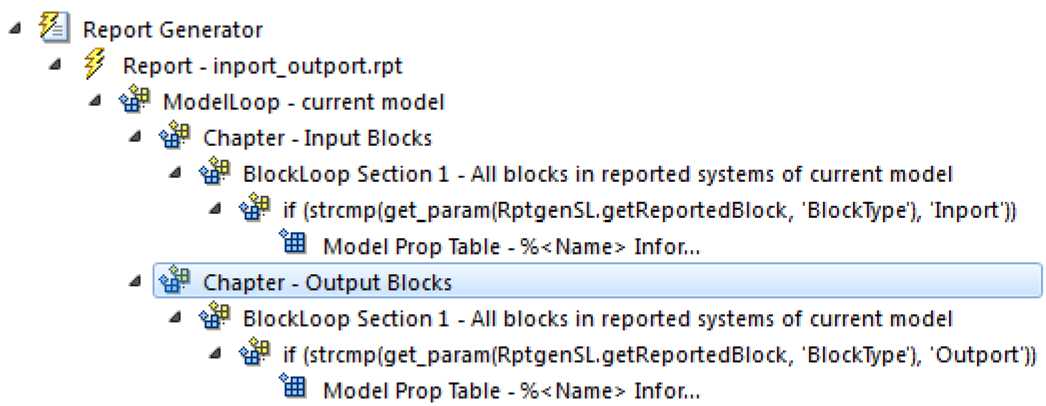


Run the Report

1 Select `inport_outport.rpt`.
2 From the context menu, select **Report**.

The report includes a chapter with properties for the Inport blocks only.

If you wish, create a second chapter that reports on Outport blocks only, as shown below.



Loop Context Functions

In this section...

You can use these loop context functions in similar ways as shown in "Filter with Loop Context Functions" on page 6-22.

For Simulink Modeling Elements

Modeling Element	Looping Component	Function
Simulink modeling elements		
Block	Block Loop	`RptgenSL.getReportedBlock`
Signal	Signal Loop	`RptgenSL.getReportedSignal`
System	System Loop	`RptgenSL.getReportedSystem`
Model	Model Loop	`RptgenSL.getReportedModel`

For Stateflow Modeling Elements

Modeling Element	Looping Component	Function
Object	Object Loop	`RptgenSF.getReportedObject`
State	State Loop	`RptgenSF.getReportedState`
Chart	Chart Loop	`RptgenSF.getReportedChart`

Edit Figure Loop Components

In this section...

Figure Loop in a Report

This example uses the Figure Loop, which is representative of many types of loops. The Figure Loop component runs its child components several times. In each iteration, the Figure Loop applies its child components to Handle Graphics figures. The `figloop-tutorial` report setup file creates a report that documents several Handle Graphics figures.

1 At the MATLAB command prompt, enter:

```
setedit figloop-tutorial
```

2 To display the Handle Graphics figures, enter:

```
figloopfigures
```

The figures `Membrane Data`, `An Application`, and `Peaks Data` appear on the screen because their `visible` property is `'on'`. The `Invisible Membrane Data` and `An Invisible Application` figures do not appear on screen because their `visible` property is `'off'`. These invisible figures exist, but they are hidden.

3 In the Report Explorer, in the Outline pane on the left, select the Figure Loop component called `Figure Loop Section 1`.

The Properties pane for the Figure Loop component appears.

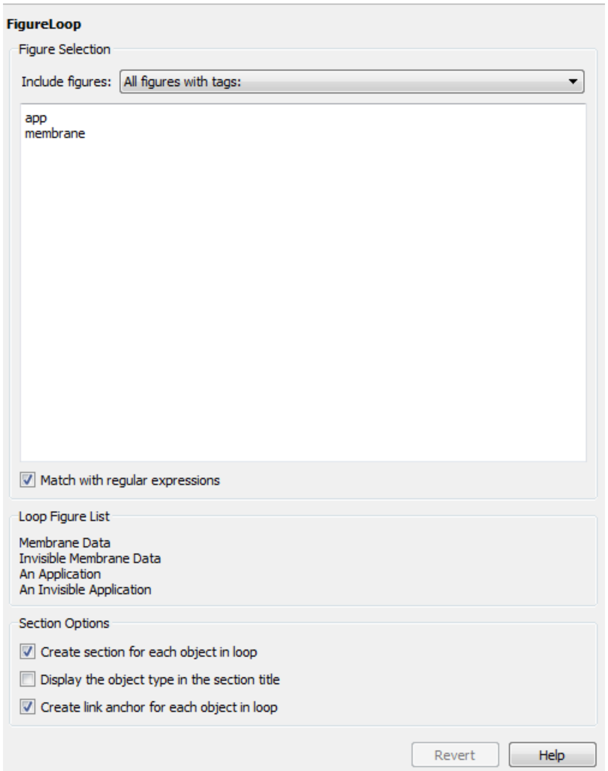


Figure Properties

Figure properties control which figures appear in the report. Table 1.1 of the `figloop-tutorial` report includes a summary of the properties of the figures used in this tutorial.

Table 1.1. Figure Properties

Name	Tag	Visible	HandleVisibility
Membrane Data	membrane	on	on
Invisible Membrane Data	membrane	off	on
An Application	app	on	off
An Invisible Application	app	off	off
Peaks Data	peaks	on	on

For this example, do not change these properties. For more information, see "Add, Replace, and Delete Properties in Tables" on page 6-13.

Loop on the Current Figure

To include only the current figure in the report, select `Current figure only` from the **Include figures** selection list. The current figure is the figure that is current when the report generates. This figure may not be the same figure that you selected as the current figure in the Report Explorer before report generation. For example, if the report generation process creates figures in your report, the last figure created with `HandleVisibility` set to `'on'` is the current figure.

Loop on Visible Figures

To include snapshots of all visible figures in your report, in the **Include figures** selection list, select `Visible figures`. This option inserts a snapshot and Property Table for all figures that are currently open and visible.

1. Select the **Data figures only (Exclude applications)** option to exclude figures from the loop whose `HandleVisibility` parameter is `'off'`.
2. To generate the report, in the Report Explorer toolbar click the **Report** button.

In the generated report, scroll down to "Chapter 2 Figures in Report." The `Membrane Data` and `Peaks Data` figures appear in the generated report.

Loop on Figures with Tags

To include figures with specified tags in the report:

1. In the **Include figures** selection list, select the `All figures with tags` option.
2. In the list of tags, delete `membrane`.
3. Click **Report** to generate the report.

The `An Application` and `An Invisible Application` figures appear in the report. They both have an `app` tag.

Modify Loop Section Options

In a loop, a *section* refers to a space in the generated report in which information, including text, images, and tables, appears. You can alter the appearance of sections in each loop appear in the report by using the options in the Figure Loop component's Section Options pane.

- **Create Section for Each Object in Loop** — Create an individual section for each object found in the loop, using the object title as the section title. This option is useful when a loop does not contain a Chapter/Subsection component that organizes the loop results.
- **Display the Object Type in the Section Title** — Precede section titles with object titles. Enable this option by selecting **Create section for each object in loop**. For example:

  1. Enter `membrane` back in the list of tags.
  2. Generate the `figloop-tutorial` report.

     The figures produced by the loop are:

     ```
     Membrane Data
     Invisible Membrane Data
     An Application
     An Invisible Application
     ```
  3. Enable the **Create section for each object in loop** option.
  4. Enable the **Display the Object Type in the Section Title** option.
  5. Generate the `figloop-tutorial` report.

     The figures produced are now:

     ```
     Figure - Membrane Data
     Figure - Invisible Membrane Data
     Figure - An Application
     Figure - An Invisible Application
     ```

     The figures produced are now:

     ```
     Figure - Membrane Data
     Figure - Invisible Membrane Data
     Figure - An Application
     Figure - An Invisible Application
     ```
- **Create a Link Anchor for Each Object in Loop** — Create a hyperlink to the object in the generated report.

7

Compare Simulink Model XML Files

About Simulink Model XML Comparison

In this section...
"Creating XML Comparison Reports" on page 7-2
"Using XML Comparison Reports" on page 7-3

Creating XML Comparison Reports

If you have Simulink Report Generator software, you can compare XML text files from Simulink models.

You can select a pair of Simulink models to compare their XML files. You can use models from any version of Simulink. The XML comparison tool produces a comparison report based on the SLX files. You can use the report to explore the differences, view the changes highlighted in the original models, and merge differences.

You can access the XML comparison tool from:

- The MATLAB Current Folder browser context menu
- The MATLAB Comparison Tool
- The MATLAB command line
- The Simulink Editor **Analysis** menu
- The Simulink Project Modified Files view

The XML comparison tool compares the XML files using the "Chawathe" algorithm, as described in the paper:
Change Detection in Hierarchically Structured Information, Sudarshan Chawathe, Anand Rajaraman, and Jennifer Widom; SIGMOD Conference, Montreal, Canada, June 1996, pp. 493-504.

This conference paper is based upon work published in 1995: see `http://dbpubs.stanford.edu:8090/pub/1995-45`.

The Simulink Report Generator XML comparison functionality is an extension of the MATLAB Report Generator XML comparison feature.

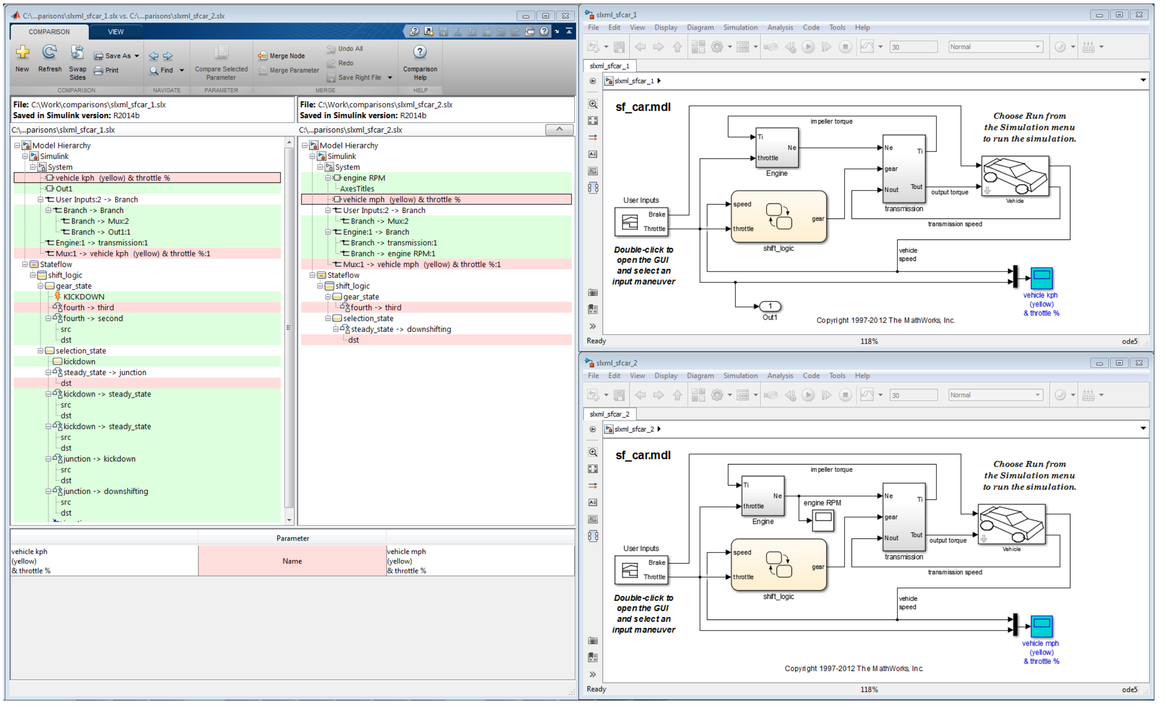
You can use the XML comparison tool with both model file formats, SLX and MDL. If the selected files are `.mdl` files, the XML comparison tool first exports the `.mdl` files to

SLX files in a temporary directory. The XML comparison tool then produces a comparison report based on the SLX files.

For more information on creating reports, see “Select Simulink Models for XML Comparison” on page 7-5.

Using XML Comparison Reports

You can display XML comparison reports in the MATLAB Comparison Tool. The comparison tool processes the output of the XML comparison into an interactive report with links that you can click to reverse annotate from the XML tag comparison to the corresponding Simulink models. “Reverse annotation” means when you click items in the report, Simulink Report Generator displays the corresponding items highlighted in the original models, as shown in the following example.



The XML comparison report shows a hierarchical view of the portions of the two XML files that differ. The report does not show sections of the files that are identical.

If the files are identical you see a message reporting there are no differences.

If files have not been saved, you see an error message informing you that you must save modified or newly created models before running an XML comparison.

Note: It might not be possible for the analysis to detect matches between previously corresponding sections of files that have diverged too much.

Change detection in the Chawathe analysis is based on a scoring algorithm. Items match if their Chawathe score is above a threshold. The Simulink Report Generator implementation of Chawathe's algorithm uses a comparison pattern that defines the thresholds assigned to particular node types (e.g., “block”). For more information, see “How the Matching Algorithm Works” in the MATLAB Report Generator documentation.

For more information on using the report, see “Compare Simulink Model XML Files” on page 7-8.

To control reverse annotation, see “Display Items in Original Models” on page 7-17.

To merge differences, see “Merge Simulink Models from the Comparison Report” on page 7-21.

For more information about the Comparison Tool, see “Comparing Files and Folders” in the MATLAB documentation.

Select Simulink Models for XML Comparison

In this section...

To learn what you can do with XML comparison reports, see "About Simulink Model XML Comparison" on page 7-2.

Select Files from the Simulink Editor

To compare files using the Simulink Editor:

1. Select **Analysis** > **Compare Simulink XML Files**.

   The Select Files or Folders for Comparison dialog box opens.
2. If the Editor currently displays a model, the current model name and path appear automatically selected in the **First file or folder** edit box. Use the browse buttons to locate and select files for the first and second model files.
3. When you click **Compare**, the XML comparison tool performs the analysis, and displays the resulting report in the Comparison Tool.

Select Files from the Current Folder Browser

To compare two files from the Current Folder browser:

- For two files in the same folder, select the files, right-click and select **Compare Selected Files/Folders**.
- To compare files in different folders:

  1. Select a file, right-click and select **Compare Against**

2. Select the second file to compare in the Select Files or Folders for Comparison dialog box.
3. Leave the default **Comparison type**, `Simulink XML text comparison`.
4. Click **Compare**.

If the selected files are XML or model files, the XML text comparison tool performs a Chawathe analysis and displays a report in the Comparison Tool.

For more information about comparisons of other file types (e.g., text, MAT, or binary) with the Comparison Tool, see "Comparing Files and Folders" in the MATLAB documentation.

Select Files from a Simulink Project

If you have a Simulink Project using source control, you can create an XML comparison report from the Modified Files view of the Simulink Project Tool. For details, see "Project Management ".

Select Files from the Comparison Tool

To compare files using the Comparison Tool, from the MATLAB Toolstrip, in the **File** section, select the **Compare** button. In the dialog box select files to compare.

If the selected files are XML or model files, the XML text comparison tool performs a Chawathe analysis and displays a report in the Comparison Tool.

Select Files from the Command Line

To compare XML files from the command line, enter

```
visdiff(filename1, filename2)
```

where `filename1` and `filename1` are XML files or Simulink models.

If the files are models, the XML comparison tool performs the comparison on the XML files. This XML comparison functionality is an extension to the MATLAB `visdiff` function. `visdiff` produces a report in the Comparison Tool.

To create an `xmlcomp.Edits` object at the command line without opening the Comparison Tool, enter:

```
Edits = slxmlcomp.compare(modelname_A,modelname_B)
```

See “Export Results to the Workspace” on page 7-24 for information about the `xmlcomp.Edits` object.

Choose a Comparison Type

If you specify two XML or model files to compare using either the Current Folder Browser or the `visdiff` function, then the Comparison Tool automatically performs the default comparison type. The defaults are `XML text comparison` for XML files and `Simulink XML text comparison` for model files.

To change comparison type, either create a new comparison from the Comparison Tool, or use the **Compare Against** option from the Current Folder browser. You can change comparison type in the Select Files or Folders for Comparison dialog box. For example, if you want the MATLAB text differences report for XML or model files, change the comparison type to `Text comparison` in the dialog before clicking **Compare**.

Examples of XML Comparison

For examples with instructions, see:

- `slxml_radar_matlab_function`
- `slxml_sfcar`
- `slxml_truthtables`

For information on using and understanding the report and the XML comparison functionality, refer to “Compare Simulink Model XML Files” on page 7-8.

Compare Simulink Model XML Files

In this section...

Navigate the Simulink XML Comparison Report

You can select a pair of Simulink models to compare their XML files. You can use models from any version of Simulink. The XML comparison tool produces a comparison report based on the SLX files. You can use the report to explore the differences, view the changes highlighted in the original models, and merge differences.

The XML Comparison report shows changes only, not the entire XML text file contents. The report shows a hierarchical view of the portions of the two XML files that differ. The report does not show sections of the files that are identical. To learn about the report, see "About Simulink Model XML Comparison" on page 7-2.

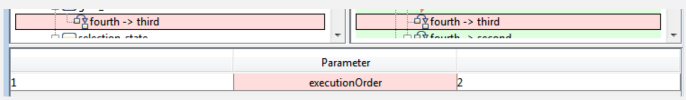
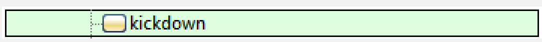
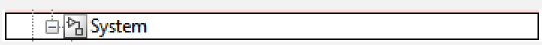
To *step through differences*, use the **Comparison** tab on the toolstrip. To move to the next or previous group of differences, on the **Comparison** tab, in the **Navigate** section, click the arrow buttons to go to the previous or next difference. See "Step Through Changes" on page 7-10.

You can also click to select items in the hierarchical trees and observe the following display features:

- Selected items appear highlighted in a box.
- If the selected item is part of a matched pair it is highlighted in a box in both left and right trees.

- When you select an item, the original model displays and the corresponding item is highlighted. See "Explore Changes in the Original Models" on page 7-11.

Report item highlighting indicates the nature of each difference as follows:

Type of report item	Highlighting	Notes
Modified	Pink	Modified items are matched pairs that differ between the two files. When you select a modified item it is highlighted in a box in both trees. Example of a modified pair of nodes:  Changed parameters for the selected pair are displayed in a separate **Parameters** panel for review. If strings are too long to display in the **Parameters** table, right-click and select **Compare as Text** to open a new comparison of the parameters. Example of modified parameters: 
Unmatched	Green	When you select an unmatched item it is highlighted in a box in one tree only. Example of an unmatched node: 
Container	None	Rows with no highlighting indicate a container item that contains other modified or unmatched items. Example of a container node: 

Icons indicate the category of item, for example: model, subsystem, Stateflow machine or chart, block, line, parameter, etc.

To expand or filter the tree view, use the **View** tab controls on the toolstrip for the following functions:

- **Expand All** — Expands every item in the tree.

Tip Right-click to expand or collapse the hierarchy within the selected tree node.

- **Collapse All** — Collapses all items in the tree to the most compact view possible.
- **Filter** — Opens the Filter list. Select check boxes to enable or disable display of categories of changes in the report. Use the filters to show only the changes you are interested in. By default the report hides all nonfunctional changes, such as repositioning of items. Turn off filters to explore *all* differences including nonfunctional changes. See “Filter Out Differences” on page 7-13.

If you want to swap the files, on the **Comparison** tab, select **Swap Sides**. The report swaps the sides and reruns the comparison. This action runs the Chawathe analysis again. **Refresh** also runs the Chawathe analysis again to refresh the comparison report.

To create a new report, see “Select Simulink Models for XML Comparison” on page 7-5.

For examples with instructions, see also “Examples of XML Comparison” on page 7-7 .

Step Through Changes

On the **Comparison** tab, in the **Navigate** section, if you click the **Next** arrow button (or press the Down key) repeatedly, you can step through every group of changes in the report, in the following order:

1. The first time you click **Next**, it selects the first changed (pink) or inserted (green) node on the left tree.
2. Step through the differences with the **Next** button.

   - When selected items have a match in the right tree then they are also highlighted.
   - Next skips white nodes with no color background. White nodes are parts of the hierarchy that contain no differences.
   - If there is an insertion or deletion with child nodes, **Next** skips the child nodes if they are all also insertions or deletions. For example, if you insert a subsystem, **Next** selects the top subsystem node, then skips all the nodes inside the subsystem (if they are all also insertions) and selects the next difference.

- **Next** minimizes context switching when highlighting in models. **Next** steps through all differences at the same level of the model, subsystem, or chart, in both left and right trees in the report, before moving to the next level of the report. For example, you step through all differences in a subsystem in the left and right trees, before moving to another subsystem.

3 When you have stepped through all changes, **Next** returns to the beginning of the left tree.

If you click an item in the report, the **Next/Previous** controls will step through changes from the point you selected.

Explore Changes in the Original Models

When you compare the XML text files from Simulink® models, you can choose to display the corresponding items in the original models when you select report items. You can use this reverse annotation function to explore the changes in the original models. When you select an item, the report invokes reverse annotation to the original model and highlights the corresponding item in the model.

Control the display by using the **View** tab **Highlight in Models** button and the **Always Highlight** check box.

Tip Click a Subsystem contents node to see the report highlight all visible modified objects in the subsystem.

For details, see "Display Items in Original Models" on page 7-17.

Merge Differences

Tip You can only merge from left to right. If you want to merge into the other file, use **Swap Sides** before you start merging. Swap Sides reverts any merges already made and creates a new comparison report for the original files.

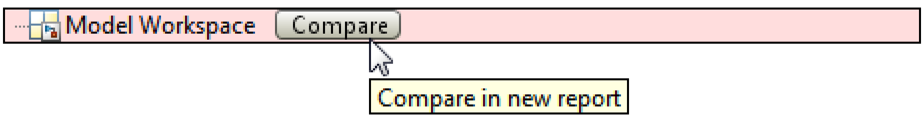
To merge a selection, use the following buttons on the **Comparison** tab, in the **Merge** section:

- **Merge Node** — Merge the selected node from the left side of the report to the right.
- **Merge Parameter** — Merge the selected parameter from the left side of the report to the right.
- **Undo All** — Revert all merge operations.

For more information, see "Merge Simulink Models from the Comparison Report" on page 7-21.

Open Child Comparison Reports for Selected Nodes

If additional comparisons are available for particular nodes, you see a **Compare** button to open a report for that pair of nodes. For example, if there are differences in the Model Workspace, you can click **Compare** to open a new report to explore differences in variables.



You can open child reports for parameters, MATLAB Function blocks, truth tables and Model Workspaces.

- To compare parameters, click the Parameters pane, then on the **Comparison** tab select **Compare Selected Parameter**. This opens a new report for the currently selected pair of parameters. Use this when the report cannot display all the details in the Parameters pane, e.g., long strings or a script.
- If the original models contain MATLAB Function block components, and if differences are found, the XML comparison tool lists them in the Stateflow section of the report. Click the **Compare** button at the end of the MATLAB Function block report items to open new comparisons in the Comparison Tool, showing the text difference reports for the MATLAB Function block components. You can merge differences in MATLAB Function block code from the text comparison report. See "Merge Simulink Models from the Comparison Report" on page 7-21, and the example `slxml_radar_matlab_function`.
- If the original models contain truth tables, and if differences are found, the XML comparison tool lists them in the Stateflow section of the report.
    - Click the **Compare** button at the end of the MATLAB Function node to see a summary of all changes.

- Click the `truthtable` node to reverse annotate and display both truthtable editors.
- Click the **Compare** button at the end of the `Condition Table` node to open a new text comparison showing only Condition differences.
- Similarly click the **Compare** button for `Action Table` to view only Action changes.

See the example `slxml_truthtables`.

Understand the Report Hierarchy and Matching

To understand the report, it is helpful to understand how the Chawathe results from the XML text files relate to the original models.

Hierarchical node tags (such as subsystem tags in the `.xml` file) appear twice in the tree as nested nodes. This is because the container node and the contents can have separate differences in their properties. This feature of the XML report allows you to distinguish between property differences of the node itself, and differences contained within nodes nested inside.

To understand matching results within an XML comparison report, see "How the Matching Algorithm Works" in the MATLAB Report Generator documentation.

Note: It might not be possible for the analysis to detect matches between previously corresponding sections of files that have diverged too much.

If you cannot see changes you expected to see in the report, on the **View** tab, click the **Filter** button to turn off filters and see *all* identified changes. See "Filter Out Differences" on page 7-13.

Filter Out Differences

You can use the **Filter** button on the **View** tab to control display of categories of changes. Turn off filtering to view *all* identified changes.

In the **Filter** list, select check boxes to enable or disable display of categories of changes in the report. Use the filters to show only the changes you are interested in. By default the report hides all nonfunctional changes, such as repositioning of items. Turn off filters to explore all differences including nonfunctional changes. Try this if you cannot see changes you expected to see in the report.

Categories for filtering include:

- **Hide changes in lines**. Hide all changes to signal lines including functional changes.
- **Hide nonfunctional changes**. The report processing identifies certain items in the XML file as nonfunctional, for example, tags representing parameters such as block, system, chart or label positions, font and color settings for blocks and lines, and system print and display settings. The report processing tries to identify "consequential" changes as nonfunctional (that is, changes as a consequence of another change). For example, if a block name changes from `block_A` to `block_B`, a line emerging from that block has a change in its source block parameter. This change in the line parameters is considered nonfunctional. Lines are highly functional, but line changes can be very noisy because of changes in blocks they connect to.
- **Hide changes in graphical interface**. This information is a summary of inports and outports at the top level of the model. Filter graphical interface changes to avoid duplication in the report, as any changes in root ports are also reported as functional changes where you can use reverse annotation.
- **Hide changes in block parameter defaults**. Hiding changes in defaults can avoid duplication in the report, as any changes in blocks are also reported as functional changes where you can use reverse annotation. Block parameter defaults are an undocumented part of the Simulink XML file that store the default parameters for the blocks used in a model.

Exceptions

The report does *not* filter out changes to Block and System names, annotations, and Stateflow Notes as nonfunctional, even though changes to these items do not affect the outcome of simulation. The report always displays these changes to facilitate review of code changes, because they can contain important information about users' intentions.

In certain rare cases the report filters out changes that can impact the behavior of the design. By default moves are filtered as nonfunctional, but in the following cases moves can change design behavior:

- Moving blocks can in some cases change the execution order.
- In a Stateflow chart, if you move states or junctions so that they intersect, the model fails to simulate.

To view these types of changes in the report, turn off the filter for nonfunctional changes.

Change Color Preferences

You can change and save your diff color preferences for the Comparison tool. You can apply your color preferences to all comparison types.

1. On the MATLAB Home tab, click **Preferences**.
2. In the Preferences dialog box, under **MATLAB**, click **Comparison**.
3. Edit color settings as desired for differences and merges. View the colors in the **Sample** pane.

   The **Active Settings** list displays **Default (modified)**.
4. To use your modified settings in the comparison, click **Apply** and refresh the comparison report.
5. To return to the default color settings, in the Preferences dialog box, click **Reset** and click **Apply**. Refresh the comparison report.
6. If you want to save your modified color preferences for use in future MATLAB sessions, click **Save As**. Enter a name for your color settings profile and click **OK**.

   After saving settings, you can select them in the **Active Settings** list.

Save Comparison Results

To save your comparison results, use these **Comparison** tab buttons:

- **Save As** > **Save as HTML** — Opens the Save dialog box, where you can choose to save a printable version of the XML comparison report. See "Save Printable HTML Report" on page 7-24.
- **Save As** > **Save to Workspace** — Export XML comparison results to workspace. See "Export Results to the Workspace" on page 7-24.

Related Examples

- "Select Simulink Models for XML Comparison" on page 7-5
- "Display Items in Original Models" on page 7-17
- "Merge Simulink Models from the Comparison Report" on page 7-21

- “Compare XML from Models Managed with Subversion” on page 7-30
- “Review Changes”
- “Source Control in Simulink Project”

More About

- “About Simulink Model XML Comparison” on page 7-2
- “Comparing XML Files from Models with Identical Names” on page 7-27
- “Work with Referenced Models and Library Links” on page 7-28

Display Items in Original Models

In this section...

Highlighting in Models

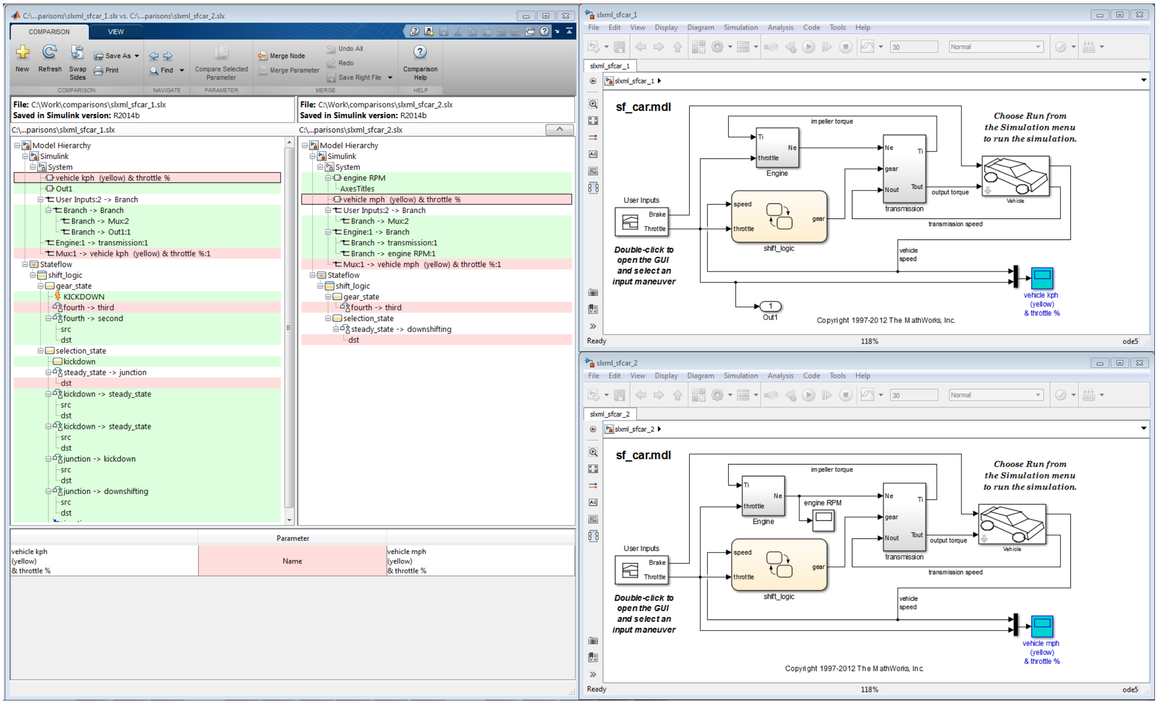
When you compare the XML text files from Simulink models, you can choose to display the corresponding items in the original models when you select report items. You can use this *reverse annotation* function to explore the changes in the original models. When you select an item, the report invokes reverse annotation to the original model and highlights the corresponding item in the model.

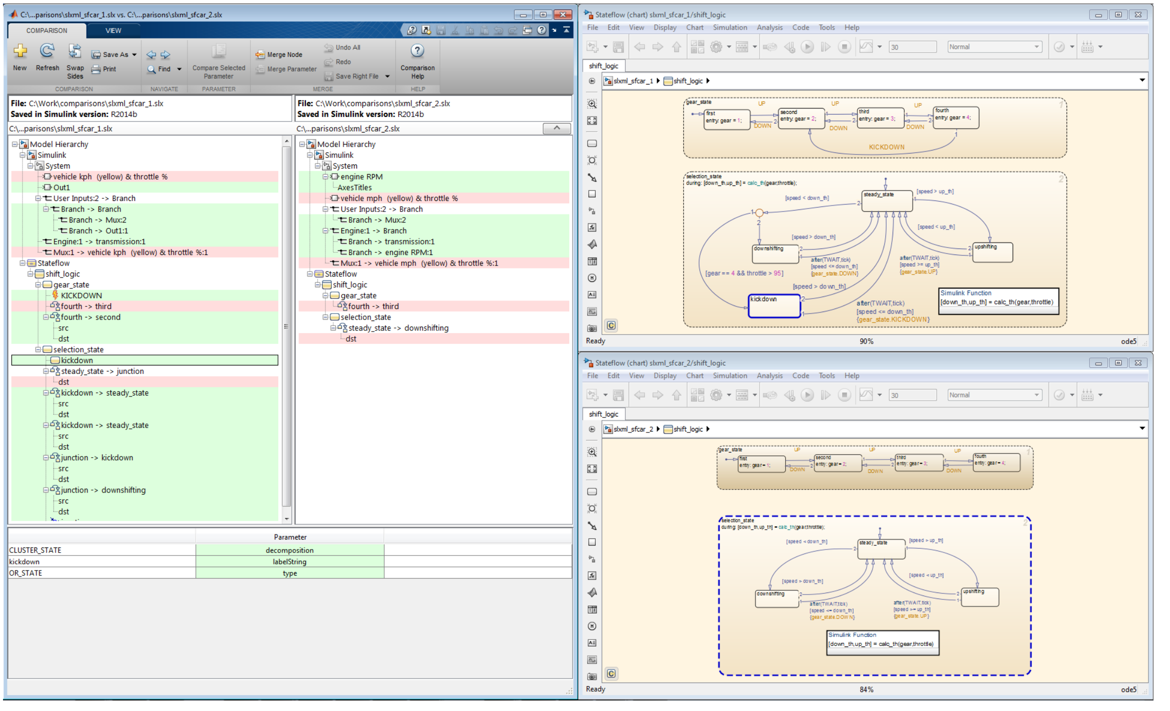
Tip If you click a Subsystem contents node, the report highlights all visible modified objects in the subsystem.

Click a report entry to view the highlighted item (or its parent) in the model:

- If the item occurs in both models, they both appear with highlighting.
- If there is no match for the item, the unmatched report item row is green. It is considered unique and appears highlighted by itself. An appropriate system in the other model also displays to show the context of the missing item.
- If the XML comparison tool cannot highlight an item directly (e.g., configuration parameters), then it highlights the nearest ancestor of the selected node.

The following screenshots show reverse annotation of Simulink and Stateflow items in original models using the example slxml_sfcar.


File: C:\Work\comparisons\slxml_sfcar_1.slx
Saved in Simulink version: R2014b
File: C:\Work\comparisons\slxml_sfcar_2.slx
Saved in Simulink version: R2014b
Model Hierarchy
Simulink
System
vehicle kph (yellow) & throttle %
Out1
User Inputs:2 -> Branch
Branch -> Branch
Branch -> Mux:2
Branch -> Out1:1
Engine:1 -> transmission:1
Mux:1 -> vehicle kph (yellow) & throttle %:1
Stateflow
shift_logic
gear_state
KICKDOWN
fourth -> third
fourth -> second
selection_state
kickdown
steady_state -> junction
kickdown -> steady_state
junction -> kickdown
junction -> downshifting
engine RPM
AxesTitles
vehicle mph (yellow) & throttle %
Engine:1 -> Branch
Branch -> transmission:1
Branch -> engine RPM:1
Mux:1 -> vehicle mph (yellow) & throttle %:1
steady_state -> downshifting
Parameter
Name
slxml_sfcar_1
slxml_sfcar_2
sf_car.mdl
impeller torque
Engine
transmission
output torque
Vehicle
transmission speed
User Inputs
Brake
Throttle
shift_logic
vehicle speed
Choose Run from the Simulation menu to run the simulation.
Double-click to open the GUI and select an input maneuver
Out1
engine RPM
vehicle kph (yellow) & throttle %
vehicle mph (yellow) & throttle %
Copyright 1997-2012 The MathWorks, Inc.
Ready
118%
ode5



Control Highlighting in Models

To control highlighting in models, on the **View** tab in the Comparison Tool, select or clear the check box **Always Highlight**. You can click the **Highlight in Models** button to highlight the currently selected report node at any time. This can be useful if you turn off automatic highlighting and only want to display specific nodes.

By default, models display to the right of the comparison report, with the model corresponding to the left side of the report on top, and the right below. If you move or resize the models your position settings are respected by subsequent model highlighting operations within the same session. The tool remembers your window positions.

If you want to preserve window positions across sessions, position the window, and then enter:

```
slxmlcomp.storeWindowPositions
```

This preserves the placement of any Simulink windows, Stateflow windows, and truth table windows.

To stop storing window positions and return to the defaults, enter:

```
slxmlcomp.clearWindowPositions
```

View Changes in Model Configuration Parameters

You can use the report to explore differences in the model Configuration Parameters. If you select a Configuration Parameter item, the report displays the appropriate root node pane, if possible, of both Configuration Parameters dialog boxes.

The Parameters pane of the report displays the label text from the dialog controls (or the parameter name if it is command line only), and the parameter values. Some configuration parameters have a different hierarchy in the XML file and the dialog box. You can right-click to merge a selected parameter value in the Parameters pane.

Related Examples

- “Select Simulink Models for XML Comparison” on page 7-5
- “Compare Simulink Model XML Files” on page 7-8
- “Merge Simulink Models from the Comparison Report” on page 7-21
- “Review Changes”

More About

- “About Simulink Model XML Comparison” on page 7-2

Merge Simulink Models from the Comparison Report

In this section...

Merge Models

You can merge Simulink models from the XML text comparison report. You can merge individual parameters, blocks or entire subsystems.

The merge feature enables you to merge two versions of a design modeled in Simulink.

Tip You can only merge from left to right. If you want to merge into the other file, use Swap Sides before you start merging. Swap Sides reverts any merges already made and creates a new comparison report for the original files.

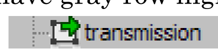
You can merge from the left model to the right model using the XML text files. You can click Undo to revert all merge operations. You can merge modified, added or deleted nodes in the report as follows:

1. Select a report item you want to merge.
2. On the **Comparison** tab, in the **Merge** section, click **Merge Node** to merge the selected node. Merge is disabled when you cannot merge the selected node. For example, you cannot merge the top level model nodes, data nodes, or nodes within configuration settings.

   Tip Merge blocks before lines, and merge states and junctions before merging transitions. See "Merging Tips" on page 7-22.

   ---

3. View the results in the report and the models.

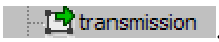
   The report merges the selected node from the left side of the report to the right. Merged report nodes have gray row highlighting, and a green merge arrow if the node has an icon, e.g., 

.

The merge copies the change (a modified, added or deleted item) from the left model to the right model. If the node exists only in the left tree, then the merge inserts it into the right tree. The software attempts to connect all lines to blocks after the merge.

4. To merge individual parameters, right-click an item in the **Parameters** pane and select **Merge Left to Right**. Alternatively, click the **Merge Parameter** button in the **Merge** section of the toolstrip.

   You cannot insert or delete parameters, and not all parameters can be merged.

   If you merge all possible parameters for a node then the report marks that node as merged, e.g., 

. If you partially merge some parameters of a node, the report marks the node as partially merged with a green merge arrow icon and no gray row highlighting.
5. (Optional) To revert all merge operations, on the **Comparison** tab, in the **Merge** section, click **Undo All**. A dialog prompts you to confirm you want to throw away all merge operations and revert the report and models to their original state.
6. Inspect your merge changes in the Simulink Editor. If necessary, connect any lines that the software did not connect automatically. The comparison report does not update to show any changes you make in the Editor.
7. Save the model in the Editor.

Merging Tips

- You must merge blocks before lines in the Simulink part of the report, and in the Stateflow section you must merge states and junctions before merging transitions, or the report cannot make the connections.

  For an example showing how to merge a change involving multiple nodes, see `slxml_sfcar`.
- If you want to merge subsystems, be aware that in XML text files, subsystems are represented by two nodes, the container and the contents. The two nodes have the same name but different properties, for example, name changes are a property of the container node. You can merge the container parameters and contents independently. If you want to merge a subsystem and all its properties, merge both the container and the contents nodes.
- You cannot insert or delete parameters, and not all parameters can be merged. For example, you cannot merge Simulink Identifier (`SID`) parameters.

- If you change filter settings after any merge operations, you will lose your merge changes. A dialog prompts you to confirm you want to throw away all merge operations and revert the report and models to their original state. If you click **Yes** to continue, the Chawathe analysis runs again and you see a new report with the new filtering applied.
- For information on merging between models with identical names, see "Comparing XML Files from Models with Identical Names" on page 7-27.

Merge MATLAB Function Block Code

1 To merge differences in MATLAB Function block code, create a comparison report for the parent models.
2 (Optional) On the View tab, turn off **Highlight in Models**. Otherwise the parent models display each time you merge a difference in the MATLAB Function block code text comparison.
3 Next to the MATLAB Function block node in the XML comparison report, click the **Compare** button .

   A new text comparison report opens.
4 In the text comparison report, select a difference in the code and click **Merge** to copy the selected difference from the left block to the right block.
5 After you finish merging differences, save the parent model in the Editor.

Related Examples

- "Select Simulink Models for XML Comparison" on page 7-5
- "Compare Simulink Model XML Files" on page 7-8
- "Display Items in Original Models" on page 7-17
- "Source Control in Simulink Project"
- "Review Changes"

More About

- "About Simulink Model XML Comparison" on page 7-2
- "Comparing XML Files from Models with Identical Names" on page 7-27

Export, Print, and Save XML Comparison Results

In this section...

Save Printable HTML Report

To save a printable version of an XML comparison report,

1. On the **Comparison** tab, in the **Comparison** section, select **Save As** > **Save as HTML**.

   The Save dialog box opens, where you can choose to save a printable version of the XML comparison report.
2. Select a file name and location to save the report.

The report is a noninteractive HTML document of the differences detected by the Chawathe algorithm for printing, sharing, or archiving a record of the comparison. If you have applied filters, your filtered results appear in the printable report.

Export Results to the Workspace

To export the XML comparison results to the MATLAB base workspace,

1. On the **Comparison** tab, in the **Comparison** section, select **Save As** > **Save to Workspace**.

   The Input Variable Name dialog box appears.
2. Specify a name for the export object in the dialog and click **OK**. This action exports the results of the XML comparison to an `xmlcomp.Edits` object in the workspace.

The `xmlcomp.Edits` object contains information about the XML comparison including file names, filters applied, and hierarchical nodes that differ between the two XML files.

To create an `xmlcomp.Edits` object at the command line without opening the Comparison Tool, enter:

```
Edits = slxmlcomp.compare(modelname_A,modelname_B)
```

Property of `xmlcomp.Edits`	Description
`Filters`	Array of filter structure arrays. Each structure has two fields, Name and Value.
`LeftFileName`	File name of left model exported to XML.
`LeftRoot`	`xmlcomp.Node` object that references the root of the left tree.
`RightFileName`	File name of right model exported to XML.
`RightRoot`	`xmlcomp.Node` object that references the root of the right tree.
`TimeSaved`	Time when results exported to the workspace.
`Version`	MathWorks release-specific version number of `xmlcomp.Edits` object.

Property of `xmlcomp.Node`	Description
`Children`	Array of `xmlcomp.Node` references to child nodes, if any.
`Edited`	Boolean — If `Edited = true` then the node is either inserted (green) or part of a modified matched pair (pink).
`Name`	Name of node.
`Parameters`	Array of parameter structure arrays. Each structure has two fields, Name and Value.
`Parent`	`xmlcomp.Node` reference to parent node, if any.
`Partner`	If matched, `Partner` is an `xmlcomp.Node` reference to the matched partner node in the other tree. Otherwise empty `[]`.

Save Comparison Log Files in a Zip File

Temporary comparison files accumulate in *`tempdir`*`/MatlabComparisons/XMLComparisons/TempDirs/`. These temporary files are deleted when you close the related comparison report.

You can zip the temporary files (such as log files) created during XML text comparisons for sharing or archiving. While the comparison report is open, enter:

```
xmlcomp.zipTempFiles('c:\work\myexportfolder')
```

The destination folder must exist. The output reports the zip file name:

```
Created the zipfile "c:\work\myexportfolder\20080915T065514w.zip"
```

To view the log file for the last comparison in the MATLAB Editor, enter:

```
xmlcomp.showLogFile
```

Comparing XML Files from Models with Identical Names

You can compare XML text from files of the same name. To complete the operation, the XML comparison tool copies one of the models to a temporary folder, because Simulink cannot have two models of the same name in memory at the same time. The XML comparison tool creates a read-only copy of one model named `modelname_TEMPORARY_COPY`, and compares the resulting XML files.

Warning When you use reverse annotation from the report, one of the models displayed is a temporary copy with a new name. The temporary copy is read-only, to avoid making changes that can be lost.

Alternatively, you can run the comparison by renaming or copying one of the files.

All *merge* operations merge from left to right, so you cannot accidentally merge to a temporary copy. Merge operations on models with identical names copy changes from the left (temporary copy) model to the right model. If you swap sides, the report always places a new temporary copy on the left side of the report, so any merges change the original model file and never a temporary copy.

If one of the models is open when you try to compare XML files, a dialog box appears where you can click **Yes** to close the file and proceed, or **No** to abort. You must close open models before the XML comparison tool can compare XML files from two models with the same name. The problem requiring you to close the loaded model is called “shadowed files”. In some cases, another model with the same name might be in memory, but not visible. See “Shadowed Files ” in the Simulink documentation for more information.

If you want to automatically close open models of the same name when comparing XML files and not see the dialog box again, run these commands:

```
opt = slxmlcomp.options
opt.setCloseSameNameModel(true)
```

This is persistent across MATLAB sessions. To revert to default behavior and be prompted whether or not to close the open model every time, enter:

```
opt = slxmlcomp.options
opt.setCloseSameNameModel(false)
```

Work with Referenced Models and Library Links

The XML comparison report applies only to the currently selected models, and does not include changes to any referenced models or linked libraries. For compatibility with source control and peer review workflows, the comparison report shows only changes in the files selected for comparison.

Tip If you want to examine your whole hierarchy instead, try using a Simulink Project, where you can examine modified files and dependencies across your whole project, and compare to selected revisions. See "Project Management ".

If you are creating an XML comparison report for models that contain referenced models, and you have more than one referenced model with the same name, then your MATLAB path can affect the results. For example, this can happen if you generate an XML comparison report for the current version of your model and a previous baseline. To avoid seeing unexpected changes in model reference blocks, make sure that your referenced models are not on your MATLAB path before you generate the report.

The reason why results can change is that Simulink records information in the top model about the interface between the top model and the child model. This interface information in the top model enables incremental loading and diagnostic checks without any need to load child models.

When you load a model (for example, to compare XML) then Simulink refreshes the interface information for referenced models if it can find the child model. Simulink can locate the child model if it is on the path. If another model of the same name is higher on the path, Simulink updates the interface information for that other model before comparing XML. This can produce entries for model reference blocks in the comparison report that you did not expect. If you make sure your referenced models are not on your path before you generate the report, then you can avoid these unexpected results. If both model versions are off the path, the interface information in the top model is not refreshed during the XML comparison process. Instead the cached information is used, resulting in a valid XML comparison report.

With library links, Simulink does not update the cached interface information when comparing XML, and so the report correctly captures library interfaces. However with both referenced models and library links, Simulink updates the information when displaying the model. When displaying report items in original models, you may see unexpected results because Simulink may find a model or library that is higher in the path. To obtain the clearest results, make sure that the models and associated libraries are temporarily removed from the path. By removing the files from the path you will see unresolved library links and referenced models when you view the original models, but their interfaces will be correct and will correctly align with the comparison report.

Compare XML from Models Managed with Subversion

In this section...

Work with Subversion

Simulink Projects provide built-in Subversion source control integration. You can create an XML comparison report from the Modified Files view of the Simulink Project Tool. See “Project Management ”.

Alternatively, you can customize your external Configuration Management tools to call the XML comparison functionality in the Simulink Report Generator, as described on this page.

Comparing two versions of the same file is a common workflow when using Configuration Management tools. If your Configuration Management tool is configurable, you can customize your *Diff* operations on Simulink model files from your Configuration Management tool to call the XML comparison functionality in the Simulink Report Generator. This allows you to compare two versions of the XML from the same model file and generate a report of the differences.

TortoiseSVN and Subversion are a popular suite of open-source version control tools. The following example describes how to configure TortoiseSVN to use the Simulink Report Generator XML comparison. You can register the XML comparison function with TortoiseSVN as an extension-specific diff program to use for model files. When you perform a TortoiseSVN diff on a model file, TortoiseSVN uses the XML comparison to generate a report. This workflow describes a typical usage of Subversion on a Windows PC.

1. Configure TortoiseSVN to use the `fileComparisonDriver` function for model files.
2. When you perform a TortoiseSVN diff on a model file, the `fileComparisonDriver` function invokes the `visdiff` function to generate an XML comparison report.

Optionally, you can also configure TortoiseSVN to use the same function to call the Comparison Tool for `.mat` files and for Simulink manifest files (`.smf` files).

Configure TortoiseSVN

This example is compatible with Release 2008b+ onwards and was tested with Subversion 1.7.7 on Windows 7 Enterprise.

Configure TortoiseSVN to use the XML comparison tool for model files, as follows:

1. Right-click a file in Windows Explorer and select **TortoiseSVN** > **Settings**.
2. In the TortoiseSVN Settings dialog box, click **Diff Viewer** under **External Programs** in the tree, then click the **Advanced** button.
3. In the Advanced Diff settings dialog box, add an entry to specify what to do with model files.

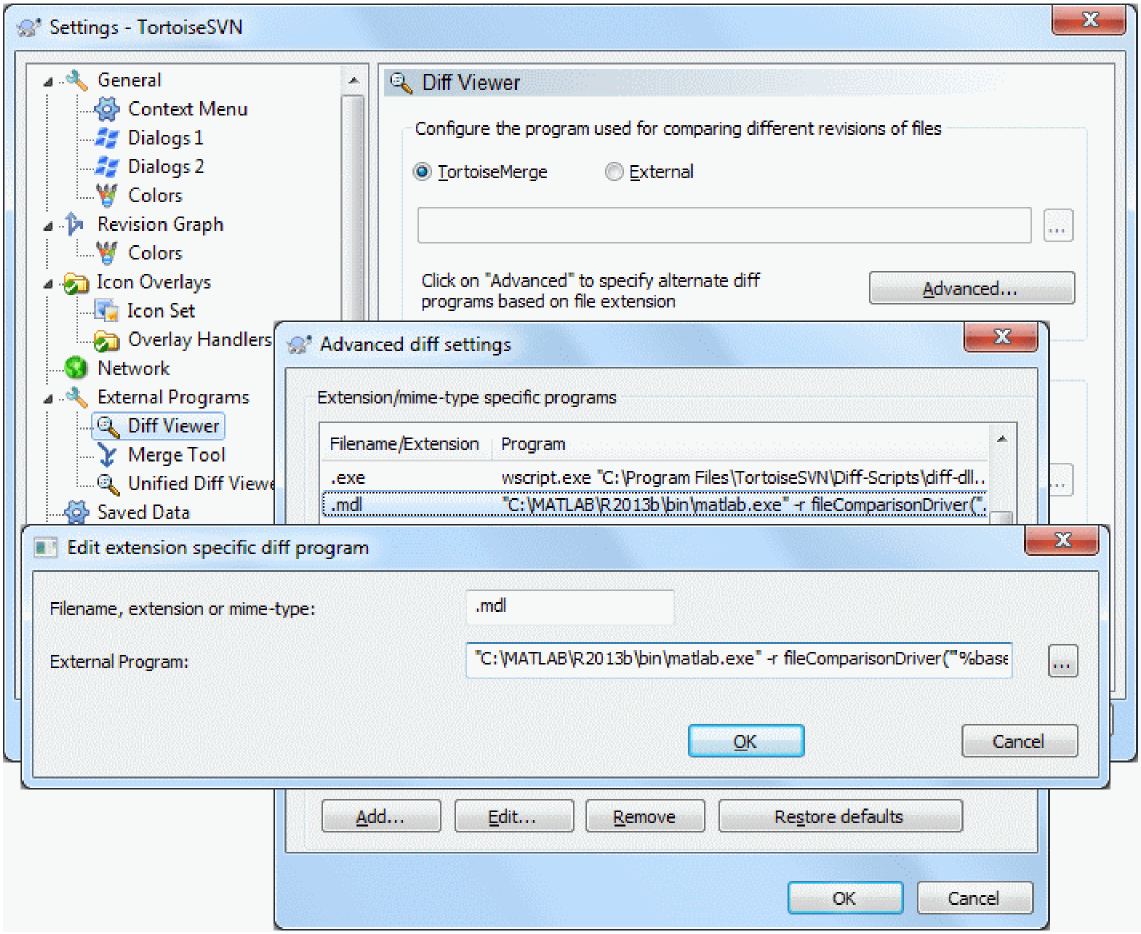
   a. Click **Add**.

   b. In the Add extension specific diff program dialog box, enter `.mdl` or `.slx` for the **Extension** and enter the following command in the **External Program** edit box:

   ```
   "matlabroot\bin\matlab.exe" -r fileComparisonDriver("'%base'","'%mine'") -nospl
   ```

   Replace *`matlabroot`* with the path to the specific location on your computer for your MATLAB installation, for example, `C:\Program files\MATLAB\R2009a`.

   The following example shows this setup on an R2013b installation.



4 Click **OK** to apply your changes and close all the Tortoise SVN dialog boxes.

5 If you also want to use the MATLAB Comparison Tool for `.mat` files and for Simulink manifest files (`.smf` files) you can repeat the steps to add exactly the same **External Program** command for `.mat` files and `.smf` files.

Test TortoiseSVN Setup

To test your setup, follow these steps:

1 Start MATLAB and open, modify, and save a Simulink model that is managed in a Subversion archive. This creates a local working copy that is different to the head repository copy.

2 In Windows Explorer, right-click your modified file, and select **TortoiseSVN** > **Diff**.

TortoiseSVN runs a new instance of MATLAB. MATLAB loads and runs the `fileComparisonDriver.m` file, located in the folder `matlab\toolbox\rptgenext\rptgenextdemos\slxmlcomp`. The `fileComparisonDriver` function performs these steps:

1 Creates temporary copies of the current working version of the Simulink model and the previously stored version of the model.

2 Compares the XML text files from both models and generates a comparison report displayed in the MATLAB Comparison Tool.

The function must preprocess the file names by creating renamed temporary copies because Subversion uses a temporary file naming convention that is not compatible with Simulink because of invalid delimiting characters. The branch and version information is embedded in the temporary model names. See also “Comparing XML Files from Models with Identical Names” on page 7-27 for information about using the report and a warning to avoid losing work in the temporary models.

Other TortoiseSVN workflows using Diff operations are also supported, such as comparing two versions in an archive.

Components — Alphabetical List

For a list of MATLAB Report Generator components, see the MATLAB Report Generator documentation.

Acronym Definitions

Create table of Polyspace acronyms used in report and their full forms

Description

This component creates a table containing the acronyms used in the report and their full forms. Aronyms are used for Polyspace® Code Prover™ checks and Polyspace result status.

Annotation Loop

Run child components multiple times for each Simulink annotation in current context

Description

This component runs its child components multiple times for each Simulink annotation in the current context. The parent component determines the context.

- `Model Loop`: Reports on all annotations inside the reported portion of the reported model.
- `System Loop`: Reports on all annotations inside the current system.
- `Block Loop` or `Signal Loop`: Does nothing.

Loop Options

The Loop Options pane displays information about the current context. You can sort `Alphabetically by text` or `In traversal order`.

Child components of the `Annotation Loop` consider their context to be annotations when the report is running.

For example, the following components report on the looped annotation:

- `Simulink Automatic Table`
- `Simulink Linking Anchor`
- `Simulink Name`
- `Simulink Property`
- `Simulink Property Table`

Use a Summary Table component to show annotation objects in reports. Each Summary Table component creates a single table with each reported annotation on a single row of the table.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to each object in the loop, the generated report.

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`, `Simulink Linking Anchor`, `Simulink Name`, `Simulink Property`, `Simulink Property Table`, `Simulink Summary Table`

Block Execution Order List

Create a list or table of all nonvirtual blocks in the model, showing order in which they execute

Description

This component creates a list or table of all nonvirtual blocks in the model, showing the order in which they execute.

For more information about virtual and nonvirtual blocks, see “About Blocks” in the Simulink documentation.

Properties

- **List Title**:
  - `Automatic`: Generates a list or table title automatically.
  - `Custom`: Enables you to enter a title.
- **Include block type information**: Include each block's `BlockType` property in the list or table.

- **Look under nonvirtual subsystems**: The default is `Automatic (On for models, Off for systems)`. Set it to `On` or `Off`.

Insert Anything into Report?

Yes. List.

Class

`rptgen_sl.csl_blk_sort_list`

See Also

`Block Loop`

Block Loop

Run child components for each block in the current system, model, or signal

Description

This component runs its child components for each block contained in the current system, model, or signal.

For conditional processing based of blocks, you can use the `RptgenSL.getReportedBlock` function. For more information, see "Loop Context Functions".

Report On

This pane describes the type of object on which this component operates.

- **Automatic list from context**: Report on all blocks in the current context. The parent component of the Block Loop determines its context. If this component does not have the `Model Loop`, `System Loop`, `Signal Loop`, or `Block Loop` as its parent, selecting this option causes this component to report on all blocks in all models.
  - `Model Loop`: Reports on all blocks in the current model.
  - `System Loop`: Reports on all blocks in the current system.
  - `Signal Loop`: Reports on all blocks connected to the current signal.
- **Custom - use block list**: Enables you to specify a list of blocks on which to report. Enter the full path of each block.

Loop Options

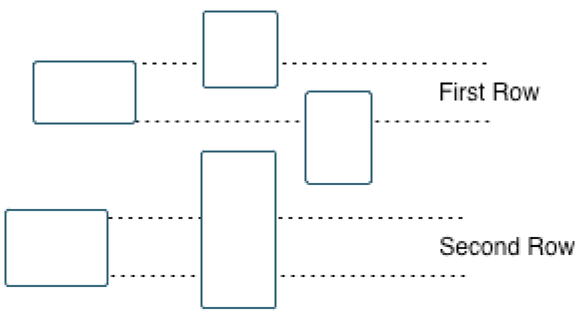
Choose block sorting options and reporting options in this pane.

- **Sort blocks**:

  Use this option to select how to sort blocks (applied to each level in a model):

  - `Alphabetically by block name`. Sorts blocks alphabetically by their names.

- `Alphabetically by system name`. Sorts systems alphabetically. The report lists blocks in each system, but in no particular order.
- `Alphabetically by full Simulink path`. Sorts blocks alphabetically by Simulink path.
- `By block type`. Sorts blocks alphabetically by block type.
- `By block depth`. Sorts blocks by their depth in the model.
- `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



- `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
- `By traversal order`. Sorts blocks by traversal order.
- `By simulation order`. Sorts blocks by execution order.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. The `%<>` notation can denote a string or cell array. The following example reports on the `theta dot` integrator block and the `theta` integrator block in the model `simppend`, using the variable `Z={ 'simppend/theta'}`:

```
simppend/theta dot
%<Z>
```

The generated report includes information about the following blocks:

- `simppend/theta dot`
- `simppend/theta`

For more information, see `%<VariableName> Notation` on the `Text` component reference page in the MATLAB Report Generator documentation.

- **Search for Simulink property name/property value pairs**: Reports only on Simulink blocks with specified property name/property value pairs.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each block found in the loop.
- **Display the object type in the section title**: Automatically inserts the object type into the section title in the generated report.
- **Create link anchor for each object in loop**: Create a hyperlink to the block in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sl.csl_blk_loop`

See Also

`Model Loop`, `Signal Loop`, `System Loop`, `Simulink Linking Anchor`, `Simulink Name`, `Simulink Property`, `Simulink Property Table`, `Simulink Summary Table`

Block Type Count

Count number of each block type in the current model or system

Description

This component counts the number of each block type in the current model or system. Within a model, this component counts blocks underneath masks and inside library links.

For more information about block types, see "About Blocks" in the Simulink documentation.

Count Types

The parent of this component determines where to count block types:

- `Model Loop`: Reports all block types in the current model:
  - `All blocks in model`: Counts block types in the entire model.
  - `All blocks in reported systems`: Counts block types only in systems that appear in the report.
- `System Loop`: Reports all block types in the current system.

Table Content

- **Table title**: Allows you to enter the table title.
- **Show block names in table**: Includes a column that displays all block names in the table.
- **Sort table**:
  - `Alphabetically by block type`: Sorts blocks alphabetically by block type.
  - `By number of blocks`: Sorts by decreasing number of occurrences.
- **Show total count**: Displays total number of block types.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_blk_count`

See Also

`Block Loop`, `Model Loop`, `System Loop`

Bus

Create list of signals exiting from `Bus Selector` block

Description

This component creates a list of signals exiting a `Bus Selector` block. The list contains signals leaving from the reported block or downstream buses and signals.

The parent of this component determines which buses appear in the report:

- `Model Loop`: Includes all buses in the current model.
- `System Loop`: Includes all buses in the current system.
- `Block Loop` : If the current block is a bus block, then the report includes that block.
- `Signal Loop`: Includes all buses connected to the current signal.

If the Bus component does not have a looping component as its parent, it reports on all buses in all open models.

Properties

- **Show Bus Hierarchy**: Specifies whether the list displays downstream buses hierarchically.
- **Insert linking anchor for bus blocks**: Inserts a linking anchor for each bus block. This property designates the list item as the location to which other links for that block point. (For more information, see the `Simulink Linking Anchor` or `Link` component reference pages.) Do not use this option if you have already specified an anchor location for the bus block with an `Object Linking Anchor` component.
- **Insert linking anchor for signals**: Inserts a linking anchor for each signal. This property designates the list item as the location to which other links for that signal point. For more information, see the `Simulink Linking Anchor` or `Link` component reference pages.) Do not use this option if you have already specified an anchor location for the signal with an `Object Linking Anchor` component.
- **Title**: Inserts a title before each list. This attribute supports the `%<varname>` notation. For more information, see `%<VariableName> Notation` on the `Text` component reference page in the MATLAB Report Generator documentation.

Insert Anything into Report?

Yes. List.

Class

`rptgen_sl.csl_blk_bus`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `Simulink Linking Anchor`, `System Loop`,

Chart Loop

Run child components for specified Stateflow charts

Description

This component runs its children for specified Stateflow charts.

For conditional processing for a chart, you can use the `RptgenSF.getReportedChart` function. For more information, see “Loop Context Functions”.

Report On

- **Automatic list from context**: Report on all chart blocks in the context set by the parent of this component.
  - `Model Loop`: Reports on all Stateflow chart blocks in the current model.
  - `System Loop`: Reports on all Stateflow chart blocks in the current system.
  - `Signal Loop`: Reports on all Stateflow chart blocks connected to the current signal.
  - `Machine Loop`: Reports on the current block if it is in a Stateflow chart.

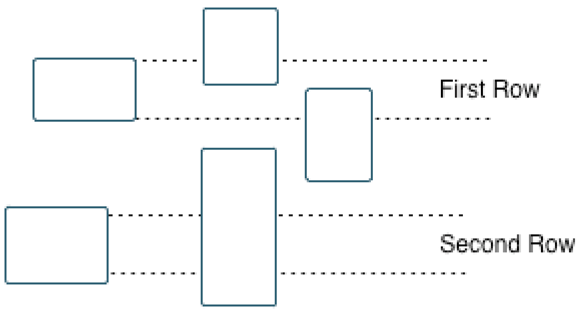
  If the `Chart Loop` component has any other type of component as its parent, selecting this option causes it to report on all Stateflow chart blocks.
- **Custom - use block list**: Reports on a specified list of Stateflow chart blocks.

Loop Options

Choose chart block sorting options and reporting options in this pane.

- **Sort blocks**: Specifies how to sort blocks (applied to each level in a model). This option is available if you select `Automatic list from context` in the **Report On** section, or if you select `Custom - use block list` and the `Sort blocks` option.
  - `Alphabetically by block name`. Sorts blocks alphabetically by name.
  - `Alphabetically by system name`. Sorts systems alphabetically by name. Lists blocks in each system, but in no particular order.

- `Alphabetically by full Simulink path`. Sorts models alphabetically by their full paths.
- `By block type`. Sorts blocks alphabetically by block type.
- `By depth`. Sorts blocks by their depth in the model.
- `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



- `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
- `By simulation order`. Sorts blocks by execution order.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. The `%<>` notation can denote a string or cell array. For more information, see `%<VariableName> Notation` on the `Text` component reference page in the MATLAB Report Generator documentation.

- **Search for Simulink property name/property value pairs**: Reports on Simulink blocks with specified property name/property value pairs.
- **Search Stateflow**: Reports on Stateflow charts with specified property name/ property value pairs.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sf.csf_chart_loop`

See Also

`Block Loop`, `Machine Loop`, `Model Loop`, `Signal Loop`, `System Loop`, `Simulink Function System Loop`

Code Generation Summary

Insert version number information, list of generated files, tables summarizing code generation configuration information, and subsystem maps into report

Description

This component reports the following information:

- Version number information
- List of generated files
- Code generation configuration information
- Subsystem map

Summary

- **General information**: Includes the following information in the report:
  - Model name and version
  - Simulink Coder version number
  - List of full paths of generated files
- **Configuration settings**: Includes tables that list optimization and Simulink Coder target selection and build process Configuration Parameter settings.
- **Subsystem map**: Includes in the report a unique mapping between subsystem numbers and subsystem labels in the model.

Traceability Report

Use settings from model: When you select this option, the report uses all of the following configuration settings, as specified in your model. Deselecting this option allows you to turn off one or more of these settings as needed:

- **Eliminated/virtual blocks**
- **Traceable blocks**

- **Traceable StateFlow Objects**
- **Traceable MATLAB Function Blocks**

For more information on these configuration settings, see "Code Generation Pane: Report" in the Simulink Coder documentation.

Insert Anything into Report?

Yes. Tables and list.

Class

`RptgenRTW.CCodeGenSummary`

See Also

`Import Generated Code`

Code and Verification Information

Create table of verification times and code characteristics

Description

This component creates tables containing verification times and code characteristics such as number of lines.

Properties

Include Verification Time Information

If you select this option, the report contains verification times broken down by phase.

- For Polyspace Bug Finder™, the phases are `compilation`, `pass0`, `pass1`, etc.
- For Polyspace Code Prover, the phases are `compilation`, `global`, `function`, etc.

Include Code Details

If you select this option, the report contains the following code characteristics:

- Number of files
- Number of lines
- Number of lines without comment

Code Metrics Details

Create table of Polyspace metrics broken down by file and function

Description

This component creates a table containing metrics from a Polyspace project. If you select the appropriate properties, the metrics appear broken down by file and function.

Properties

Project Metrics

If you select this option, the report contains the following metrics about the project:

- Number of direct recursions
- Number of files
- Number of headers
- Number of protected and unprotected shared variables

File Metrics

If you select this option, the report contains the following metrics about each file in the project:

- Estimated function coupling
- Lines without comment
- Comment density
- Total lines

Function Metrics

If you select this option, the report contains the following metrics about each function in the project:

- Number of lines within body
- Number of executable lines
- Number of `goto` statements
- Number of call levels
- Number of called functions
- Number of call occurrences
- Number of function parameters
- Number of paths
- Number of `return` statements
- Number of instructions
- Cyclomatic complexity
- Language scope
- Number of calling functions

Code Metrics Summary

Create table of Polyspace metrics

Description

This component creates a table containing metrics from a Polyspace project. The metrics are the same as those displayed under "Code Metrics Details". However, the file and function metrics are not broken down by individual files and functions. Instead, the table provides the minimum and maximum value of a file metric over all files and a function metric over all functions.

Code Verification Summary

Create table of Polyspace analysis results

Description

This component creates tables containing the following results:

- Number of results
- Number of coding rule violations for each coding rule type such as MISRA C®
- Number of defects, for Polyspace Bug Finder results
- Number of checks of each color, for Polyspace Code Prover results
- Whether the project passed or failed the software quality objective

Properties

Include Checks from Polyspace Standard Library Stub Functions

Unless you deselect this option, the tables contain Polyspace Code Prover checks that appear in Polyspace stubs for the standard library functions.

Coding Rules Details

Create table of coding rule violations broken down by file

Description

This component creates tables containing coding rule violations broken down by each file in the Polyspace project. For each rule violation, the table contains the following information:

- Rule number
- Rule description
- Function containing the violation
- Line and column number
- Review information such as classification, status and comments

Properties

Select Coding Rules Type

Using this option, you can choose which coding rule violations to display. You can display violations for the following set of coding rules:

- MISRA C rules
- MISRA® AC AGC rules
- MISRA C++ rules
- JSF® C++ rules
- Custom coding rules

Display by

Using this option, you can break down the display of coding rule violations by file.

Coding Rules Summary

Create table with number of coding rule violations

Description

This component creates a table containing the number of coding rule violations. You can choose whether to break this information down by rule number or file.

Properties

Select Coding Rules Type

Using this option, you can choose which coding rule violations to display. You can display violations for the following set of coding rules:

- MISRA C rules
- MISRA AC AGC rules
- MISRA C++ rules
- JSF C++ rules
- Custom coding rules

Include Files/Rules with No Problems Detected

If you select this option, the table displays:

- Files containing no coding rule violations
- Rules that your code does not violate

Display by

Using this option, you can break down the display of coding rule violations by:

- Rule number

- File

Configuration Parameters

Create table of analysis options and coding rules

Description

This component creates two tables:

- Polyspace Settings: The analysis options that you used to obtain your results. The table lists command-line version of the options along with their values.
- Coding Rules Configuration: The coding rules whose violations you checked for. The table lists the rule number, rule description and other information about the rules.

Defects Summary

Create table of Polyspace Bug Finder defects

Description

This component creates a table of Polyspace Bug Finder defects. From this table, you can see the number of defects of each type.

Properties

Include Checkers with No Defects Detected

If you select this option, the table includes all defects that Polyspace Bug Finder detects, including those that do not occur in your code.

Documentation

Insert text extracted from `DocBlock` blocks in Simulink models

Description

This component inserts text extracted from `DocBlock` blocks in Simulink models. It can have the following components as its parent:

- `Model Loop`
- `System Loop`
- `Block Loop`

The specified report format determines the format of the `DocBlock` block data inserted into the report:

- `HTML`: Imports HTML data into the report.

  Note: For non-English HTML DocBlock text that you want to include in a Documentation component, use UTF-8 file encoding. Use a simple text editor to create the HTML code.

- `RTF`: Imports RTF data into the report.

Properties

- **Import file as**: Specifies how to format the imported information. The following example shows how each option works, using the following text as input:

  ```
  First row.
    Second row.

  Third row follows blank line.
  ```

  - `Plain text (ignore line breaks)`. Imports plain text without any line breaks (no paragraphs), as in this example:

    ```
    First row. Second row. Third row follows blank line.
    ```

- `Paragraphs defined by line breaks`. Imports the text contained in paragraphs defined by line breaks (hard returns or carriage returns), as in this example:

  ```
  First row.
  Second row.

  Third row follows blank line.
  ```

- `Paragraphs defined by empty rows`. Imports text contained in paragraphs defined by empty rows (rows that do not contain text), as in this example:

  ```
  First row. Second row.

  Third row follows blank line.
  ```

- `Text (retain line breaks)`. Imports plain text, including line breaks, as in this example:

  ```
  First row.
  Second row.

  Third row follows blank line.
  ```

- `Fixed-width text (retain line breaks)`. Imports fixed-width text (all letters have the same width or size) including line breaks, as in this example:

  ```
  First row.
    Second row.

  Third row follows blank line.
  ```

  Tip This option is useful for importing MATLAB files.

- **Insert linking anchor for blocks**: Inserts a linking anchor for each `DocBlock` block that designates the location where other links for that block point. (See the `Simulink Linking Anchor` or `Link` component reference pages for more help.) Do not use this option if you have already specified an anchor location for a `DocBlock` block with an `Object Linking Anchor` component.

Insert Anything into Report?

Yes. Text, paragraph, or external RTF/HTML data.

Class

`rptgen_sl.csl_blk_doc`

See Also

`Block Loop`, `Model Loop`, `Simulink Linking Anchor`, `System Loop`

Fixed Point Block Loop

Run child components for the Simulink model, system, or signal defined by parent component

Description

This component runs its children for the Simulink model, system, or signal that its parent defines. Options for the parent component are:

- `Model Loop`
- `System Loop`
- `Signal Loop`

Report On

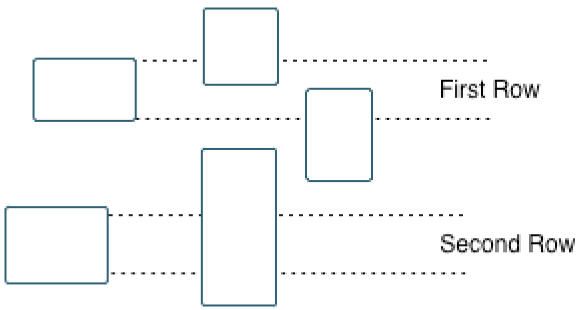
- **Automatic list from context**: Reports on all fixed-point blocks in the context of the parent of this component. For example, if the parent component is the `System Loop`, then this component reports on all fixed-point blocks in the current system. If this component does not have a looping component as its parent, then selecting this option causes the component to report on all fixed-point blocks in all models.
- **Custom - use block list:**: Reports on a specified list of blocks.

Loop Options

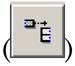
Choose block sorting options and reporting options in this pane.

- **Sort blocks**: Specifies how to sort blocks (applied to each level in a model). This option is available if you select the `Automatic list from context` option in the **Report On** section, or if you select `Custom - use block list` and the `Sort blocks` options.
  - `Alphabetically by block name`. Sorts blocks alphabetically by name.
  - `Alphabetically by system name`. Sorts systems alphabetically. Lists blocks in each system, but in no particular order.

- `Alphabetically by full Simulink path`. Sorts blocks alphabetically by Simulink path.
- `By block type`. Sorts blocks alphabetically by block type.
- `By block depth`. Sorts blocks by their depth in the model.
- `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



- `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
- `By traversal order`. Sorts blocks by traversal order.
- `By simulation order`. Sorts blocks by execution order.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. For more information, see `%<VariableName> Notation` on the `Text` component reference page in the MATLAB Report Generator documentation.

- **Search for Simulink name/property value pairs**: Reports only on blocks with the specified property name/property value pairs. To enable searching, click the check box. In the first row of the property name and property value table, click inside the edit box, delete the existing text, and type the property name and value. To add a row, use the **Add row** button ().

For information about subsystem property names and values, in "Block-Specific Parameters", see the "Ports & Subsystems Library Block Parameters" section.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_fp.cfp_blk_loop`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `Simulink Linking Anchor`, `System Loop`

Fixed Point Logging Options

Set fixed-point options like in Fixed Point Tool

Description

This component sets fixed-point options like those set in the Fixed Point Tool (invoked by running the `fxptdlg` function).

This component must be a child of the `Model Loop` component. Use this component to set the following options on the current model:

- Data type override
- Fixed-point instrument mode
- Logging type

This component can have child components. It is a good practice to use this component with a `Model Simulation` component as its child. This approach sets fixed-point properties for the model for the purpose of the simulation, and then restores them to their original values after the simulation is complete.

Data Type Override

- **Use local settings**: Overrides data types according to the value of this parameter set for each subsystem. Otherwise, settings for parent systems override those of child systems.
- **Scaled double**: Overrides the output data type of all blocks in the current system or subsystem with doubles. However, this option maintains the scaling and bias specified in the mask of each block.
- **Doubles**: Overrides the output data type of all blocks in the current system or subsystem with doubles. The overridden values have no scaling or bias.
- **Singles**: Overrides the output data type of all blocks in the current system or subsystem with singles. The overridden values have no scaling or bias.
- **Off**: Does not perform any data type override on any block in the current system or subsystem.

Fixed-Point Instrumentation Mode

Specify logging options in this section. For logged blocks, minimum and maximum simulation values are written to the workspace.

- **Use local settings**: Logs data according to the value of this parameter set for each subsystem. Otherwise, settings for parent systems always override those of child systems.
- **Min, max, and overflow**: Logs minimum value, maximum value, and overflow data for all blocks in the current system or subsystem.
- **Overflow only**: Logs only overflow data for all blocks in the current system or subsystem.
- **Force off**: Logs no data for any block in the current system or subsystem. Use this selection to work with models containing fixed point-enabled blocks, if you do not have a Fixed-Point Designer license.

For more information on logging simulation results, see “Propose Fraction Lengths Using Simulation Range Data” in the Fixed-Point Designer documentation.

Logging Type

Specify how to record logs in this section:

- **Overwrite log**: Clears information in the logs before new logging data is entered.
- **Merge log**: Merges new logging data with previously logged information.

Insert Anything into Report?

No.

Class

`rptgen_fp.cfp_options`

See Also

`Model Simulation`

Fixed Point Property Table

Insert table that reports on Fixed-Point Designer block property name/property value pairs

Description

This component inserts a table that reports on Fixed-Point Designer block property name/property value pairs.

Table

Select a preset table, which is already formatted and configured, in the **preset table** list in the upper-left corner of the attributes page.

- **preset table**

  Specifies the type of object property table.

  - `Default`
  - `Mask properties`
  - `Block limits`
  - `Out-of-range errors`
  - `All fixed-point properties`
  - `Blank 4x4`

  To apply the specified table, select the table and click **Apply**.
- **Split property/value cells**: Split property name/property value pairs into separate cells. For the property name and property value to appear in adjacent horizontal cells in the table, select the **Split property/value cells** check box. In this case, the table is in split mode, so there only one property name/property value pair can exist in a cell. If there is more than one name/property pair in a cell, only the first pair appears in the report. The report ignores all subsequent pairs.

  For the property name and property value to appear together in one cell, clear the **Split property/value cells** check box. That option specifies nonsplit mode. Nonsplit mode supports more than one property name/property value pair and text.

Before switching from nonsplit mode to split mode, make sure that there is only one property name/property value pair per table cell. If you have more than one property name/property value pair or text in one cell, only the first value pair appears in the report. Subsequent pairs and text are omitted.

- **Display outer border**: Display the outer border of the table in the generated report.

Table Cells

Select table properties to modify. The selection in this pane affects the available fields in the **Cell Properties** pane.

Cell Properties

- **Contents**

  Modify the contents of the table cell selected in the **Table Cells** pane.
- **Show as**: Specifies the format for the contents of the table cell.
  - `PROPERTY Value`
  - `Value`
  - `Property Value`
  - `Property: Value`
  - `PROPERTY: Value`
  - `Property - Value`
  - `PROPERTY - Value`
- **Alignment**: Aligns the contents of the table cell.
  - `Center`
  - `Left`
  - `Right`
  - `Double justified`
- **Lower border**: Displays the lower border of the table in the generated report.
- **Right border**: Displays the right border of the table in the generated report.

Creating Custom Tables

To create a custom table, edit a preset table, such as the `Blank 4x4` table. Add and delete rows and add properties. To open the Edit Table dialog box, click **Edit**.

For details about creating custom property tables, see “Property Table Components”.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_fp.cfp_prop_table`

See Also

`Fixed Point Summary Table`

Fixed Point Summary Table

Table of specified fixed-point block properties or parameters

Description

This component displays properties or parameters of specified fixed-point blocks in a table.

Properties

Table title

Choose a table title in the generated report:

- `Automatic`: Generates a title automatically from the parameter.
- `Custom`: Specifies a custom title.

Property Columns

Property name

This field displays the object properties to include in the Summary Table in the generated report.

- To add a property:
  1. Select the appropriate property level in the menu
  2. Select the property to add from the selection list and click **Add**.
- To delete a property, select the property name and click the **Delete** button.
- To move properties up and down in the list, click the **Up** and **Down** buttons.

Note: Some entries in the list of available properties (such as `Depth`) are "virtual" properties that you cannot access using the `get_param` command. The properties used for property/value filtering in the block and system loop components must be retrievable by the `get_param`. Therefore, you cannot configure your Summary Table to report on all blocks of `Depth` == 2.

Transpose table

Enabling this check box changes the summary table rows into columns in the generated report, putting the property names in the first column and the values in the other columns.

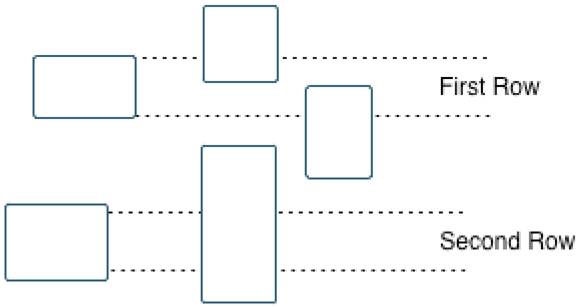
Object Rows

- **Insert anchor for each row**: Inserts an anchor for each row in the summary table.
- **Report On**: Specifies blocks on which to report:
  - `Automatic list from context`. Reports on all blocks in the current context.
  - `Custom - use block list`. Reports on a specified list of blocks. To include a given block in the report, specify its full path.

Loop Options

- **Sort blocks**: Specifies how to sort blocks (applied to each level in a model):
  - `Alphabetically by block name`. Sorts blocks alphabetically by name.
  - `Alphabetically by system name`. Sorts systems alphabetically. Lists blocks in each system, but in no particular order.
  - `Alphabetically by full Simulink path`. Sorts blocks alphabetically by Simulink path.
  - `By block type`. Sorts blocks alphabetically by block type.
  - `By block depth`. Sorts blocks by their depth in the model.
  - `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The

other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



- `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
- `By traversal order`. Sorts blocks by traversal order.
- `By simulation order`. Sorts blocks by execution order.

- **Search for Simulink property name/property value pairs**: Reports only on Simulink blocks with specified property name/property value pairs.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_fp.cfp_summ_table`

See Also

`Fixed Point Property Table`

Import Generated Code

Import source and header files generated by Simulink Coder software, and custom files specified as part of model

Description

This component imports source and header files generated by Simulink Coder software. It also imports custom files that you specify as part of your model.

Properties

- **Source files (auto-generated)**: Includes the following files in the report:
  - `.c` and `.cpp` source files generated by Simulink Coder software.
  - Simulink Coder source files, such as the setup file and supporting files in the build folder.

  This check box is selected by default. Clear it to omit source files.
- **Header files (auto-generated)**: Includes the following files in the report:
  - `.h` and `.hpp` header files generated by Simulink Coder software.
  - Simulink Coder header files in the build folder.

  This check box is selected by default. Clear it to omit source files.
- **Custom files**: Includes custom source files that you specify in the **Code Generation** > **Custom Code** pane of the Configuration Parameters dialog box. This check box is deselected by default.

Insert Anything into Report?

Yes. Generated code listings.

Class

`RptgenRTW.CImportCode`

See Also

`Code Generation Summary`

Look-Up Table

Report on lookup table blocks

Description

The Look-Up Table component reports on the following blocks in the Simulink Lookup Tables library. Some examples of the lookup table blocks include:

- 1-D Lookup Table
- n-D Lookup Table
- Cosine
- Interpolation Using Prelookup
- Direct Lookup Table (n-D)

The Look-Up Table component inserts a figure and/or table into the report. The table contains input and output numeric values. A figure plots these values.

Note: The Look-Up Table component does not display a table or plot for the Direct Lookup Table (n-D) block if the block is configured to generate the table during simulation as a block input. Instead, the Look-Up Table displays a note in the report to the effect that the table is generated dynamically during simulation.

Look-Up Table Options

This pane allows you to specify the types of lookup table blocks to include in the report and how they appear. If you select none of the check boxes in this pane, the component does not insert anything into the report.

- The Look-Up Table displays results according to the type of its parent component:
  - `Model Loop`: Includes all lookup tables in the current model.
  - `System Loop`: Includes all lookup tables in the current system.
  - `Block Loop`: If the current block is a lookup table, the reports that block.
  - `Signal Loop`: Includes all lookup tables connected to the current signal.

  - If the Look-Up Table does not have any of the looping components as its parent, it includes all lookup tables in all open models.
- **Plot 1-D data**: Plots data from a `1-D Lookup Table` block. Choose the plot type, `Line plot` or `Bar plot`, from the corresponding list. The input data appears on the horizontal or *x*-axis, and the output data appears on the vertical or *y*-axis.

  For more information on line and bar plots, see "2-D and 3-D Plots" in the MATLAB Graphics documentation.
- **Create table for 1-D data**: Creates a table that contains numeric data values from the `1-D Lookup Table` block.
- **Plot 2-D data**: Creates a plot of `2-D Lookup Table` blocks. You can specify whether the data appears as a surface plot or a line plot. The line plot is best for small data sets, and the surface plot for larger tables. For more information on surface and line plots, see "2-D and 3-D Plots" in the MATLAB Graphics documentation.

  Note: This option creates a 2-D slice through n-D data.
- **Create table for 2-D data**: Creates a table that contains numeric data values from the `2-D Lookup Table` block.
- **Create table for N-D data**: Creates a table that contains numeric data values from the `n-D Lookup Table` block.

Print Options

- **Image file format**: Specifies the image file format. Select `Automatic HG Format` (the default) to choose automatically the format best suited for the output format that you chose in the Report component. Otherwise, choose an image format that your output viewer can read.
  - `Automatic SL Format` (Uses the Simulink file format selected in the Preferences dialog box)
  - `Bitmap (16m-color)`
  - `Bitmap (256-color)`
  - `Black and white encapsulated PostScript`
  - `Black and white encapsulated PostScript (TIFF)`
  - `Black and white encapsulated PostScript2`

    - `Black and white encapsulated PostScript2 (TIFF)`
    - `Black and white PostScript`
    - `Black and white PostScript2`
    - `Color encapsulated PostScript`
    - `Color encapsulated PostScript (TIFF)`
    - `Color encapsulated PostScript2`
    - `Color encapsulated PostScript2 (TIFF)`
    - `Color PostScript`
    - `Color PostScript2`
    - `JPEG high quality image`
    - `JPEG medium quality image`
    - `JPEG low quality image`
    - `PNG 24-bit image`
    - `TIFF - compressed`
    - `TIFF - uncompressed`
    - `Windows metafile`
- **Paper orientation**:
    - `Landscape`
    - `Portrait`
    - `Rotated`
    - `Use figure orientation`: Uses the orientation for the figure, which you set with the `orient` command.
    - `Full page image (PDF only)`: In PDF reports, scales images to fit the full page, minimizes page margins, and maximizes the size of the image by using either a portrait or landscape orientation.

  For more information about paper orientation, see the `orient` command in the MATLAB documentation.
- **Image size**: Allows you to specify the image size in the report by selecting `Use figure PaperPositionMode setting` and setting the `PaperPositionMode` property of the Handle Graphics figure.

- `Automatic (same size as on screen)`:
- `Custom`: Specifies a custom image size. Set the image size using the **Size** field and **Units** list.

  For more information on paper position mode, see `orient` in the MATLAB documentation.

- **Size**: Allows you to enter the size of the Handle Graphics figure snapshot in the format `wxh` (width times height). This field is active only if you choose `Custom` in the **Image size** list box.
- **Units**: Allows you to enter for the size of the Handle Graphics figure snapshot. This field is active only if you choose `Custom` in the **Image size** list box.
- **Invert hardcopy**: Causes the Handle Graphics `InvertHardcopy` property to invert colors for printing. In other words, this option changes dark colors to light colors and light colors to dark colors. To change colors in your image, choose one of the following options:
  - `Automatic`: Automatically changes a dark axes colors to light axes colors. If the axes color is a light color, this option does not invert the color.
  - `Invert`: Changes dark axes colors to light axes colors, and light axes colors to dark axes colors.
  - `Don't invert`: Does not change the colors in the image that appears on the screen for printing.
  - `Use figure's InvertHardcopy setting`: Uses the `InvertHardcopy` property set in the Handle Graphics image.
  - `Make figure background transparent`: Makes the image background transparent.

Display Options

- **Scaling**: Controls size of the image, as displayed in a browser. Making an image larger using this option does not affect the storage size of the image, but the quality of the displayed image may decrease as you increase or decrease the size of the displayed image.

  Generally, to achieve the best and most predictable display results, use the default setting of `Use image size`.

  - `Use image size`: Causes the image to appear the same size in the report as on screen (default).
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
- **Size**: Specifies the size of the snapshot in the form `w h` (width, height) format. This field is active only if you choose `Fixed size` in the **Scaling** selection list.
- **Max size**: Specifies the maximum size of the snapshot in the form `w h` (width, height). This field is active only if you choose `Zoom` from the **Scaling** selection list.
- **Units**: Allows you to enter units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** list box.
- **Alignment**: Only reports in PDF or RTF format support this property.
  - `Auto`
  - `Right`
  - `Left`
  - `Center`
- **Title**: Enter text to appear above the snapshot.
- **Caption**: Enter text to appear under the snapshot.

Insert Anything into Report?

Yes. Figure and/or table.

Class

`rptgen_sl.csl_blk_lookup`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

Machine Loop

Run child components for specified Stateflow machines

Description

This component runs its child components for selected Stateflow machines. The behavior of this component depends on its parent component. If it has no parent, the `Machine Loop` runs its child components for all machines. If it has the `Model Loop` is its parent, it runs its child components for all machines in the model.

Loop Options

Search Stateflow

If selected, searches states that you specify in the field that appears under the check box.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to each object in the loop.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sf.csf_machine_loop`

See Also

`Model Loop`

Missing Requirements Block Loop

Apply all child components to blocks that do not have requirements

Description

This component runs its child components for each block in the current system, model, or signal that do not have associated requirements.

For more information on working with looping components, see “Logical and Looping Components”.

Report On

This pane describes the type of object on which this component operates.

- **Automatic list from context**: Report on all blocks in the current context that do not have associated requirements. The parent component of the Block Loop component determines its context. If this component does not have the `Model Loop`, `System Loop`, `Signal Loop`, or `Block Loop` as its parent, selecting this option causes this component to report on all blocks in all models that do not have associated requirements.
  - `Model Loop`: Reports on all blocks in the current model with no associated requirements.
  - `System Loop`: Reports on all blocks in the current system with no associated requirements.
  - `Signal Loop`: Reports on all blocks connected to the current signal with no associated requirements.
- **Custom - use block list**: Enables you to specify a list of blocks on which to report. Enter the full path of each block.

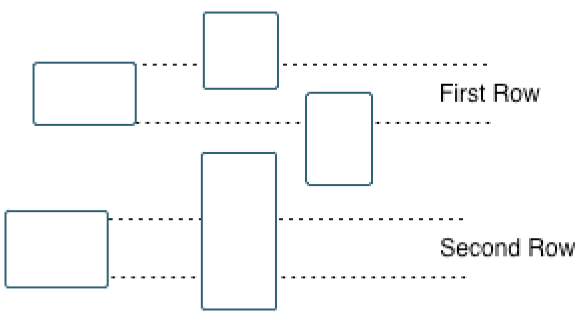
Loop Options

Choose block sorting options and reporting options in this pane.

- **Sort blocks**:

Use this option to select how to sort blocks (applied to each level in a model):

- `Alphabetically by block name`: Sorts blocks alphabetically by their names.
- `Alphabetically by system name`: Sorts systems alphabetically. Lists the blocks in each system, but in no particular order.
- `Alphabetically by full Simulink path`: Sorts blocks alphabetically by Simulink path.
- `By block type`: Sorts blocks alphabetically by block type.
- `By block depth`: Sorts blocks by their depth in the model.
- `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



- `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
- `By traversal order`. Sorts blocks by traversal order.
- `By simulation order`. Sorts blocks by execution order.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. The `%<>` notation can denote a string or cell array. The following example reports

on the `theta dot` integrator block and the `theta` integrator block in the model `simppend`, using the variable `Z={ 'simppend/theta'}`:

```
simppend/theta dot
%<Z>
```

The generated report includes information about the following blocks:

- `simppend/theta dot`
- `simppend/theta`

For more information, see `%<VariableName> Notation` on the `Text` component reference page in the MATLAB Report Generator documentation.

- **Search for Simulink property name/property value pairs**: Reports only on Simulink blocks with specified property name/property value pairs that do not have associated requirements.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each block found in the loop.
- **Display the object type in the section title**: Automatically inserts the object type into the section title in the generated report.
- **Create link anchor for each object in loop**: Create a hyperlink to the block in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class Name

`RptgenRMI.NoReqBlockLoop`

See Also

`Block Loop`, `Missing Requirements System Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Signal Loop`, `Requirements Summary Table`, `Requirements System Loop`, `Requirements Table`

MATLAB Function

Insert information about MATLAB Function block contents

Description

This component displays tables with information about MATLAB code included in MATLAB Function blocks. You specify which of the following kinds of information to include in the report:

- Function properties — Parameter settings for the MATLAB Function block
- Argument properties — Properties of the function arguments (for example, complexity)
- The function script — MATLAB code of the function
- Supporting functions — User-defined functions and, optionally, MATLAB functions that are included in the MATLAB Function block function.

For details about MATLAB Function blocks, see the MATLAB Function block reference page.

Use the MATLAB Function component within a section, paragraph, or table.

Note: To view the contents of a MATLAB Function block in a Web viewer, use the Web view feature of the Simulink Report Generator. In the Web view, hover your cursor over the MATLAB Function block. For details, see "Model Web Views".

Function Properties Table

- **Include function properties**: Generates a table with function property information.
- **Table title**: Insert a title for the function properties table.
  - `Automatic`: Use the default title for the table.
  - `Custom`: Use the title that you specify for the table.
- You can change the header text for property and value columns of the function properties table. In the `Header` column, double-click to change the header text. The `Width` column indicates the relative width, in relative terms, based on the smallest

width you specify. For example, for a three-column table, if the first column width is `1`, and the column width of the other two columns is `3`, then the second and third columns is three times wider than the first column.
- **Grid lines**: Show grid lines for the table.
- **Spans page width**: Make the table as wide as the page.

Argument Summary Table

- **Include argument summary table**: Generate a table with summary information about the MATLAB Function block function arguments.
- **Table title**: Insert a title for the argument summary table.
  - `Automatic`: Use the default title for the table.
  - `Custom`: Use the title that you specify for the table.
- **Argument Summary Table Options**: Specify the property columns to include in the table.
  - To add a property column:
    1. In the table on the right, select a property near where you want to insert the new property column.
    2. From the list of properties to the left of the table, select a property that you want to add to the table.
    3. Click the left-arrow button.
    4. If necessary, use the up or down arrow button to position the new column.
  - To delete a property column, select the property in the table and click the right-arrow button
  - You can change the header text for property and value columns of the table. In the `Header` column, double-click to change the header text. The `Width` column indicates the relative width, in relative terms, based on the smallest width you specify. For example, for a three-column table, if the first column width is `1`, and the column width of the other two columns is `3`, then the second and third columns is three times wider than the first column.
- **Grid lines**: Show grid lines for the table.
- **Spans page width**: Make the table as wide as the page.
- **Column alignment**: Align the text in each column:

  - `Left`
  - `Center`
  - `Right`
  - `Double justified`

Detailed Argument Report

- **Include detailed argument report**: Generate a table with detailed information about the MATLAB Function block function arguments.
- **Argument Property Table Format Options**: Specify the argument property columns to include in the table.
  - **Table title**: Insert a title for the argument properties table.
    - `Automatic`: Use the default title for the table.
    - `Custom`: Use the title that you specify for the table.
  - You can change the header text for property and value columns of the table. In the `Header` column, double-click to change the header text.
  - **Grid lines**: Show grid lines for the table.
  - **Spans page width**: Make the variable table as wide as the page on which the table appears.
- **Include function script**: Include the script for the function.
- **Highlight script syntax**: Use colors to highlight syntax keywords.
- **Include supporting functions**: Include a list of functions invoked directly or indirectly by the function script. If you specify to include supporting functions in the report, also specify whether to include both MATLAB and user-defined functions or just user-defined functions.
- **Supporting Function Table Format Options**:
  - **Table title**: Insert a title for the supporting functions table.
    - `Automatic`: Use the default title for the table.
    - `Custom`: Use the title that you specify for the table.
  - You can change the header text for property and value columns of the table. In the `Header` column, double-click to change the header text. The `Width` column

indicates the relative width, in relative terms, based on the smallest width you specify. For example, for a three-column table, if the first column width is 1, and the column width of the other two columns is 3, then the second and third columns is three times wider than the first column.

- **Grid lines**: Show grid lines for the table.
- **Spans page width**: Make the table as wide as the page.

Insert Anything into Report?

Yes. Tables and, optionally, code.

Class

`rptgen_sl.csl_emlfcn`

See Also

`Stateflow Property`

Missing Requirements System Loop

Loop only on systems and subsystems that do not have associated requirements

Description

This component runs its child components for each system or subsystem defined by the parent component that does not have associated requirements. Insert this component as the child of a `Model Loop` component to include systems and subsystems that do not have any associated requirements in the report.

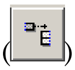
Report On

- **Loop on Systems**:
  - **Select systems automatically**: Reports on all systems in the current context that do not have associated requirements.
    - `Model Loop`: Reports on systems in the current model.
    - `System Loop`: Reports on the current system.
    - `Signal Loop`: Reports on the parent system of the current signal.
    - `Block Loop`: Reports on the parent system of the current block.

    If this component does not have any of these components as its parent, selecting this option reports on all systems in all models that do not have associated requirements.
- **Custom - use system list**: Reports on a list of specified systems. Specify the full path of each system.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. The `%<>` notation can denote a string or cell array. For more information, see `%<VariableName> Notation` on the `Text` component reference page.

Loop Options

- **Sort Systems**: Specifies how to sort systems.

    - `Alphabetically by system name` (default): Sorts systems alphabetically by name.
    - `By number of blocks in system`: Sorts systems by number of blocks. The list shows systems by decreasing number of blocks. In other words, it shows the system with the largest number of blocks that do not have requirements appears first in the list.
    - `By system depth`: Sorts systems by their depth in the model.
    - `By traversal order`: Sorts systems in the traversal order.
- **Search for**: Reports only on blocks with the specified property name/property value pairs. To enable searching, click the check box. In the first row of the property name and property value table, click inside the edit box, delete the existing text, and type the property name and value. To add a row, use the **Add row** button ().

  For information about subsystem property names and values, in “Block-Specific Parameters”, see the “Ports & Subsystems Library Block Parameters” section.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Number sections by system hierarchy**: Hierarchically numbers sections in the generated report. Requires that **Sort Systems** be set to `By traversal order`.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`RptgenRMI.NoReqSystemLoop`

See Also

`Block Loop`, `Missing Requirements Block Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Signal Loop`, `Requirements Summary Table`, `Requirements System Loop`, `Requirements Table`, `System Loop`

Model Advisor Report

Insert Model Advisor report or link to Model Advisor report for current model

Description

This component inserts a Model Advisor report for the current model into the report if the report is in HTML format. For other report formats, it inserts a link to a Model Advisor report for the current model. For more information about Model Advisor reports, see “View and Save Model Advisor Reports” in the Simulink documentation.

Properties

Use existing report: Includes an existing Model Advisor report in the report. This check box is selected by default. Clearing this option generates a new Model Advisor report.

Insert Anything into Report?

Yes, a Model Advisor report.

Class

`rptgen_sl.CModelAdvisor`

See Also

`Model Change Log`

Model Change Log

Construct model history table that displays model revision information

Description

Run this component before you run the `Model Simulation` component. It uses a reported model's `ModifiedHistory` parameter to construct a model history table that displays information about each logged revision to the model. This model history table includes:

- The author of each change
- The model version of the change
- The time and date of the change
- A description of the change

For more information on model history, see "Log Comments History" in the Simulink documentation.

Tip If your model has a long revision history, consider limiting the number of revisions reported.

Table Columns

Choose the information displayed in the model revision table in this section:

- **Author name**: Includes the name of the person who last revised the model.
- **Version**: Includes the version number of the model.
- **Date changed**: Includes the revision date of the model.
- **Description of change**: Includes a description of the revision to the model.

Table Rows

- **Limit displayed revisions to**: Limits the number of revisions that appears in the report.

- **Show revisions since date**: Limits the number of revisions that appears in the report by date. Enter the date in the corresponding text field. This field supports `%<varname>` notation. For example, the default value, `%<datestr(now-14)>`, returns revision history for the last two weeks.

Table Display

Choose how the model revision history table appears in this section.

- **Table title**: Specifies the title of the table.
- **Sort order**: Sorts the table entries from most recent to oldest, or from oldest to most recent.
- **Date format**: Specifies a preferred date format for the date/time stamps in the table.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_mdl_changelog`

See Also

`Model Advisor Report`

Model Configuration Set

Insert active configuration set of a model into a report

Description

This component displays a table with the active configuration set for the model.

For information about configurations sets, see “Manage a Configuration Set”.

Display Options

- **Title**: Specifies a title for the table in the generated report.
  - `Automatic`: Generates a title automatically from the parameter.
  - `Custom`: Specifies a custom title.
  - `None`: Uses no title.
- **Show configuration set table grids**: Show grid lines for the table.
- **Make configuration set tables page wide**: Make the table as wide as the page.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_mdl_cfgset`

See Also

`Model Loop`, `System Loop`

Model Loop

Loop on Simulink models and systems, as specified by child components

Description

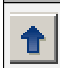
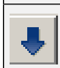
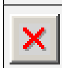
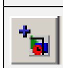
This component loops on Simulink models and systems, as specified by child components. For example, you can use a `Model Loop` with a child `System Loop` to report on the subsystems of the specified system.

Consider making these components children of the `Model Loop` (although the `Model Loop` not necessarily required to be the immediate parent of a given component).

For conditional processing based of blocks, you can use the `RptgenSL.getReportedBlock` function. For more information, see “Loop Context Functions”.

Models to Include

You can add a model to the list by clicking **Add New Model to List**. The following table shows the buttons you can use to move a model up or down in the list, or to add or delete a model.

Button	Action
	Move a model up in the list.
	Move a model down in the list.
	Remove a model from the list.
	Add a new model to the list.

Model Options

- **Active**: Includes a given model in the loop. This option is selected by default. Clearing this option omits the model from the loop.

  This option allows you to temporarily omit one or more models from a report.
- **Model name**: Specifies the model name.
  - `Current block diagram`
  - `All open models`
  - `All open libraries`
  - `Block diagrams in current directory`
  - `Custom block diagram`: Selecting this option automatically sets the **Starting system(s)** field `$top` to start in the model root system.
  - `%<VariableName>`: For more information, see `%<VariableName> Notation` on the `Text` component reference page in the MATLAB Report Generator documentation.
- **Traverse model**: Specifies the systems to traverse.
  - `All systems in model`
  - `Selected system(s) only`
  - `Selected system(s) and ancestors`
  - `Selected system(s) and children`
- **Look under masks**: Specifies how to handle masks.
  - `Functional masks only`
  - `No masks`
  - `All masks`
  - `Graphical masks only`

  For more information, see “Masking” in the Simulink documentation.
- **Follow library links**: Specifies library links to include.
  - `Do not follow library links`
  - `Include library links`

- `Include unique library links`

For more information, see “Work with Library Links ” in the Simulink documentation.

- **Model reference**: Specifies whether to report on models referenced by a Model block. If you want to report on referenced models, then you can control the depth of the model referencing hierarchy and whether to report on model reference variants.

  - `Do not follow model reference blocks`: Do not report on blocks contained in referenced models.
  - `Follow all model reference blocks`: Report on blocks contained in all models that any part of the model hierarchy references.
  - `Follow model reference blocks defined in current model`: Report on blocks in models that the currently selected model references.
  - `<Custom model reference depth>`: Report on blocks in models that your specified level in the model reference hierarchy references.
- **Include all variants**: Report on all model reference variants. To enable this option, set the **Model reference** option to report on blocks in referenced models.
- **Starting system(s)**: Specifies the system in which to start the loop. Available options depend on the value that you select in the **Traverse model** option. Selecting any option other than `All systems in model` for **Traverse model** activates the **Starting system(s)** option.

  If you do not enter a model name in the **Model name** option, then select either `Root model` or `Current` to specify where to start the loop.

  If you specify a model name in the **Model name** option, then the **Starting system(s)** option provides an edit box in which you can enter:

  - The full path of a subsystem or subsystems
  - `$top` to start the loop in the model root system
  - `$current` to start the loop in the currently selected system

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.

- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Examples

Generating Reports on Specified Systems and their Subsystems

This example shows how to loop over a specified system and its subsystems in the sample model `sldemo_auto_climate_elec`, which the Simulink software includes.

1. (Optional) To open the `sldemo_auto_climate_elec` model, at the MATLAB command prompt, enter the following command:

   ```
   sldemo_auto_climate_elec
   ```

   Explore the model to familiarize yourself with its subsystems.
2. Open the Report Explorer.
3. Create a report setup file by clicking **File** > **New**.
4. Save the report setup file by clicking **File** > **Save As**. Give it the name `sldemo_auto_report`.
5. Add a `Chapter/Subsection` component to the report setup file to include information about model subsystems:

   a. In the Library pane in the middle, double-click `Chapter/Subsection` to add it to the report setup file.
   b. For **Title**, choose `Custom`. In the title field, enter `Description of subsystems`.
6. Add a `Model Loop` as a child of the `Chapter/Subsection` component. This loops over the `ClimateControlSystem` system and its subsystems in the `sldemo_auto_climate_elec` model:

   a. In the Library pane in the middle, double-click `Model Loop` to add it to the report setup file. By default, the Report Explorer adds that component as a child of the `Chapter/Subsection` component.
   b. In the Model Loop properties pane, from the **Model name** selection list, select `<Custom block diagram>`.

c In the **Model name** field, delete the text `<Custom block diagram>`, and then enter `sldemo_auto_climate_elec.slx`. Click any component in the report setup file to add this model to the **Models to include** list.

d In the **Traverse model** selection list, select `Selected system(s) and children`.

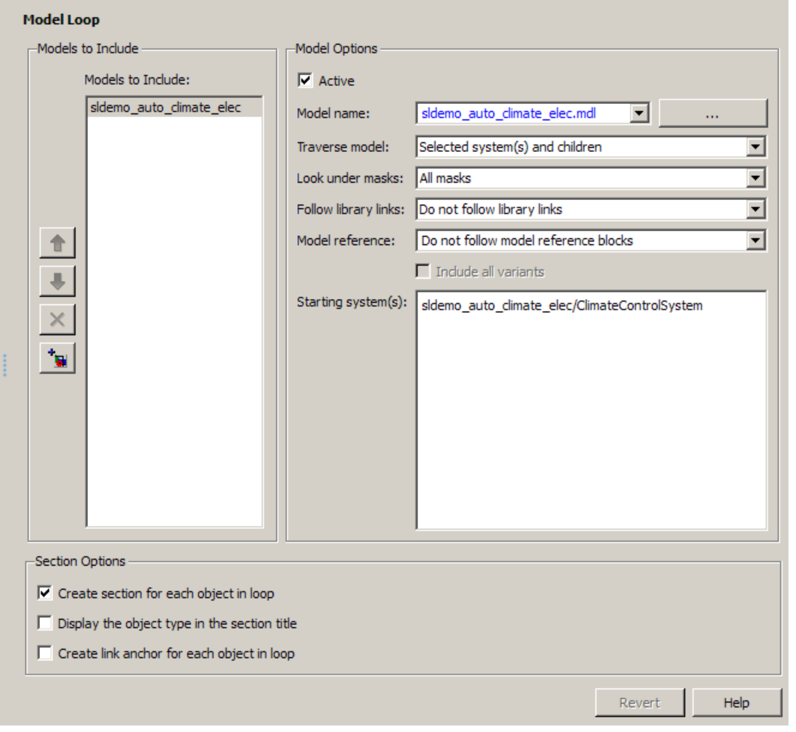
e In the **Look under masks** selection list, select `All masks`.

f In the **Model reference** selection list, select `Do not follow model reference blocks`.

g In the **Starting system(s)** field, enter `sldemo_auto_climate_elec/ClimateControlSystem`. Because you selected `Selected system(s) and children` for **Traverse model**, the `Model Loop` loops over `sldemo_auto_climate_elec/ClimateControlSystem` and its subsystems.

h Under **Section Options**, select the **Create section for each object in loop** check box. Selecting this option creates separate sections in the generated report for each model over which the component loops.

The `Model Loop` properties pane looks as follows.



7 Save the report by clicking **File** > **Save**.

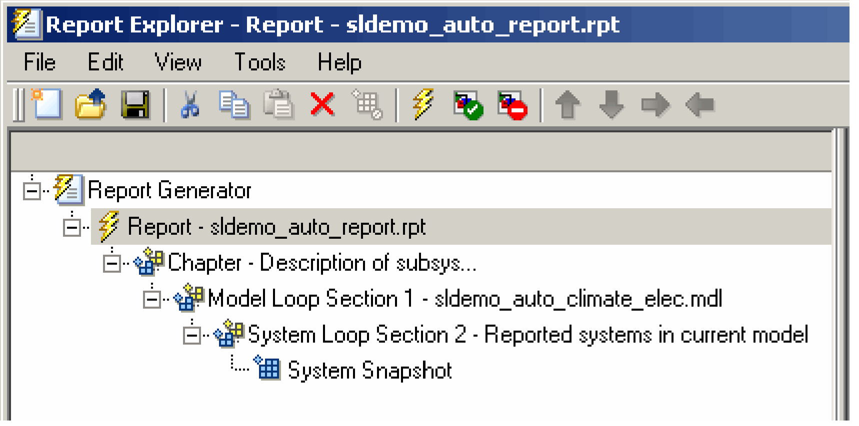
8 Add a `System Loop` as a child of the `Model Loop` component.

- **a** In the Library pane in the middle, double-click `System Loop` to add it to the report setup file. By default, Model Explorer adds this component as a child of the `Model Loop` component.
- **b** In the `System Loop` properties pane, under **Section Options**, select the **Create section for each object in loop** check box. Selecting this option creates a section in the generated report for each subsystem on which the component loops. Accept the default values for all other fields.

9 Add a `System Snapshot` component as a child of the `System Loop` component. This step creates snapshots of all the subsystems of `ClimateControlSystem` in the generated report. In the Library pane in the middle, double-click `System Snapshot`. By default, Model Explorer adds this component as a child of the `System Loop` component.

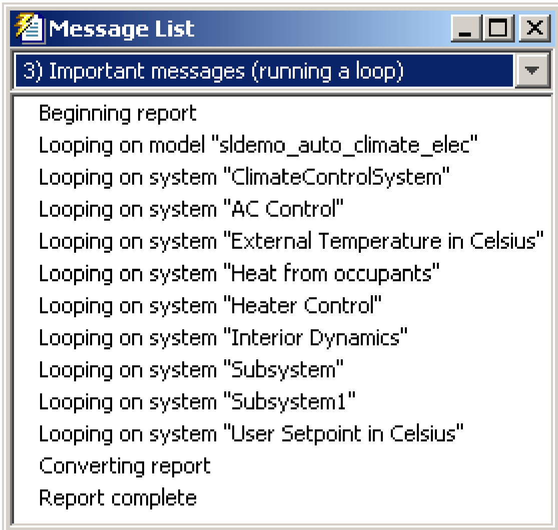
10 Save the report.

The report setup file hierarchy now looks as follows.



11 Run the report by clicking **File** > **Report**.

The report loops on the system `ClimateControlSystem` of the `sldemo_auto_climate_elec` model and all of its subsystems, as shown in the following Message List.



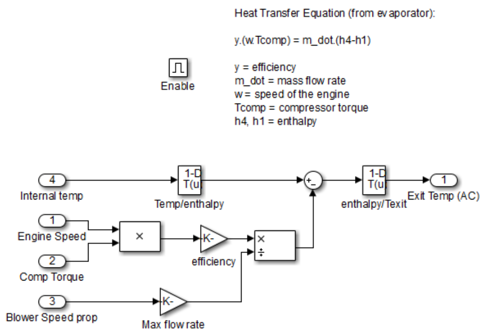
Below is an excerpt from the generated report.

Chapter 1. Description of subsystems

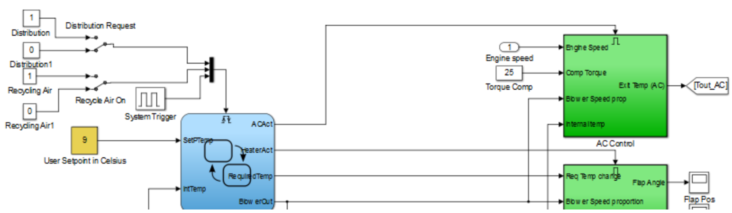
Table of Contents

sldemo_auto_climate_elec

AC Control



ClimateControlSystem

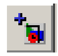


Temporarily Omitting a Model from a Loop

This example shows how to use the `Model Loop` **Active** check box to temporarily omit a model from the loop. This example uses the report setup file that you created in Generating Reports on Specified Systems and their Subsystems, `sldemo_auto_report.rpt`, and the model `f14`, which the Simulink software includes.

1. In the Report Explorer, click **File** > **Open**, and then open `sldemo_auto_report.rpt` by double-clicking it.

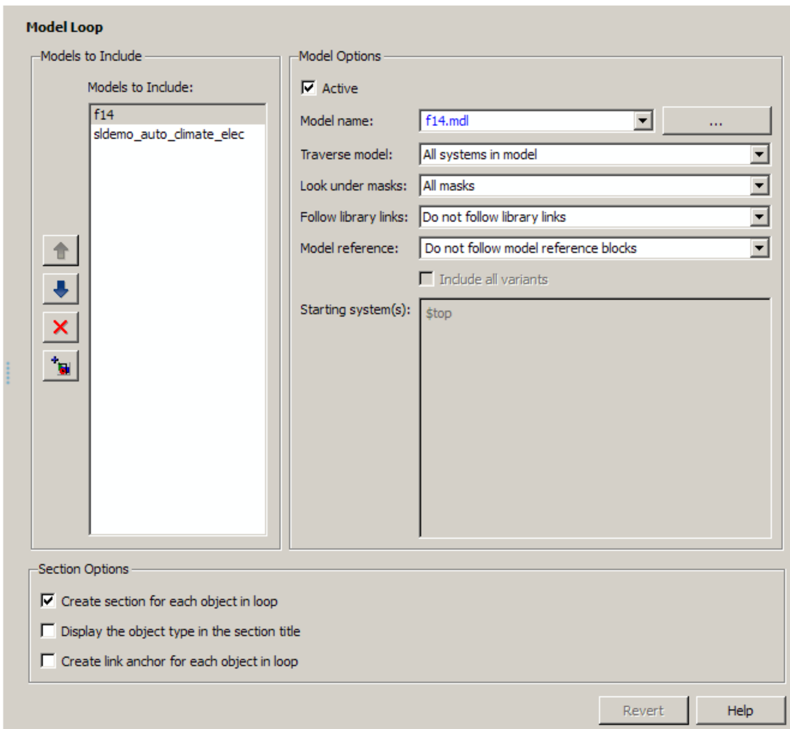
2 In the Outline pane on the left, click `Model Loop Section 1 - sldemo_auto_climate_elec`.

3 In the `Model Loop` properties pane, click the  button to add a model to the **Models to include** list.

4 In the `Model Loop` properties pane, from the **Model name** selection list, select `<Custom block diagram>`.

5 In the **Model name** field, delete the text `<Custom block diagram>` and enter `f14.mdl`.

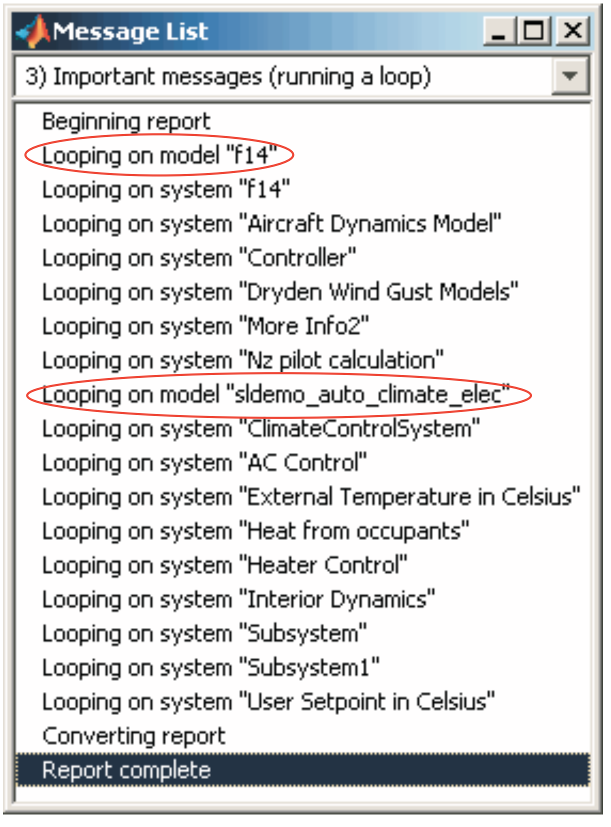
The `Model Loop` properties pane now looks as follows.



6 Save the report setup file.

7 Generate the report.

The report generation process loops over the specified systems in the `f14` and `sldemo_auto_climate_elec` models, as shown in the following message box.



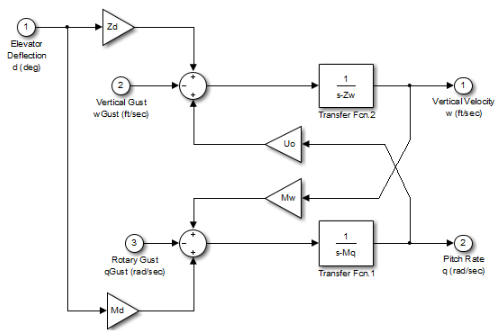
Below is an excerpt from the generated report.

Chapter 1. Description of subsystems

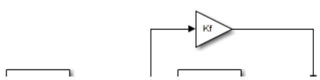
Table of Contents

f14

Aircraft Dynamics Model



Controller



8. In the **Models to include** list, click `f14` to select it.
9. Clear the **Active** check box to omit `f14` model information from the generated report.
10. Rerun the report.

    The report now includes information only on the `sldemo_auto_climate_elec` model, as shown at the end of the previous example, Generating Reports on Specified Systems and their Subsystems.

11 To reactivate the `f14` model, in the Model Loop **Models to include** list, select the `f14` model and then select the **Active** check box.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sl.csl_mdl_loop`

See Also

`Block Loop`, `System Loop`

Model Simulation

Run current model with specified simulation parameters

Description

This component runs the current model using specified simulation parameters. Ensure that this component has the `Model Loop` component as its parent.

For more information on simulation parameters, see "Configure Simulation" in the Simulink documentation.

I/O Parameters

Use model's workspace I/O variable names

Use the names of the parameters specified in the Simulation Parameters dialog box.

The following options are available if you do not select the **Use model's workspace I/O variable names** option:

- **Time** : Specifies a new variable name for the `Time` parameter.
- **States**: Specifies a new variable name for the `States` parameter.
- **Output**: Specifies a new variable name for the `Output` parameter.

Timespan

Use model's timespan values: Use the model's `Start time` and `Stop time` values, as specified in the **Solver** tab in the Simulation Parameters dialog box.

The following options are available if you do not select the **Use model's timespan values** option:

- **Start**: Specifies a simulation starting time.
- **Stop**: Specifies a simulation ending time.

Note: If you set the stop time of your model to `inf` (infinity) in Simulink or on this component attribute page, Simulink Report Generator terminates the model simulation after 60 seconds. Terminating the report prevents the report generation process from entering an infinite loop.

Simulation Options

- **Compile model before simulation**: Compiles the model before simulating, preserving scope content. Select this option if:
  - You use Simulink Coder Summary properties.
  - You sort systems or blocks by simulation order.
  - You use scope snapshots.
- **Simulation status messages**: Displays simulation status messages, or inserts them into the report.
  - `Display to command line`: Sends messages to a command-line window.
  - `Display to Report Generator Message List`: Sends messages to the Simulink Report Generator message window.
  - `Insert into report`: Includes messages in the report.
- **Simulation parameters**: Specifies simulation parameters.

Insert Anything into Report?

No.

Class

`rptgen_sl.csl_mdl_sim`

See Also

`Model Loop`

Object Loop

Run child components for Stateflow objects, and then insert table into report

Description

This component runs its child components for each Stateflow object and inserts a table into the generated report.

For conditional processing of Stateflow objects, you can use the `RptgenSF.getReportedObject` function. For more information, see “Loop Context Functions”.

Object Types

- **Report on “Data” objects**: Includes Stateflow data objects in the loop.
- **Report on “Event” objects**: Includes Stateflow event objects in the loop.
- **Report on “Transition” objects**: Includes Stateflow transition objects in the loop.
- **Report on “Junction” objects**: Includes Stateflow junction objects in the loop.
- **Report on “Target” objects**: Includes Stateflow target objects in the loop.
- **Report on “Note” objects**: Includes Stateflow note objects in the loop.

Loop Options

- **Report depth**: Specifies the level at which to loop:
  - `Local children only` (Default). Reports only on children one level down.
  - `All objects`. Reports on all Stateflow objects.
- **Skip autogenerated charts under truth tables**: Excludes autogenerated charts under truth tables from the report.
- **Remove objects which do not contain more information than a snapshot**: Excludes objects that contain only a snapshot.
- **Search Stateflow**: Reports on Stateflow charts with specified property name/property value pairs.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Automatically inserts the object type into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the Stateflow object in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sf.csf_obj_loop`

See Also

`Stateflow Filter`, `Stateflow Hierarchy`, `Stateflow Hierarchy Loop`, `Simulink Function System Loop`

Recursive Functions

Create table of recursive functions

Description

This component creates a table containing the recursive functions in your source code. For each recursive function, the table lists its immediate caller.

Report Customization (Filtering)

Create filters that apply to your Polyspace reports

Description

This component allows you to filter unwanted information from existing Polyspace report templates. To apply global filters, place this component immediately below the node representing the report name.

Properties

Code Metrics Filters

The properties in table below apply to the inclusion of code metrics in your report.

Property	Purpose	User Action
Include Project Metrics	Choose whether to include metrics about your Polyspace project.	Select the check box to include project metrics.
Project metrics to include	Specify project metrics to include or exclude from report.	Enter a regular MATLAB expression.
Include File Metrics	Choose whether to include per file metrics in report.	Select the check box to include per file metrics.
File Metrics > Files to include	Specify files to include or exclude when reporting file metrics.	Enter a regular MATLAB expression.
File metrics to include	Specify file metrics to include or exclude from report.	Enter a regular MATLAB expression.
Include Function Metrics	Choose whether to include per function metrics in report.	Select the check box to include per function metrics.

Property	Purpose	User Action
Function Metrics > Files to include	Specify files to include or exclude when reporting function metrics.	Enter a regular MATLAB expression.
Functions to include	Specify functions to include or exclude when reporting function metrics.	Enter a regular MATLAB expression.
Function metrics to include	Specify function metrics to include or exclude from report.	Enter a regular MATLAB expression.

Coding Rules Filters

The properties in table below apply to the inclusion of coding rule violations in your report.

Property	Purpose	User Action
Files to include	Specify files to include or exclude when reporting coding rule violations.	Enter a regular MATLAB expression.
Coding rule numbers to include	Specify coding rules to include or exclude when reporting coding rule violations.	Enter a regular MATLAB expression.
Classifications to include	Specify classifications to include or exclude when reporting coding rule violations.	Enter a regular MATLAB expression.
Status types to include	Specify statuses to include or exclude when reporting coding rule violations.	Enter a regular MATLAB expression.

Run-time Check Filters

The properties in table below apply to the inclusion of Polyspace Code Prover checks in your report.

Property	Purpose
Red Checks	Specify whether to include red checks in your report. Red checks indicate proven run-time errors.
Gray Checks	Specify whether to include gray checks in your report. Gray checks indicate unreachable code.
Orange Checks	Specify whether to include orange checks in your report. Orange checks indicate possible run-time errors.
Green Checks	Specify whether to include green checks in your report. Green checks indicate that an operation does not contain a specific run-time error.
Inspection Point Checks	Specify whether to include inspection point checks in your report. These checks allow an user to find the values that a variable can take at a certain point in the code.
Unreachable Functions	Specify whether to include unreachable functions in your report.

Advanced Filters

The properties in table below apply to the inclusion of metrics, coding rule violations and Polyspace Code Prover checks in your report.

Property	Purpose	User Action
Justification status	Choose whether to report only justified checks, only unjustified checks or all checks.	Choose an option from the dropdown list.
Files to include	Specify files to include or exclude from your report.	Enter a regular MATLAB expression.
Check types to include	Specify Polyspace Code Prover checks to include in your report.	Enter a regular MATLAB expression.

Property	Purpose	User Action
Function names to include	Specify functions to include or exclude from your report.	Enter a regular MATLAB expression.
Classification types to include	Specify classifications to include or exclude from your report.	Enter a regular MATLAB expression.
Status types to include	Specify statuses to include or exclude from your report.	Enter a regular MATLAB expression.
Comments to include	Specify comments to include or exclude from your report.	Enter a regular MATLAB expression.

More About

- "Regular Expressions"

Requirements Block Loop

Apply child components to blocks with requirements

Description

This component applies its child components to blocks with associated requirements.

Report On

- **Automatic list from context**: If selected, this option reports on all blocks in the current context. The parent of the `Requirements Block Loop` component determines its context.

  - `Model Loop`: Reports on all blocks with requirements in the current model.
  - `System Loop`: Reports on all blocks with requirements in the current system.
  - `Signal Loop`: Reports on all blocks with requirements connected to the current signal.

  If the `Requirements Block Loop` does not have the `Model Loop`, `System Loop`, `Signal Loop`, or `Block Loop` component as its parent, it reports on all blocks in all models.
- **Custom - use block list**: Reports on a list of blocks with specified requirements. Enter the full paths of each block into this field.

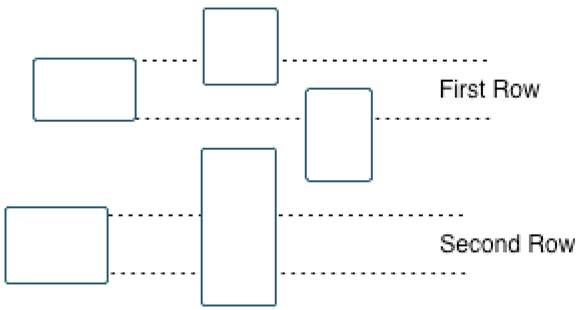
Loop Options

- **Sort blocks**

  Specify how to sort blocks (applied to each level in a model):

  - `Alphabetically by block name`: Sorts blocks alphabetically by name.
  - `Alphabetically by system name`: Sorts systems and subsystems alphabetically by name. (Blocks in each system do not appear in alphabetical order).

    - `Alphabetically by full Simulink path`: Sorts blocks alphabetically by Simulink path.
    - `By block type`: Sorts blocks alphabetically by block type.
    - `By block depth`: Sorts blocks by their depth in the model.
    - `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



    - `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
    - `By traversal order`: Sorts blocks by traversal order.
    - `By simulation order`: Sorts blocks by execution order.
- **Search for Simulink property name/property value pairs**: Reports on Simulink blocks with specified property name/property value pairs that have associated requirements.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each block found in the loop that has associated requirements.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the block in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`RptgenRMI.CBlockLoop`

See Also

`Missing Requirements Block Loop`, `Missing Requirements System Loop`, `Model Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Signal Loop`, `Requirements Summary Table`, `Requirements System Loop`, `Requirements Table`

Requirements Documents Table

Insert table of linked requirements documents

Description

This component creates a table that lists all requirements documents linked to model objects.

Table Options

- **Show documents linked to**
  - `Simulink and Stateflow objects`: Inserts requirements documents linked to both Simulink and Stateflow objects in the model.
  - `Simulink objects`: Inserts requirements documents linked only to Simulink objects in the model.
  - `Stateflow objects`: Inserts requirements documents linked only to Stateflow objects in the model.
- **Table title**: Specifies a title for the table.
  - `No title`
  - `Model name` (Default)
  - `Custom`

Table Columns

- **Replace document paths with links**: Inserts links to requirements documents when possible.
- **When replacing with links, note absolute vs. relative file path**: Indicates absolute or relative file paths when including links to requirements documents.
- **Include document modification time**: Includes the document modification information.

- **Count # references to each document**: Includes a count of the number of references to the requirements document in the model.

Document References

- **Replace file names with document IDs in the main body of the report**: Includes shortened IDs to identify requirements documents to simplify the requirements documents table.
- **Retrieve full module path for DOORS links (requires login)**: This option applies only to DOORS® requirements. Append the DOORS module ID to the module path in the DOORS database if the module information is not stored with the model.

Insert Anything into Report?

Yes. Table.

Class

`RptgenRMI.ReqDocTable`

See Also

, `Requirements Summary Table`, `Requirements Table`

Requirements Signal Loop

Apply all child components to signal groups with requirements

Description

The Requirements Signal Loop component applies all child components to signal groups that have requirements in Signal Builder blocks.

Properties

- **Create link anchor for each object in loop**: Creates a hyperlink to each object with requirements in the loop.
- **Display the object type in the section title**: Inserts the object name with requirements into the section title.
- **Create section for each object in loop**: Creates a hyperlink to each object with requirements in the loop.
- **Section Type**: Specifies the section type to insert. If you choose `Automatic`, the Simulink Report Generator software determines the appropriate section type:
  - `Book`
  - `Chapter`
  - `Section 1`
  - `Section 2`
  - `Section 3`
  - `Section 4`
  - `Section 5`
  - `Simple Section`
  - `Automatic`

Report On

Loops on signal groups in systems:

- `Collect all Signal Builders`: Processes all Signal Builder blocks, looking for signal groups with requirements.
- `Custom - use list`: Processes all subsystems in the user-defined list. If a subsystem on the list does not have requirements, the Simulink Report Generator software does not include it in the report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`RptgenRMI.CSystemLoop`

See Also

`Missing Requirements Block Loop`, `Missing Requirements System Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Summary Table`, `Requirements System Loop`, `Requirements Table`, `Signal Loop`

Requirements Summary Table

Properties of blocks, systems, or Stateflow objects with associated requirements

Description

This component displays properties of blocks, systems, or Stateflow objects with associated requirements.

Object Type

Choose the object type to display in the generated report.

- `Block` (Default)
- `System`
- `Stateflow`

The selected object type affects the options available in the **Property Columns** pane.

Table Title

Specify a table title in the generated report.

- `Automatic`: Generates a title automatically from the parameter.
- `Custom`: Specifies a custom title.

Property Columns

- Object properties to include in the Requirements Summary Table appear in a list.
  - To add a property:
    1. Select the appropriate property level in the text box on the left.
    2. In the text box on the right, select the property that you want to add and click **Add**.
  - To delete a property, select the property name and click **Delete**.

`%<SplitDialogParameters>` is a unique property that you can specify for Requirements Summary Tables where the object type is `Block`. This property generates multiple summary tables, grouped by block type. Each Summary Table group contains the dialog box parameters for that block.

Some entries in the list of available properties (such as `Depth`) are "virtual" properties that you cannot access using the `get_param` command. The properties used for property/value filtering in the block and System Loop components must be retrievable by the `get_param`. Therefore, you cannot configure your Requirements Summary Table to report on all blocks of `Depth` == 2.

- **Remove empty columns**: Removes empty columns from the table.
- **Transpose table**: Changes the summary table rows into columns in the generated report, putting the property names in the first column and the values in the other columns.

Object Rows

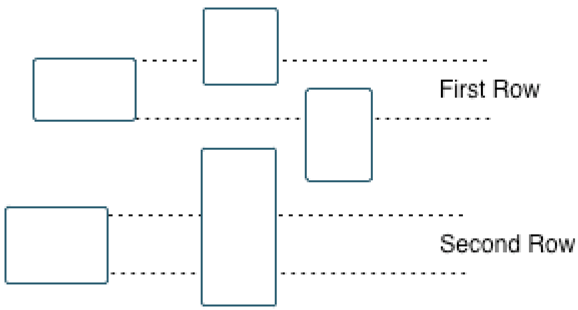
- **Insert anchor for each row**: Inserts an anchor for each row in the Requirements Summary Table.
- **Report On**
  - `Automatic list from context`: Reports on all blocks in the current context. The parent of this component determines its context.
  - `Custom - use block list`: Reports on a list of blocks that you specify, and enters the block names in the corresponding field. Specify the full path of each block.

Loop Options

Choose block sorting options and reporting options in this pane.

- **Sort blocks**: Use this option to select how to sort blocks (applied to each level in a model):
  - `Alphabetically by block name`: Sorts blocks alphabetically by name.
  - `Alphabetically by system name`. Sorts systems alphabetically. Lists blocks in each system, but in no particular order.

- `Alphabetically by full Simulink path`: Sorts blocks alphabetically by Simulink path.
- `By block type`: Sorts blocks alphabetically by block type.
- `By block depth`: Sorts blocks by their depth in the model.
- `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



- `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
- `By traversal order`: Sorts blocks by traversal order.
- `By simulation order`: Sorts blocks by execution order.

- **Search for Simulink property name/property value pairs**: Reports on blocks with specified property name/property value pairs.

Insert Anything into Report?

Yes. Table.

Class

`RptgenRMI.CSummaryTable`

See Also

`Block Loop`, `Missing Requirements Block Loop`, `Missing Requirements System Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Signal Loop`, `Requirements System Loop`, `Requirements Table`

Requirements System Loop

Apply child components to systems with requirements

Description

This component applies its child components to systems with associated requirements.

Report On

- **Loop on systems**
  - `Select systems automatically`: If selected, this option reports on all systems in the current context. The parent of the component determines the context of this setting:
    - `Model Loop`: Reports on systems in the current model.
    - `System Loop`: Reports on the current system.
    - `Signal Loop`: Reports on the parent system of the current signal.
    - `Block Loop`: Reports on the parent system of the current block.

    If the `Requirement System Loop` does not have any of these components as its parent, selecting this option reports on all systems with requirements in all models.
  - `Custom - use system list`: Reports on a list of specified systems. Enter the full path of each system.

Loop Options

- **Sort Systems**:
  - `Alphabetically by system name` (default): Sorts systems alphabetically by name.
  - `By number of blocks in system`: Sorts systems by the number of blocks in the system. The list displays systems by decreasing number of blocks; the system with the largest number of blocks appears first in the list.

  - `By system depth`: Sorts systems by their depth in the model.
  - `By traversal order`: Sorts systems in the traversal order .
- **Search for**: Reports on Simulink blocks with specified property name/property value pairs.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Number sections by system hierarchy**: Numbers sections in the generated report hierarchically. Requires that **Sort Systems** be set to `By traversal order`.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`RptgenRMI.CSystemLoop`

See Also

`Missing Requirements Block Loop`, `Missing Requirements System Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Signal Loop`, `Requirements Summary Table`, `Requirements Table`, `System Loop`

Requirements Table

Requirements links for current context

Description

This component creates a table that contains information from the Simulink Verification and Validation software. Objects can have multiple requirements. Each requirement is a row in the table.

Table Options

- **Show requirements for current**: Specifies the object type to display.
  - `Simulink object`
  - `Stateflow object`
- **Table title**: Specifies a title for the table.
  - `No title`
  - `Object name` (Default)
  - `Custom`

Table Columns

- **Description**: Includes the object description in the table.
- **Document name**: Includes the report name in the table.
- **Locations within document**: Includes the locations of the object within the document in the table.
- **Requirement keyword**: Includes the requirement keyword for the object in the table.

Insert Anything into Report?

Yes. Table.

Class

`RptgenRMI.CReqTable`

See Also

`Missing Requirements Block Loop`, `Missing Requirements System Loop`, `Requirements Block Loop`, `Requirements Documents Table`, `Requirements Signal Loop`, `Requirements Summary Table`, `Requirements System Loop`, `Stateflow Automatic Table`, `Stateflow Name`

Run-time Checks Details Ordered by Color/File

Create overrides for global filters in Polyspace reports

Description

This component adds detailed information about the run-time checks to your report. This component can also be used to override global filters in specific chapters of your report. Use the following workflow when using filters in your report:

1. To create filters that apply to all chapters of your report, use the **Report Customization (Filtering)** component. For more information, see Report Customization (Filtering).
2. To override some of the filters in individual chapters, use the **Run-time Checks Details Ordered by Color/File** component. Select the **Override Global Report filter** box.

Properties

Categories To Include

The properties in table below apply to the inclusion of Polyspace Code Prover checks in your report.

Property	Purpose
Red Checks	Specify whether to include red checks in your report. Red checks indicate proven run-time errors.
Gray Checks	Specify whether to include gray checks in your report. Gray checks indicate unreachable code.
Orange Checks	Specify whether to include orange checks in your report. Orange checks indicate possible run-time errors.

Property	Purpose
Green Checks	Specify whether to include green checks in your report. Green checks indicate that an operation does not contain a specific run-time error.
Inspection Point Checks	Specify whether to include inspection point checks in your report. These checks allow an user to find the values that a variable can take at a certain point in the code.
Unreachable Functions	Specify whether to include unreachable functions in your report.

Advanced Filters

The properties in table below apply to the inclusion of metrics, coding rule violations and Polyspace Code Prover checks in your report.

Property	Purpose	User Action
Justification status	Choose whether to report only justified checks, only unjustified checks or all checks.	Choose an option from the dropdown list.
Files to include	Specify files to include or exclude from your report.	Enter a regular MATLAB expression.
Check types to include	Specify Polyspace Code Prover checks to include in your report.	Enter a regular MATLAB expression.
Function names to include	Specify functions to include or exclude from your report.	Enter a regular MATLAB expression.
Classification types to include	Specify classifications to include or exclude from your report.	Enter a regular MATLAB expression.
Status types to include	Specify statuses to include or exclude from your report.	Enter a regular MATLAB expression.
Comments to include	Specify comments to include or exclude from your report.	Enter a regular MATLAB expression.

Run-time Checks Details Ordered by Review Information

Create table with Polyspace Code Prover checks ordered by review information

Description

This component creates tables displaying the Polyspace Code Prover checks in your code. All checks with same combination of **Classification** and **Status** appear in the same table.

Run-time Checks Summary Ordered by File

Create table with Polyspace Code Prover checks ordered by file

Description

This component creates a table displaying the number of Polyspace Code Prover checks per file in your code.

Properties

Sort the data

Use this option to sort the rows in the table alphabetically by filename or by percentage of unproven code.

Display as

Use this option to display the number of checks in a table or in bar charts.

Display ratio of checks in a file

Select this option to display the number of checks of a certain color as a ratio of total number of checks in the file.

Include checks from Polyspace standard library stub functions

Select this option to include the checks from Polyspace standard library stub functions in your display.

Scope Snapshot

Insert images of scopes and XY graphs

Description

This component inserts images of scopes and XY graphs. Examples of blocks for which this component inserts snapshots include:

- Scope (and Floating Scope) blocks and the XY Graph block (Simulink)
- Spectrum Analyzer and Time Scope blocks (DSP System Toolbox™)
- Video Viewer (Computer Vision System Toolbox™)
- Blocks in the Simulink Control Design™ Linear Analysis Plots library (for example, the Bode Plot block)

If the model has not been simulated, scopes are empty. For more information, see the `Model Simulation` component reference page.

The parent component of the Scope Snapshot determines its behavior.

- `Model Loop` or `no Simulink looping component`: Includes all XY graphs and scopes in the current model.
- `System Loop`: Includes all XY graphs and scopes in the current system.
- `Block Loop`: Includes the current block when it is an XY graph or scope.
- `Signal Loop`: Includes all XY graphs and scopes connected to the current signal.

  If the Scope Snapshot does not have any of the Simulink looping components as its parent, it includes all XY graphs and scopes in all open models.

Scope Options

- **Report on closed scopes**: Takes a snapshot of all scopes in context. This option forces closed scopes to open when the report is generating.
- **Autoscale time axis**: Scales the Simulink scope time axis to include the entire log.

Print Options

- **Image file format**: Specifies the image file format (for example, JPEG, TIFF, etc.). Select `Automatic HG Format` (the default) to choose the format best suited for the specified output format automatically. Otherwise, choose an image format that your output viewer can read.
  - `Automatic HG Format` (uses the Simulink file format selected in the Preferences dialog box)
  - `Bitmap (16m-color)`
  - `Bitmap (256-color)`
  - `Black and white encapsulated PostScript`
  - `Black and white encapsulated PostScript (TIFF)`
  - `Black and white encapsulated PostScript2`
  - `Black and white encapsulated PostScript2 (TIFF)`
  - `Black and white PostScript`
  - `Black and white PostScript2`
  - `Color encapsulated PostScript`
  - `Color encapsulated PostScript (TIFF)`
  - `Color encapsulated PostScript2`
  - `Color encapsulated PostScript2 (TIFF)`
  - `Color PostScript`
  - `Color PostScript2`
  - `JPEG high quality image`
  - `JPEG medium quality image`
  - `JPEG low quality image`
  - `PNG 24-bit image`
  - `TIFF - compressed`
  - `TIFF - uncompressed`
  - `Windows metafile`
- **Paper orientation**:

    - `Landscape`
    - `Portrait`
    - `Rotated`
    - `Use figure orientation`: Uses the orientation for the figure, which you set with the `orient` command.
    - `Full page image (PDF only)`: In PDF reports, scales images to fit the full page, minimizes page margins, and maximizes the size of the image by using either a portrait or landscape orientation.

  For more information about paper orientation, see the `orient` reference page in the MATLAB documentation.

- **Image size**: Specifies the size of the Handle Graphics figure snapshot in the form `[w h]` (width, height). In the units text box, select one of the following options:

    - `Inches`
    - `Centimeters`
    - `Points`
    - `Normalized`

- **Invert hardcopy**: Inverts colors for printing; changes dark colors to light colors and light colors to dark colors.

    - `Automatic`: Automatically changes dark axes colors to light axes colors. If the axes color is a light color, this option does not invert the color.
    - `Invert`: Changes dark axes colors to light axes colors and light axes colors to dark axes colors.
    - `Don't invert`: Does not change the colors in the image on the screen for printing.
    - `Use figure's InvertHardcopy setting`: Uses the `InvertHardcopy` property set in the Handle Graphics image.
    - `Make figure background transparent`: Makes the image background transparent.

Display Options

- **Scaling**: Controls size of the image, as displayed in a browser. Making an image larger using this option does not affect the storage size of the image, but the quality of the displayed image may decrease as you increase or decrease the size of the displayed image.

  Generally, to achieve the best and most predictable display results, use the default setting of `Use image size`.

  - `Use image size`: Causes the image to appear the same size in the report as on screen (default).
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
- **Size**: Specifies the size of the snapshot in the form `w h` (width, height). This field is active only if you choose `Fixed size` from the **Scaling** selection list.
- **Max size**: Specifies the maximum size of the snapshot in the form `w h` (width, height). This field is active only if you choose `Zoom` from the **Scaling** selection list.
- **Units**: Specifies the units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** list box.
- **Alignment**: Only reports in `PDF` or `RTF` format support this property.
  - `Auto`
  - `Right`
  - `Left`
  - `Center`
- **Title**: Specifies a title for the snapshot figure.
  - `Block name`: Uses the block name as the title.
  - `Full Simulink path name`: Uses the Simulink path as the title.
  - `Custom`: Specifies a custom title.
- **Caption**: Select or enter a short text description for the snapshot figure.
  - `No caption`
  - `Automatic (use block description)`. Uses the Simulink block description as the caption.

- `Custom`. Specifies a short text description for the snapshot figure.

Insert Anything into Report?

Yes. Image.

Class

`rptgen_sl.csl_blk_scope`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

Signal Loop

Run child components for each signal contained in current system, model, or block

Description

The Signal Loop component runs its child components for each signal contained in the current system, model, or block. The parent component determines the behavior of this component.

- `Model Loop`: Loops on all signals in the current model.
- `System Loop`: Loops on all signals in the current system. Choose not to report on the following types of signals by clearing the corresponding option in the **Section options** area:
  - `System input signals`
  - `System output signals`
  - `System internal signals`
- `Signal Loop`: Loops on the current signal.
- `Block Loop` : Loops on all signals connected to the current block. Choose not to report on the following types of signals by clearing the corresponding option in the **Section options** area:
  - `Block input signals`
  - `Block output signals`
- If the Signal Loop does not have a looping component as its parent, it loops on all signals in all models. Choose not to report on the following types of signals by clearing the corresponding option in the **Section options** area:
  - `Block input signals`
  - `Block output signals`
  - `System input signals`
  - `System output signals`
  - `System internal signals`

For conditional processing of signals, you can use the `RptgenSL.getReportedSignal` function. For more information, see "Loop Context Functions".

Select Signals

- **Include block input signals**: Loops on signals that feed into blocks. This option is valid only when the parent component of this component is a`Block Loop`.
- **Include block output signals**: Loops on signals that leave the block. This option is valid only when the parent component of this component is a`Block Loop`.
- **Include system input signals**: Loops on signals coming from inports. This option is valid only when the parent component of this component is a`System Loop`.
- **Include system internal signals**: Loops on system internal signals. This option is valid only when the parent component of this component is a`System Loop`.
- **Include system output signals**: Loops on signals going to outports. This option is valid only when the parent component of this component is a`System Loop`.
- **Sort signals**: Specifies how to sort signals:
  - `Alphabetically by signal name`: Sorts signals alphabetically by name.
  - `Alphabetically by signal name (exclude empty)`: Sorts signals alphabetically by name.
  - `Alphabetically by system name`: Sorts alphabetically by parent system names. Lists signals in each system, but in no particular order.
  - `By signal depth`: Sorts signals by their depth in the model.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Automatically inserts the object type into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sl.csl_sig_loop`

See Also

`Block Loop`, `Model Loop`, `System Loop`

Simulink Automatic Table

Insert two-column table with information on selected model, system, signal, or block

Description

This component inserts a two-column table that contains details for the selected model, system, signal, or block into a generated report.

Options

- **Show current**: Includes a specified Simulink software feature in the generated report.
  - `Automatic`: Uses the context of the parent loop.
  - `Model`
  - `System`
  - `Block`
  - `Annotation`
  - `Signal`
- **Properties list**: Specifies whether to select properties manually or automatically.
  - `Determine properties automatically`: Uses blocks and dialog box properties.
  - `Show properties`: Enables you to specify properties manually.
- **Show full path name**: Displays the full path of the selected Simulink model.
- **Display property names as prompts**: Displays property names as prompts in the generated report. The report includes the dialog box string instead of the underlying code property.

Display Options

- **Table title**: Displays a table title in the generated report.

  - `Name`: Automatically generates a title from the parameter.
  - `Custom`: Specifies a custom title.
  - `No title`: Does not include a title.
- **Header row**: Select a header row for the table in the generated report.
  - `No header`: Includes no header row.
  - `Type and Name`: Includes a header row with columns for name and object type.
  - `Custom`: Includes a custom header.
- **Don't display empty values**: Excludes empty parameters in the generated report.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_auto_table`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

Simulink Dialog Snapshot

Insert snapshots of Simulink editor dialog boxes

Description

This component takes snapshots of Simulink editor dialog boxes. You use it to display the current settings associated with an object or document the appearance of your custom mask dialog boxes.

The parent component of this component determines the behavior of this component.

- `Block Loop`: Documents the dialog box of the current reported block.
- `System Loop`: Documents the dialog box of the current reported system.

Format

- **Image file format**: Specifies the format for the snapshot image file. The **automatic** format chooses `BMP` format for PDF files, and `PNG` for other formats.
- **Show all tabs**: Automatically generates images for all the tabs for the dialog box. If you clear this check box, Simulink Report Generator creates an image of only the first tab.

Display Options

- **Scaling**: Controls size of the image, as displayed in a browser. Making an image larger using this option does not affect the storage size of the image, but the quality of the displayed image may decrease as you increase or decrease the size of the displayed image.

  Generally, to achieve the best and most predictable display results, use the default setting of `Use image size`.

  - `Use image size`: Causes the image to appear the same size in the report as on screen (default).
  - `Fixed size`: Specifies the number and type of units.

  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
- **Size**: Specifies the size of the snapshot in the format `w h` (width, height). This field is active only if you choose `Fixed size` from the **Scaling** selection list.
- **Max size**: Specifies the maximum size of the snapshot in the format `w h` (width, height). This field is active only if you choose `Zoom` from the **Scaling** selection list.
- **Units**: Specifies the units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** list box.
- **Alignment**: Only reports in PDF or RTF format support this property.
  - `Auto`
  - `Right`
  - `Left`
  - `Center`
- **Title**: Specifies text to appear above the snapshot.
- **Caption**: Specifies text to appear under the snapshot.

Insert Anything into Report?

Yes. Snapshot.

Class

`rptgen_sl.CDialogSnapshot`

See Also

`Block Loop`, `System Loop`

Simulink Function System Loop

Report on Simulink functions specified in a Stateflow loop

Description

This component runs its child components for each Simulink function system defined by the parent component. For example, to include Simulink functions within a given model in the report, you can include this component as the child of a Stateflow `Object Loop` or `Object Loop` component, each of which in turn is usually a child of a `Chart Loop` component.

Report On

Include subsystems in nested Simulink functions: Specifies whether to include subsystems in nested Simulink functions. By default, this option is enabled.

Loop Options

- **Sort Systems**: Specifies how to sort systems.
  - `Alphabetically by system name` (default): Sorts systems alphabetically by name.
  - `By number of blocks in system`: Sorts systems by number of blocks. The list shows systems by decreasing number of blocks; that is, the system with the largest number of blocks appears first in the list.
  - `By system depth`: Sorts systems by their depth in the model.
  - `By traversal order`: Sorts systems in traversal order.
- **Search for**: Reports only on Subsystem blocks with the specified property name/property value pairs. To enable searching, click the check box. In the first row of the property name and property value table, click inside the edit box, delete the existing text, and type the property name and value.

  For information about subsystem property names and values, in “Block-Specific Parameters”, see the “Ports & Subsystems Library Block Parameters” section.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Number sections by system hierarchy**: Hierarchically numbers sections in the generated report. Requires that **Sort Systems** be set to `By traversal order`.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sl.csl_sys_loop`

See Also

`Object Loop`, `State Loop`, `Chart Loop`, `System Loop`, `Block Loop`, `Model Loop`, `Signal Loop`

Simulink Functions and Variables

Create table that displays workspace variables and MATLAB functions used by reported blocks in Simulink models

Description

This component creates a table that displays workspace variables and MATLAB functions used by blocks in a Simulink model. The `Model Loop` component specifies the current model and systems in which the blocks appear. For example, suppose a Simulink `Gain` block has a string `cos(x)` instead of a number. The Simulink software looks for a variable `x` in the workspace and uses the `cos` function.

Functions

- **Include table of functions**: Includes a table of Simulink functions in the generated report.
- **Table Title**: Specifies a title for the table in the generated report:
  - `Automatic`: Generates a title automatically from the parameter.
  - `Custom`: Specifies a custom title.
- **Parent block**: Includes a column in the table that includes the name of the block, which contains the reported variable.
- **Calling string**: Includes the MATLAB code that calls the reported variable.
- **Include fixed-point functions (sfix, ufix, ...)**: Includes Fixed-Point Designer functions in the report.

Variables

- **Include table of variables**: Includes a table of Simulink variables in the generated report.
- **Table title**: Specifies a title for the table in the generated report.
  - `Automatic`: Generates a title automatically from the parameter.
  - `Custom`: Specifies a custom title.

- **Include Workspace I/O parameters**: Reports on variables that contain parameters with time vectors and state matrices. Set these parameters in the **Workspace I/O** pane in the Simulation Parameters dialog box in a Simulink model.

  In the following table, if any of the entries in the first column are `on`, the component looks for the variable listed in the second column. If the component finds the variable in the workspace, it includes it in the report.

Parameter name	Variable name
`LoadExternalInput`	`ExternalInput`
`SaveTime`	`TimeSaveName`
`SaveState`	`StateSaveName`
`SaveOutput`	`OutputSaveName`
`LoadInitialState`	`InitialState`
`SaveFinalState`	`FinalStateName`

- **Parent block**: Includes the name of the block that contains the reported variable.
- **Calling string**: Includes the MATLAB code that calls the reported variable.
- **Size of variable**: Includes the size of the reported variable.
- **Class of variable**: Includes the variable class to which the reported variable belongs.
- **Memory size**: Includes the amount of memory in bytes that the reported variable needs.
- **Value in workspace**: Includes the value of the reported variable.

  Large arrays may appear as `[MxN CLASS]`. For example, if you have a 300-by-200 double array, it appears in the report as `[300x200 DOUBLE]`.
- **Storage class**: Include the storage class of the reported variable.

  The title of this column is **Storage Class**. This option looks at the model's `TunableVars` property to see if any of the model variables specify their storage class. If you specify the storage class, `TunableVarsStorageClass` and `TunableVarsTypeQualifier` appear in a table column in the model variables table.

  The column entries are `TunableVarsStorageClass (TunableVarsTypeQualifier)` when `TunableVarsTypeQualifier` is not empty. If `TunableVarsTypeQualifier` is empty, the column entry is `TunableVarsStorageClass`.

Values for `TunableVarsStorageClass` include:

- `Exported Global`
- `Auto`
- `ImportedExtern`
- `ImportedExtern Pointer`

- **Data object properties**: For variables that are `Simulink.Parameter` data objects, includes the values of the object properties that you list in the edit box.

Example

This table is an example of a table created by the `Model Variables` component. This Property Table reports on the variables in the `Controller` in the `F14` model.

Variable Name	Parent Blocks	Calling String	Value
Ka	`f14/Controller/Gain3`	Ka	`0.677`
Kf	`f14/Controller/Gain`	Kf	`-1.746`
Ki	`f14/Controller/Proportional plus integral compensator`	[Ki]	`-3.864`
Kq	`f14/Controller/Gain2`	Kq	`0.8156`

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_obj_fun_var`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

Simulink Library Information

Insert table that lists library links in the current model, system, or block

Description

This component inserts a table that lists library links in the current model, system, or block.

Table Columns

- **Block**: Includes the Simulink block name in the generated table.
- **Library**: Includes the Simulink library root name in the generated table.
- **Reference block**: Includes the Simulink reference block name in the generated table.
- **Link status**: Includes the link status in the generated table.

Display Options

- **Title**: Specifies a title for the generated report.
- **Sort table by**:
  - `Block`: Sorts the table by block name.
  - `Library`: Sorts the table by library name.
  - `Reference Block`: Sorts the table by reference block name.
  - `Link Status`: Sorts the table by link status.
- **Merge repeated rows**: Merges sorted rows in the generated table.

Example

The following table sorts based on `Reference Block` column. The Report Explorer uses the `aero_guidance` model with **Merge repeated rows** deselected to generate the table.

Block	Library	Reference Block	Status
Equations of Motion (Body Axes)	Aerospace	Equations of Motion (Body Axes)	resolved
Incidence & Airspeed	Aerospace	Incidence & Airspeed	resolved
Fin Actuator	Aerospace	2nd Order Nonlinear Actuator	resolved
3DoF Animation	Aerospace	3DoF Animation	resolved
Atmosphere	Aerospace	Atmosphere model	resolved
Cm	Simulink	Interpolation (n-D) using PreLookup	resolved
Cx	Simulink	Interpolation (n-D) using PreLookup	resolved
Cz	Simulink	Interpolation (n-D) using PreLookup	resolved
Kg	Simulink	Interpolation (n-D) using PreLookup	resolved
Ki	Simulink	Interpolation (n-D) using PreLookup	resolved
Alpha Index	Simulink	PreLookup Index Search	resolved
Mach Index	Simulink	PreLookup Index Search	resolved
Mach Index	Simulink	PreLookup Index Search	resolved
|Alpha| Index	Simulink	PreLookup Index Search	resolved

When you select **Merge repeated rows**, the generated table collapses rows in the `Block` column. Each row in the `Reference Block` column is unique, as shown in the following table.

Block	Library	Reference Block	Status
Equations of Motion (Body Axes)	Aerospace	Equations of Motion (Body Axes)	resolved

Block	Library	Reference Block	Status
Incidence & Airspeed	Aerospace	Incidence & Airspeed	resolved
Fin Actuator	Aerospace	2nd Order Nonlinear Actuator	resolved
3DoF Animation	Aerospace	3DoF Animation	resolved
Atmosphere	Aerospace	Atmosphere model	resolved
Cm Cx Cz Kg Ki	Simulink	Interpolation (n-D) using PreLookup	resolved
Alpha Index Mach Index Mach Index |Alpha| Index	Simulink	PreLookup Index Search	resolved

Insert Anything Into Report?

Yes. Table.

Class

`rptgen_sl.CLibinfo`

See Also

`Block Loop`, `Model Loop`, `System Loop`

Simulink Linking Anchor

Designate locations to which links point

Description

This component designates a location to which links point. Use the `Model Loop`, `System Loop`, `Block Loop`, or `Signal Loop` component as the parent component for this component.

Properties

- **Insert text**: Specifies text to appear after the linking anchor.
- **Link from current**: Sets the current model, system, block, or signal as the linking anchor.
  - `Automatic`: Automatically selects the appropriate model, system, block, or signal as a linking anchor. If the `Model Loop` component is the parent component, the linking anchor is set on the current model. Similarly, if the `Block Loop` or `Signal Loop` is the parent component, the linking anchor is inserted for the current system, block, or signal, respectively.
  - `Model`: Sets the linking anchor to the current model.
  - `System`: Sets the linking anchor to the current system.
  - `Block`: Sets the linking anchor to the current block.
  - `Annotation`: Sets the linking anchor to the current annotation.
  - `Signal`: Sets the linking anchor to the current signal.

Note: Use only one anchor per report each object. For more information, see the `Simulink Summary Table` component reference page.

Insert Anything into Report?

Yes. A link, and possibly text, depending on attribute choices.

Class

`rptgen_sl.csl_obj_anchor`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

Simulink Name

Insert name of a Simulink model, system, block, or signal into report

Description

This component inserts the name of a Simulink model, system, block, or signal into the report.

Using this component as the first child component of a `Chapter/Subsection` component allows the current Simulink model, system block, or signal name to be the chapter or section title.

Properties

- **Object type**
  - `Automatic`: Automatically selects the appropriate model, system, block, or signal name as the Simulink object name to include in the report. If the Model Loop component is the parent component, the object name is the current model name. If the System Loop, Block Loop, or Signal Loop is the parent component, then the object name is the name of the current system, block, or signal, respectively.
  - `Model`: Includes the current model name in the report.
  - `System`: Includes the current system name in the report.
  - `Block`: Includes the current block name in the report.
  - `Signal`: Includes the current signal name in the report. If the signal name is empty, the signal `<handle>`, which is a unique numerical identifier to that signal, appears in the report.
  - `Annotation`: Includes the current annotation name in the report.
- **Display name as**: Display the Simulink object name in the report.
  - `Name`: For example, `f14`
  - `Type Name`: For example, `Model f14`
  - `Type - Name`: For example, `Model - f14`
  - `Type: Name`: For example, `Model: f14`

- **Show full path name**: Displays the full path of a system or block. Choosing this option for a block causes the Simulink block name to appear as <Model Name>/<System Name>/<Block Name>.

Note: This option is not available for models or signals.

Insert Anything into Report?

Yes. Text.

Class

`rptgen_sl.csl_obj_name`

See Also

`Chapter/Subsection`

Simulink Property

Insert property name/property value pair for current Simulink model, system, block, or signal

Description

This component inserts a single property name/property value pair for the current Simulink model, system, block, or signal.

Simulink Object and Parameter

- **Object type**: Specifies the Simulink object type to include in the report.
  - `System`
  - `Model`
  - `Block`
  - `Signal`
  - `Annotation`
- **System parameter name**: Specifies a Simulink parameter name to include in the generated report:
  - If you select `Model` for **Object type**, this option appears as **Model parameter name**.
  - If you select `Block` for **Object type**, this option appears as **Block parameter name**.
  - If you select `Signal` for **Object type**, this option appears as **Signal parameter name**.

Display Options

- **Title**: Choose a title to display in the generated report:
  - `Automatic`: Uses the parameter name as the title.

  - `Custom`: Specifies a custom title.
  - `None`: Uses no title.
- **Size limit**: Limits the width of the display in the generated report. Units are in pixels. The size limit for a given table is equal to the hypotenuse of the table width and height [`sqrt(w^2+h^2)`]. The size limit for text is the number of characters squared. If you exceed the size limit, the variable appears in condensed form. Setting a size limit of zero always causes the variable to display, regardless of its size
- **Display as**: Specifies a display style.
  - `Auto table/paragraph`: Displays as a table or paragraph based on the information.
  - `Table`: Displays as a table.
  - `Paragraph`: Displays as a text paragraph.
  - `Inline text`: Displays as inline, which fits in line with the surrounding text.
- **Ignore if value is empty**: If the parameter is empty, the report excludes the parameter from the generated report.

Insert Anything into Report?

Yes. Text.

Class

`rptgen_sl.csl_property`

See Also

`Block Loop`, `Model Loop`, `System Loop`

Simulink Property Table

Insert table that reports on model-level property name/property value pairs

Description

This component inserts a table that reports on model-level property name/property value pairs.

Properties

Select Object: Choose the object for the Property Table in the generated report.

- `Model`
- `System`
- `Block`
- `Signal`
- `Annotation`

For more information about selecting object types in Property Table components, see "Select Object Types".

Table

Select a preset table, which is already formatted and set up, in the preset table list in the upper-left corner of the attributes page.

- **Preset table**: Specifies the type of the object property table.
  - `Default`
  - `Simulation parameters`
  - `Version information`
  - `Simulink Coder information`

- `Summary (req. Simulink Coder)`
- `Blank 4x4`

  To apply a preset table, select the table and click **Apply**.

- **Split property/value cells**: Split property name/property value pairs into separate cells. For the property name and property value to appear in adjacent horizontal cells, select the **Split property/value cells** check box. In this case, the table is in split mode, so there is only one property name/property value pair per cell. If there is more than one name/property pair in a cell, only the first pair appears in the report. All subsequent pairs are ignored.

  For the property name and property value to appear together in one cell, clear the **Split property/value cells** check box. That setting specifies nonsplit mode. Nonsplit mode supports more than one property name/property value pair and text per cell.

  Before switching from nonsplit mode to split mode, make sure that there is only one property name/property value pair per table cell. If there is more than one property name/property value pair or text per cell, only the first property name/property value pair appears in the report. The report omits subsequent pairs and text.
- **Display outer border**: Displays the outer border of the table in the generated report.
- **Table Cells**: Specifies table properties to modify. The selection in this pane affects available fields in the **Cell Properties** pane.

Cell Properties

The options in this pane depend on the object selected in the **Table Cells** pane. If you select `%<Name> Information`, only **Contents** and **Show** appear. If you select any other object in the **Table Cells** pane, **Lower border** and **Right border** display.

- **Contents**: Enables you to change the contents of the table cell selected in the **Table Cells** pane.
- **Show as**: Specifies the format for the contents of the table cell.

  - `Value`
  - `Property Value`
  - `PROPERTY Value`

  - `Property: Value`
  - `PROPERTY: Value`
  - `Property - Value`
  - `PROPERTY - Value`
- **Alignment**: Specifies the alignment of the contents of the selected table cell in the **Table Cells** pane.
  - `Left`
  - `Center`
  - `Right`
  - `Double justified`
- **Lower border**: Displays the lower border of the table in the generated report.
- **Right border**: Displays the right border of the table in the generated report.

Creating Custom Tables

To create a custom table, edit a preset table, such as the `Blank 4x4` table. Add and delete rows and add properties. To open the Edit Table dialog box, click **Edit**.

For more information on creating custom property tables, see “Property Table Components”.

If the Simulink Coder software is not installed, `Summary (req Simulink Coder)` does not appear in this list. If you are using a report setup file that contains a summary property, the property name appears in the report, but the property value does not.

Example

The following report displays information on the `F14` model using the **Simulation Parameters** preset table.

Solver ode45	*ZeroCross* on	*StartTime* 0.0 *StopTime* 60
RelTol 1e-4	*AbsTol* 1e-6	*Refine* 1
InitialStep auto	*FixedStep* auto	*MaxStep* auto
LimitMaxRows off	*MaxRows* 1000	*Decimation* 1

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_prop_table`

See Also

`Model Loop`, `Signal Loop`, `System Loop`

Simulink Sample Time

Insert title of Simulink sample time into report

Description

This component inserts a title for a Simulink sample time into the report.

Properties

- **Table Options**
  - **Title**: Specifies a title for the table in the generated report.
  - **Grid lines**: Show grid lines for the table.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.CSampleTime`

See Also

`Chapter/Subsection`

Simulink Summary Table

Properties or parameters of specified Simulink models, systems, blocks, or signals in table

Description

This component displays properties or parameters of selected Simulink models, systems, blocks, or signals in a table.

Object type

Choose the object type to display in the generated report.

- `Block` (Default)
- `Model`
- `System`
- `Signal`
- `Annotation`

The selected object type affects the options available in the **Property Columns** pane.

Table title

Choose a title to appear in the generated report:

- `Automatic`: Automatically generates a title from the parameter.
- `Custom`: Specifies a custom title.

Property Columns

This pane displays object properties to include in the Summary Table in the generated report.

- To add a property:

1. select the appropriate property level in the text box on the left.
2. In the text box on the right, select the property that you want to add and click **Add**.

- To delete a property, select the property name and click **Delete**.

`%<SplitDialogParameters>` is a unique property for Simulink Summary Tables, where the object type is `Block`. This property generates multiple summary tables, organized by block type. Each Summary Table group contains the dialog box parameters for that block.

Some entries in the list of available properties (such as `Depth`) are “virtual” properties that you cannot access using the `get_param` command. The properties used for property/value filtering in the block and System Loop components must be retrievable by the `get_param`. Therefore, you cannot configure your Summary Table to report on all blocks of `Depth` == 2.

You can create multiple values for a property in a Simulink Summary Table. For example, to report on blocks of type `Inport`, `Outport` and `Constant`:

1. Check the **Search for Simulink property name/property value pairs** box.
2. Make sure that you set **Property Name** to `BlockType`.
3. Type the following text into the **Property Value** field:

```
\<(Inport|Outport|Constant)\>
```

Remove empty columns: Removes empty columns from the table.

Transpose table: Changes the summary table rows into columns in the generated report, putting the property names in the first column and the values in the other columns.

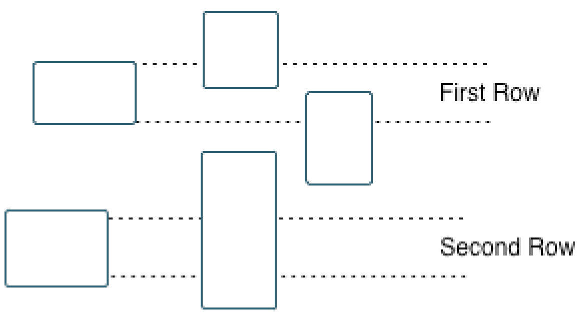
Object Rows

- **Insert anchor for each row**: Inserts an anchor for each row in the summary table.
- **Report On**:
  - `Automatic list from context`: Reports on all blocks in the current context, as set by the parent component.

- `Custom - use block list`: Reports on a list of specified blocks. Specify the full path of each block.

Loop Options

- **Sort blocks**
  - `Alphabetically by block name`: Sorts blocks alphabetically by name.
  - `Alphabetically by system name`: Sorts systems alphabetically by name. Lists blocks in each system, but in no particular order.
  - `Alphabetically by full Simulink path`: Sorts blocks alphabetically by Simulink path.
  - `By block type`: Sorts blocks alphabetically by block type.
  - `By block depth`: Sorts blocks by their depth in the model.
  - `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



  - `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.

- `By traversal order`: Sorts blocks by traversal order.
- `By simulation order`: Sorts blocks by execution order.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. The `%<>` notation can denote a string or cell array. For more information, see `%<VariableName> Notation` on the `Text` component reference page.

- **Search for Simulink property name/property value pairs**: Reports on blocks with specified property name/property value pairs.

Example

Specify the following options to generate a Summary Table in a report for on the model `F14`:

- Sort on systems by system depth.
- Include the system parameters `Name` and `Block` in the table.

The following table appears in the report.

Name	Blocks
f14	u, Actuator Model, Aircraft Dynamics Model, Angle of Attack, Controller, Dryden Wind Gust Models, Gain, Gain1, Gain2, Gain5, More Info, More Info1, Nz pilot calculation, Pilot, Pilot G force Scope, Stick Input, Sum, Sum1, alpha (rad), Nz Pilot (g)
Aircraft Dynamics Model	Elevator Deflection d (deg), Vertical Gust wGust (ft/sec), Rotary Gust qGust (rad/sec), Gain3, Gain4, Gain5, Gain6, Sum1, Sum2, Transfer Fcn.1, Transfer Fcn.2, Vertical Velocity w (ft/s), Pitch Rate q (rad/s)
Controller	Stick Input (in), alpha (rad), q (rad/s), Alpha-sensor Low-pass Filter, Gain, Gain2, Gain3, Pitch Rate Lead Filter, Proportional plus integral compensator, Stick Prefilter, Sum, Sum1, Sum2, Elevator Command (deg)
Dryden Wind Gust Models	Band-Limited White Noise, Q-gust model, W-gust model, Wg, Qg
More Info	None

Name	Blocks
More Info1	None
Nz pilot calculation	w, q, Constant, Derivative, Derivative1, Gain1, Gain2, Product, Sum1, Pilot g force (g)

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sl.csl_summ_table`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System LoopSimulink Function System Loop`

Simulink Workspace Variable

Report information about MATLAB and model workspace variables that model uses

Description

This component provides information about variables in the MATLAB (base) workspace and the model workspaces that a model uses. The report includes the value of each variable. Optionally, you can include the following information for each variable:

- Workspace where the variable is located
- Blocks that use the variable

For variables that are Simulink data objects (for example, a `Simulink.Parameter` object), the report includes the properties of the object. You can filter out properties to streamline the report.

Options

The following two options specify additional information that the report can include about each variable:

- **Show workspace**: Report whether each variable is in the MATLAB workspace or in a model workspace.
- **Show blocks that use variable**: Report the blocks that use each variable.

For variables whose values are Simulink data objects, you can filter which properties to include in the report, using one of the following approaches:

- Use the **Filter Properties** area of the dialog box to specify a standard filter.

  The standard filter options apply to all variables whose values are instances of the class or classes that you specify. For example, you can use a standard filter to filter out the `Description` property for all variables used by the model whose values use a `Simulink.Parameter` object.
- Select the **Use custom property filter** option and write MATLAB code for filtering.

Writing custom filtering code allows you to do kinds of filtering that the standard filter does not perform. Some common examples of custom filters that you might want to create are filters that filter out:

- A property for some, but not all, instances of a class
- Properties that match a regular expression

The **Filter Properties** area of the dialog box, where you specify a standard filter, has these fields:

- **Class name (* for all classes)**: Specify the class of the variables for which you want to filter out specific properties. You can specify one class at a time, or enter an asterisk (*) to specify all classes. After you enter the class name, move the cursor outside of the edit box.
- **Available Properties**: If the class that you entered in **Class name (* for all classes)** is on the MATLAB path, then this list displays the properties of that class.
- **Filtered Properties**: Displays the properties to filter out. Use the right-arrow button to add to the **Filtered Properties** list the properties that you selected in the **Available Properties** list.
- If the class that you enter is *not* on the MATLAB path, then a **Comma-separated list of properties to be filtered** edit box appears. Enter the names of properties to use for filtering.
- **Convert to Custom**: Generate custom MATLAB code that implements your **Filter Properties** standard filter settings.

Note: Selecting the **Convert to Custom** button overwrites any existing MATLAB custom filtering code for this component.

To create and apply custom filtering MATLAB code, select the **Use custom property filter** check box. Selecting this check box opens an edit box where you define a MATLAB function for filtering properties. The edit box includes a sample function (commented out) that you can use as a starting point for your filtering function. Use the `isFiltered` variable for the output of your function. For example:

- To filter out the `Owner` and `testProp` properties, in the edit box enter:

```
isFiltered = strcmpi(propertyName, 'Owner')||...
strcmpi(propertyName, 'testProp');
```

- To filter out all properties *except* for the `CoderInfo` property, in the edit box, enter:

  ```
  isFiltered = ~strcmpi(propertyName, 'CoderInfo');
  ```

If you clear the **Use custom property filter** check box, Simulink Report Generator saves your custom MATLAB filtering code, but does not use that code to filter properties.

Insert Anything into Report?

Yes. List.

Class

`rptgen_sl.csl_ws_variable`

See Also

`Simulink Workspace Variable Loop`, `Bus`, `Simulink Functions and Variables`

Software Quality Objectives - Coding Rules Summary

Create table of coding rule violations in results downloaded from Polyspace Metrics

Description

This component creates a table containing coding rule violations in results downloaded from Polyspace Metrics.

Software Quality Objectives - Run-time Checks Details

Create table of run-time checks in results downloaded from Polyspace Metrics

Description

This component creates a table containing run-time checks in results downloaded from Polyspace Metrics.

Software Quality Objectives - Run-time Checks Summary

Create table of run-time check distribution in results downloaded from Polyspace Metrics

Description

This component creates a table containing the distribution of run-time checks in results downloaded from Polyspace Metrics.

Simulink Workspace Variable Loop

Reports on Simulink workspace variable objects

Description

This component displays all variables in the Simulink workspace (base workspace and model workspaces). You can limit the variables displayed to those variables that match property name and property value pairs that you specify.

Loop Options

- **Sort**
  - `Alphabetically by text`: Sorts variables alphabetically by name.
  - `By data type`: Sorts variables alphabetically by data type.
- **Search for Simulink property name/property value pairs**: Reports on variables with specified property name/property value pairs.

Section Options

- **Create section for each object in loop**: Creates a separate section in the output for each variable.
  - If you specify to create a section for each variable, you can select the **Display the object type in the section title** to insert a variable name in each section title.
- **Create link anchor for each object in loop**: Specifies a custom title.

Insert Anything into Report?

Yes, inserts sections if you select the **Create section for each object in loop** option

Class

`rptgen_sl.csl_ws_var_loop`

See Also

`Simulink Workspace Variable`, `Bus`, `Simulink Functions and Variables`

State Loop

Run child components for all states in current context

Description

This component runs its children for all states in its context. The parent component of this component determines the context.

- `Model Loop`: Includes all states in the models.
- `System Loop`: Includes all states in the systems.
- `Machine Loop`: Includes all states in the machines.
- `Chart Loop`: Includes all states in the charts.
- `State Loop`: Includes all states in the current state.

For conditional processing based on states, you can use the `RptgenSF.getReportedState` function. For more information, see “Loop Context Functions”.

State Types

- **Include “and” and “or” states**: Includes `AND` and `OR` states in the loop.
- **Include “box” states**: Includes “box” states in the loop.
- **Include functions**: Includes “function” states in the loop.
- **Include truth tables**: Includes truth tables in the loop.
- **Include MATLAB functions**: Includes MATLAB functions in the loop.

Loop Options

- **Report depth**: Specifies the level on which to loop.
  - `Local children only`
  - `All objects`

- **Skip autogenerated charts under truth table**: Keeps autogenerated state objects under truth tables from appearing in the report.
- **Search Stateflow**: Indicates specific states to include in the loop.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

Yes, section, if you select the **Create section for each object in loop** option.

Class

`rptgen_sf.csf_state_loop`

See Also

`Chart Loop`, `Machine Loop`, `Model Loop`, `State Loop`, `System Loop`, `Simulink Function System Loop`

Stateflow Automatic Table

Insert table with properties of current Stateflow object

Description

The Stateflow Automatic Table component inserts a table that contains the properties of the current Stateflow object. Parents of this component can be:

- `Machine Loop`
- `State Loop`
- `Chart Loop`
- `Graphics Object Loop`

Display Options

- **Table title**: Specifies a title for the table in the generated report.
  - `No title`: Includes no title.
  - `Custom`: Includes a custom title.
  - `Name` (default): Uses an object name as the title.
    - `Object name`
    - `Object name with Stateflow path`
    - `Object name with Simulink and Stateflow path`
- **Header row**: Selects a header row for the table in the generated report.
  - `No header`: Includes no header row.
  - `Type and Name`: Includes a header row with columns for name and object type. When selected, this option creates a header row for the table with object name and type.
  - `Custom`: Includes a custom header.
- **Don't display empty values**: Excludes empty values from the generated report.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sf.csf_auto_table`

See Also

`Chart Loop`, `Graphics Object Loop`, `Machine Loop`, `State Loop`

Stateflow Count

Count number of Stateflow objects in current context

Description

This component counts the number of Stateflow objects in the current context.

Properties

- **Search depth**: Specifies the search depth for the count.
  - `Immediate children only` (default): Searches only children one level under the Stateflow object.
  - `All descendants`: Searches all children of the Stateflow object.
- **Sort results**: Specifies the sort method for the count results.
  - `Numerically decreasing by object count` (Default)
  - `Alphabetically increasing by object type`
- **Include a list of objects in table**: Inserts a column containing the counted objects.
- **Show total count**: Displays a total of counted objects.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sf.csf_count`

See Also

`State Loop`

Stateflow Dialog Snapshot

Insert snapshots of Stateflow editor dialog boxes

Description

This component reports on the current reported Stateflow dialog box object, depending on its context. If this component is the child of a `State Loop`, for example, the report includes information about the dialog box of the current State. Display the current settings associated with an object or document the appearance of your custom mask dialog boxes.

Format

- **Image file format**: Specifies the format for the snapshot image file. The `Automatic` format uses `BMP` format for PDF files and `PNG` for other formats.
- **Show all tabs**: Automatically generates images for all the tabs for the dialog box. If you clear this check box, the Simulink Report Generator software creates an image of only the first tab.

Display Options

- **Scaling**: Controls size of the image, as displayed in a browser. Making an image larger using this option does not affect the storage size of the image, but the quality of the displayed image may decrease as you increase or decrease the size of the displayed image.

  Generally, to achieve the best and most predictable display results, use the default setting of `Use image size`.

  - `Use image size`: Causes the image to appear the same size in the report as on screen (default).
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
- **Size**: Specifies the size of the snapshot in the form `w h` (width, height). This field is active only if you choose `Fixed size` in the **Scaling** selection list.

- **Max size**: Specifies the maximum size of the snapshot in the form `w h` (width, height). This field is active only if you choose `Zoom` in the **Scaling** selection list.
- **Units**: Specifies the units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** list box.
- **Alignment**: Aligns your snapshot. Only reports in `PDF` or `RTF` format support this property.
  - `Auto`
  - `Right`
  - `Center`
  - `Left`
- **Title**: Specifies text to appear above the snapshot.
- **Caption**: Specifies text to appear under the snapshot.

Insert Anything into Report?

Yes. Snapshot.

Class

`rptgen_sl.Cdialog boxesnapshot`

See Also

`State Loop`

Stateflow Filter

Run child components only if current object type matches specified object type

Description

This component runs its children only if the current object type, as set by its parent `Stateflow Hierarchy Loop`, matches the selected object type.

Properties

- **Object type**: Specifies the Stateflow object type to include in the report.
- **Run only if Stateflow object has at least the following number of Stateflow children**: Specifies a minimum number of children that a Stateflow object must have to include in the report.
- **Automatically insert linking anchor**: Inserts a linking anchor before the reported object. If an anchor for this object exists, this option does not create a second anchor.

Insert Anything into Report?

No.

Class

`rptgen_sf.csf_obj_filter`

See Also

`Stateflow Hierarchy Loop`

Stateflow Hierarchy

Provide visual representation of the hierarchy of a Stateflow object

Description

This component inserts a tree that shows the hierarchy of a given Stateflow object.

Tree Options

- **Construct tree from**: Specifies the object to use for the tree representation.
  - `Current object`
  - `Root of current object`: Starts reporting from the top of the hierarchy.
- **Emphasize current object in tree**: Highlights the current object in the tree representation.
- **Show number of parents**: Specifies the number of parents to include in the tree representation.
- **Show siblings**: Displays siblings in the tree representation.
- **Show children to depth**: Specifies the depth of children to display for each object in the tree representation.

Children

- **Show junctions**: Specifies the level of junction detail to display in the generated report.
  - `All`
  - `Non-redundant`
  - `None`
- **Show transitions**: Specifies the level of transition detail to display in the generated report.
  - `All`
  - `Labeled or non-redundant`

    - `Non-redundant`
    - `Labeled`
    - `None`
- **Skip autogenerated charts under truth tables**: Excludes autogenerated charts under truth tables.

List Formatting

- **List style**:
    - `Bulleted list`
    - `Numbered list`: Allows you to specify numbering options in the **Numbering style** section.
        - **Numbering style**: Allows you to specify a numbering style. This setting supports only the `RTF/DOC` report format.
            - `1,2,3,4...`
            - `a,b,c,d...`
            - `A,B,C,D...`
            - `i,ii,iii,iv...`
            - `I,II,III,IV...`

          To show the parent number in each list entry, select `Show parent number in nested list (1.1.a)`. To show only the current number or letter, select `Show only current list value (a)`.

Insert Anything into Report?

Yes. Tree graphic.

Class

`rptgen_sf.csf_hier`

See Also

`Stateflow Hierarchy Loop`

Stateflow Hierarchy Loop

Run child components on Stateflow object hierarchy

Description

This component runs its child components on the Stateflow object hierarchy.

Loop Options

- **Minimum legible font size**: Specifies the minimum font size to use in the report. The default font size, 8, is the smallest recommended font size.
- **Skip autogenerated charts under truth tables**; Excludes autogenerated charts under truth tables in the report.
- **Search Stateflow**: Reports on Stateflow charts with specified property name/property value pairs.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Insert Anything into Report?

No.

Class

`rptgen_sf.csf_hier_loop`

See Also

`Stateflow Hierarchy`

Stateflow Linking Anchor

Designate locations to which links point

Description

This component designates a location to which other links point. The linking anchor is set to the current object, as defined by the parent component.

This component must have the `Chart Loop`, `State Loop`, `Machine Loop`, or `Stateflow Filter` component as its parent.

Properties

Insert text: Specifies text to appear after the linking anchor.

Insert Anything into Report?

Yes. A link, and possibly text, depending on attribute choices.

Class

`rptgen_sf.csf_obj_anchor`

See Also

`Chart Loop`, `Machine Loop`, `State Loop`, `Stateflow Filter`,

Stateflow Name

Insert into report name of Stateflow object specified by parent component

Description

This component inserts the name of the Stateflow object, as defined by its parent component, into the report. This component must have the State Loop, Chart Loop, or Stateflow Filter component as its parent.

Using this component as the first child component of a Chapter/Subsection component allows the current Stateflow object name to be the chapter or section title.

Properties

- **Display name as**: Displays the Stateflow object name in the report.
  - Name: For example, Object
  - Type Name: For example, Object <ObjectName>
  - Type - Name: For example, Object - <ObjectName>
  - Type: Name: For example, Object: <ObjectName>
- **Display name as**: Specifies the level of detail with which the Stateflow object name displays the report.
  - Object name
  - Object name with Stateflow path
  - Object name with Simulink and Stateflow path

Insert Anything into Report?

Yes. Text.

Class

rptgen_sf.csf_obj_name

See Also

`Chapter/Subsection`, `Chart Loop`, `State Loop`, `Stateflow Filter`

Stateflow Property

Insert into report table, text, or paragraph with information on selected Stateflow object property

Description

This component inserts a table, text, or paragraph that contains details of the selected Stateflow object property.

Property to Display

Property name: Specifies the Stateflow property name to display.

Display Options

- **Title**: Specifies a title to display in the generated report.
  - `Automatic`: Uses the parameter name as the title.
  - `Custom`: Specifies a custom title.
  - `None`: Specifies no title.
- **Size limit**: Specifies the width of the display in the generated report. Units are in pixels. The size limit for a given table is the hypotenuse of the width and height of the table, `sqrt(w^2+h^2)`. The size limit for text is the number of characters squared. If you exceed the size limit, the variable appears in condensed form.

  Setting a size limit of `0` always displays the variable in long form, regardless of its size.
- **Display as**: Specifies a display style from the menu.
  - `Auto table/paragraph` (default): Displays as a table or paragraph based on the information.
  - `Table`: Displays as a table.
  - `Paragraph`: Displays as a text paragraph.

  - `Inline text`: Displays in line with the surrounding text.
- **Ignore if value is empty**: Excludes empty parameters from the generated report.

Insert Anything into Report?

Yes. Text, paragraph, or table.

Class

`rptgen_sf.csf_property`

See Also

`Paragraph`, `Table`, `Text`, `Stateflow Name`

Stateflow Property Table

Insert into report property-value table for Stateflow object

Description

This component inserts a property-value table for a Stateflow object into the report. Use the `Stateflow Filter` component as the parent of this component.

For more information on working with Property Table components, see "Property Table Components".

Table

Select a preset table, which is already formatted and set up, in the preset table list in the upper-left corner of the attributes page.

- **preset table**: Specifies a type of table to display the object property table.
  - `Default`
  - `Machine`
  - `Chart`
  - `State`
  - `Truth table`
  - `EM function`
  - `Data`
  - `Event`
  - `Junction`

  To apply a preset table, select the table and click **Apply**.
- **Split property/value cells**: Splits property name/property value pairs into separate cells.
  - For the property name and property value to appear in adjacent horizontal cells, select the **Split property/value cells** check box. In this case, the table is in split

mode, there is only one property name/property value pair per cell. If there is more than one name/property pair in a cell, only the first pair appears in the report. The report ignores all subsequent pairs.

  - For the property name and property value to appear together in one cell, clear the **Split property/value cells** check box. This setting is nonsplit mode. Nonsplit mode supports more than one property name/property value pair and text.
  - Before switching from nonsplit mode to split mode, make sure that there is only one property name/property value pair per table cell. When there is more than one property name/property value pair or any text in a given cell, only the first property name/property value pair appears in the report. The report omits subsequent pairs and text.
- **Display outer border**: Displays the outer border of the table in the generated report.
- **Table Cells**: Specifies table properties to modify. The selection in this pane affects the available fields in the **Cell Properties** pane.

Cell Properties

The options in the **Title Properties** pane depend on the object selected in the **Table Cells** pane. If you select `%<Name>`, only **Contents** and **Show** appear. If you select any other object in the **Table Cells** pane, **Lower border** and **Right border** appear.

- **Contents**: Modifies the contents of the table cell selected in the **Table Cells** pane.
- **Alignment**: Justifies the contents of the selected table cell in the **Table Cells** pane.
  - `Left`
  - `Center`
  - `Right`
  - `Double justified`
- **Show As**: Specifies the format for the contents of the table cell.
  - `Value`
  - `Property Value`
  - `PROPERTY Value`
  - `Property: Value`

- `PROPERTY: Value`
- `Property - Value`
- `PROPERTY - Value`
- **Lower border**: Displays the lower border of the table in the generated report.
- **Right border**: Displays the right border of the table in the generated report.

Creating Custom Tables

You can edit a preset table, such as the `Blank 4x4` table, to create a custom table. Add and delete rows and add properties. To open the Edit Table dialog box, click **Edit**.

For details about creating custom property tables, see “Property Table Components”.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sf.csf_prop_table`

See Also

`Stateflow Filter`

Stateflow Snapshot

Insert into report snapshot of Stateflow object

Description

This component inserts a snapshot of a Stateflow object, defined by the `Stateflow Filter` parent component, into the report.

This component only executes if the selected object in the `Stateflow Filter` component is a graphical object, such as `Chart`, `State`, `Transition`, and `Frame`.

Snapshot

- **Image file format**: Specifies the image file format (for example, `JPEG` or `TIFF`). Select `Automatic SF Format` (default) to choose the format best suited for the specified report output format automatically. Otherwise, choose an image format that your output viewer can read.
  - `Automatic SF Format` (uses the file format selected in the Preferences dialog box)
  - `Bitmap (16m-color)`
  - `Bitmap (256-color)`
  - `Black and white encapsulated PostScript`
  - `Black and white encapsulated PostScript (TIFF)`
  - `Black and white encapsulated PostScript2`
  - `Black and white encapsulated PostScript2 (TIFF)`
  - `Black and white PostScript`
  - `Black and white PostScript2`
  - `Color encapsulated PostScript`
  - `Color encapsulated PostScript (TIFF)`
  - `Color encapsulated PostScript2`
  - `Color encapsulated PostScript2 (TIFF)`

   - `Color PostScript`
   - `Color PostScript2`
   - `JPEG high quality image`
   - `JPEG medium quality image`
   - `JPEG low quality image`
   - `PNG 24-bit image`
   - `Scalable vector graphics (SVG)`
   - `TIFF - compressed`
   - `TIFF - uncompressed`
   - `Windows metafile`
- **Paper orientation**:
   - `Portrait`
   - `Landscape`
   - `Rotated`
   - `Largest dimension vertical`: Positions the image so that its largest dimension is vertical.
   - `Use Chart PaperOrientation setting`: Uses the paper orientation setting for the chart. Use the Simulink `PaperOrientation` parameter to specify the orientation.
   - `Full page image (PDF only)`: In PDF reports, scales images to fit the full page, minimizes page margins, and maximizes the size of the image by using either a portrait or landscape orientation.

   For more information about paper orientation, see the `orient` command in the MATLAB documentation.
- **Image sizing**:
   - `Shrink image to minimum font size specified in Stateflow Hierarchy Loop`: Resizes the image so that the text label font size is the minimum font size.
   - `Fixed` and `Zoom`: Specifies the size of the image.
- **Scaling**: Specifies the percentage of the image size to which to scale it.

- **Maximum size**: Specifies the maximum size for the snapshot in the generated report in the selected units. Use `[width, height]` format. In the units text box, select `Inches`, `Centimeters`, `Points`, or `Normalized`.
- **Use printframe**: Inserts a frame around your image. Use the default frame or create a custom one.
- **Use printframe paper settings**: Uses the dimensions and parameters as set by the specified **printframe** to size your image. If you choose this option, all other options (except for **Image file format**) become inactive.

Properties

- **Include callouts to describe visible objects**: Displays descriptive callouts for visible objects.
- **Insert anchors for transitions and junctions**: Inserts anchors for transitions and junctions into the report.
  - `None`
  - `Redundant children only`
  - `All`
- **Run only if Stateflow object has at least the following number of children**: Specifies the minimum number of children that the current Stateflow object must have to include in the report. This option is inactive unless the selected object in the parent `Stateflow Filter` component is a graphical object.

  Tip This option allows you to exclude certain images to decrease the size of the report for large models.

Display Options

- **Scaling**:
  - `Use image size`: Uses the image size that you specify in the snapshot option.
  - `Zoom` and `Fixed size`: Allows you to specify the size of the image.
- **Size**: Specifies a size in inches for your image. The default is `7`-by-`9`.

- **Max size**: Specifies the maximum size of the snapshot in the format `w h` (width, height). This field is active only if you choose `Zoom` from the **Scaling** selection list.
- **Units**: Specifies the units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** list box.
- **Alignment**: Only reports in `PDF` or `RTF` format support this property.
  - `Auto`
  - `Right`
  - `Center`
  - `Left`
- **Image title**:
  - `None`(Default).
  - `Object name`: Uses the object name as the title.
  - `Full Stateflow name`: Specifies the Stateflow path and the name of the object.
  - `Full Simulink + Stateflow name`: Specifies the Simulink path and name of the object.
  - `Custom`: Enter a different title.
- **Caption**: Specifies a caption for your image.
  - `None`(Default).
  - `Custom`: Specifies a custom caption.
  - `Description`: Sets the caption to the value of the object `Description` property.

Insert Anything into Report?

Yes. Image.

Class

`rptgen_sf.csf_obj_snap`

Class

`rptgen_sf.csf_prop_table`

See Also

`Stateflow Filter`

Stateflow Summary Table

Table of properties or parameters of specified Stateflow object

Description

This component displays a table of properties or parameters of specified Stateflow objects. It can have the following parents:

- Any Stateflow looping component
- Any Simulink looping component (`Model Loop`, `System Loop`, `Block Loop`, or `Signal Loop`)

Properties

- **Object type**: Specifies the object type to display in the generated report. This value affects the options available in the **Property Columns** pane.
- **Table title**: Specifies a title for the Summary Table in the generated report.
  - `Automatic`: Generates a title automatically from the parameter.
  - `Custom`: Specifies a custom title.

Property Columns

- **Property columns**: Displays the object properties to include in the Summary Table in the generated report.
  - To add a property:
    - Select the appropriate property level in the text box.
    - In the context list under the text box, select the property that you want to add and click **Add**.
  - To delete a property, select the property name and press the **Delete** key.

  Some entries in the list of available properties (such as `Depth`) are "virtual" properties that you cannot access using the `get_param` command. The properties used for property/value filtering in the block and System Loop components must be retrievable by the `get_param`. Therefore, you cannot configure your Summary Table to report on all blocks of `Depth` == 2.

- **Remove empty columns**: Removes empty columns from the Summary Table in the generated report.
- **Transpose table**: Changes the summary table rows into columns in the generated report, putting the property names in the first column and the values in the other columns.

Object Rows

Insert anchor for each row: Inserts an anchor for each row in the summary table.

Report On

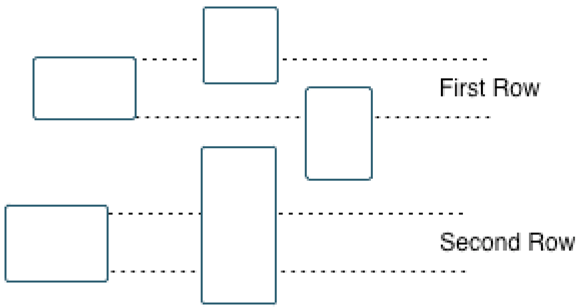
- **Automatic list from context**: Reports on all blocks in the current context, as set by the parent component.
- **Custom - use block list**: Reports on a specified list of blocks. Specify the full path of each block.

Loop Options

Choose block sorting options and reporting options in this pane.

- **Sort blocks**: Specifies how to sort blocks (applied to each level in a model):
  - `Alphabetically by block name`: Sorts blocks alphabetically by name.
  - `Alphabetically by system name`: Sorts systems alphabetically by name. Lists blocks in each system, but in no particular order.
  - `Alphabetically by full Simulink path`: Sorts blocks alphabetically by Simulink path.
  - `By block type`: Sorts blocks alphabetically by block type.
  - `By block depth`: Sorts blocks by their depth in the model.
  - `By layout (left to right)`: Sorts blocks by their location in the model layout, by *rows*. The block appearing the furthest toward the left top corner of the

model is the anchor for the row. The row contains all other blocks that overlap the horizontal area defined by the top and bottom edges of the anchor block. The other rows use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.



  - `By layout (top to bottom)`: Sorts blocks by their location in the model layout, by *columns*. The block appearing the furthest toward the left top corner of the model is the anchor for the column. The column contains all other blocks that overlap the vertical area defined by the left and right edges of the anchor block. The other columns use the same algorithm, using as the anchor the next unreported block nearest the left top of the model.
  - `By traversal order`: Sorts blocks by traversal order.
  - `By simulation order`: Sorts blocks by execution order.
- **Search for Simulink property name/property value pairs**: Reports on blocks with specified property name/property value pairs.
- **Search Stateflow**: Reports on Stateflow charts with specified property name/ property value pairs.

Insert Anything into Report?

Yes. Table.

Class

```
rptgen_sf.csf_summ_table
```

See Also

`Block Loop`, `Chart Loop`, `Model Loop`, `Object Loop`, `Signal Loop`, `State Loop`, `Stateflow Hierarchy Loop`, `System Loop`

System Filter

Run child components if current system meets specified conditions

Description

This component runs its child components if the current system meets the conditions that you specify with this component.

Properties

- **Report only if system has at least N blocks**: Specifies the minimum number of blocks that the system must include for any of the child components to run. If you enter `0`, child components run regardless of the number of blocks in the system.
- **Report only if system has at least N subsystems**: Specifies the minimum number of subsystems that the system must include for the child components to run. If you enter `0`, child components run regardless of the number of subsystems in the system.
- **Report only if system mask type is**: Specifies which masks to include in the generated report.
  - `Either masked or unmasked`
  - `Masked`
  - `Unmasked`
- **Custom filtering MATLAB code**: Specifies custom MATLAB filtering code that the System Filter applies when determining which systems and subsystems to report on in a System Loop component. The edit box includes a sample function (commented out) that you can use as a starting point for your own filtering function. Use the `isFiltered` variable for the output of your function. For example, to filter out systems and subsystems whose names start with `engine`, enter:

  ```
  isFiltered = strncmpi( currentSystem, 'engine', 6);
  ```

Insert Anything into Report?

No.

Class

`rptgen_sf.csf_obj_filter`

See Also

`System Loop`

System Hierarchy

Create nested list that shows hierarchy of specified system

Description

This component creates a nested list that shows the hierarchy of a specified system. The list can display all systems in a model, or the parents and children of the current system.

Starting System

- **Build list from**: Specifies the system or model from which to build the list.
  - `Current system`
  - `Current model`
- **Emphasize current system**: Highlights the current system or model in the generated report.

Display Systems

- **Show number of parents**: Specifies the number of parents to list.
- **Display peers of current system**: Shows the peers of the current system in the generated report.
- **Show children to depth**: Specifies the depth of children to list.

List Formatting

- **List style**:
  - `Bulleted list`
  - `Numbered list`: Allows you to select numbering options in the **Numbering style** section.

- **Numbering style**: Allows you to select a numbering style in the selection list, by setting **List style** to `Numbered List`. Only the `RTF/DOC` report format supports this option.
  - `1,2,3,4,...`
  - `a,b,c,d,...`
  - `A,B,C,D,...`
  - `i,ii,iii,iv,...`
  - `I,II,III,IV,...`

Insert Anything into Report?

Yes. List.

Class

`rptgen_sl.csl_sys_list`

See Also

`Model Loop`, `System Loop`

System Loop

Specify systems and subsystems on which to loop, as defined by parent component

Description

This component runs its child components for each system defined by the parent component. For example, to include systems and subsystems within a given model in the report, you can include this component as the child of a `Model Loop` component.

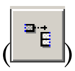
For conditional processing systems, you can use the `RptgenSL.getReportedSystem` function. For more information, see “Loop Context Functions”.

Report On

- **Loop on Systems**:
  - **Select systems automatically**: Reports on all systems in the current context as set by the parent component.
    - `Model Loop`: Reports on systems in the current model.
    - `System Loop`: Reports on the current system.
    - `Signal Loop`: Reports on the parent system of the current signal.
    - `Block Loop`: Reports on the parent system of the current block.

    If this component does not have any of these components as its parent, selecting this option reports on all systems in all models.
- **Custom - use system list**: Reports on a list of specified systems. Specify the full path of each system.
- `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace. The `%<>` notation can denote a string or cell array. For more information, see `%<VariableName> Notation` on the `Text` component reference page.
- **Include subsystems in Simulink functions**: Specifies whether to include subsystems in Simulink functions. By default, this option is enabled.

Loop Options

- **Sort Systems**: Specifies how to sort systems.
  - `Alphabetically by system name` (default): Sorts systems alphabetically by name.
  - `By number of blocks in system`: Sorts systems by number of blocks. The list shows systems by decreasing number of blocks; that is, the system with the largest number of blocks appears first in the list.
  - `By system depth`: Sorts systems by their depth in the model.
  - `By traversal order`: Sorts systems in traversal order.
- **Search for**: Reports only on blocks with the specified property name/property value pairs. To enable searching, click the check box. In the first row of the property name and property value table, click inside the edit box, delete the existing text, and type the property name and value. To add a row, use the **Add row** button ().

  For information about subsystem property names and values, in “Block-Specific Parameters”, see the “Ports & Subsystems Library Block Parameters” section.

Section Options

- **Create section for each object in loop**: Inserts a section in the generated report for each object found in the loop.
- **Display the object type in the section title**: Inserts the object type automatically into the section title in the generated report.
- **Number sections by system hierarchy**: Hierarchically numbers sections in the generated report. Requires that **Sort Systems** be set to `By traversal order`.
- **Create link anchor for each object in loop**: Creates a hyperlink to the object in the generated report.

Examples

For an example of how to use this component with a `Model Loop` as its parent, see `Model Loop`.

Insert Anything into Report?

Yes, inserts a section if you select the **Create section for each object in loop** option.

Class

`rptgen_sl.csl_sys_loop`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

System Snapshot

Insert snapshot of the current system into report

Description

This component inserts a snapshot of the current system into the report. The Snapshot options control how the image file is stored. The Properties options control whether the image display includes callouts and a print frame. The Display options control how the image is displayed in a browser.

Snapshot Options

- **Format**: Specifies the image file format (for example, `JPEG` or `TIFF`). Select `Automatic SL Format` (the default) to choose the format best suited for the specified report output format automatically. Otherwise, choose an image format that your output viewer can read.
  - `Automatic SL Format` (uses file format selected in the Preferences dialog box)
  - `Bitmap (16m-color)`
  - `Bitmap (256-color)`
  - `Black and white encapsulated PostScript`
  - `Black and white encapsulated PostScript (TIFF)`
  - `Black and white encapsulated PostScript2`
  - `Black and white encapsulated PostScript2 (TIFF)`
  - `Black and white PostScript`
  - `Black and white PostScript2`
  - `Color encapsulated PostScript`
  - `Color encapsulated PostScript (TIFF)`
  - `Color encapsulated PostScript2`
  - `Color encapsulated PostScript2 (TIFF)`
  - `Color PostScript`

  - `Color PostScript2`
  - `JPEG high quality image`
  - `JPEG low quality image`
  - `JPEG medium quality image`
  - `PNG (screenshot)`
  - `PNG 24-bit image`
  - `Scalable Vector Graphics`
  - `Windows metafile`
- **Orientation**:
  - `Largest dimension vertical`: Positions the image so that its largest dimension is vertical.
  - `Landscape`
  - `Portrait`
  - `Use system orientation`: Uses the paper orientation setting for the system. Use the Simulink `PaperOrientation` parameter to specify the orientation.
  - `Full page image (PDF only)`: In PDF reports, scales images to fit the full page, minimizes page margins, and maximizes the size of the image by using either a portrait or landscape orientation.
- **Scaling**: Controls the size of the image in the image file.
  - `Automatic` (default): Automatically scales the image to output dimensions.
  - `Custom`: Specifies image size.
  - `Zoom`: Enlarges or reduces the image size to the percent that you specify. Use **Max Size** to specify the maximum size other than the default for the image.

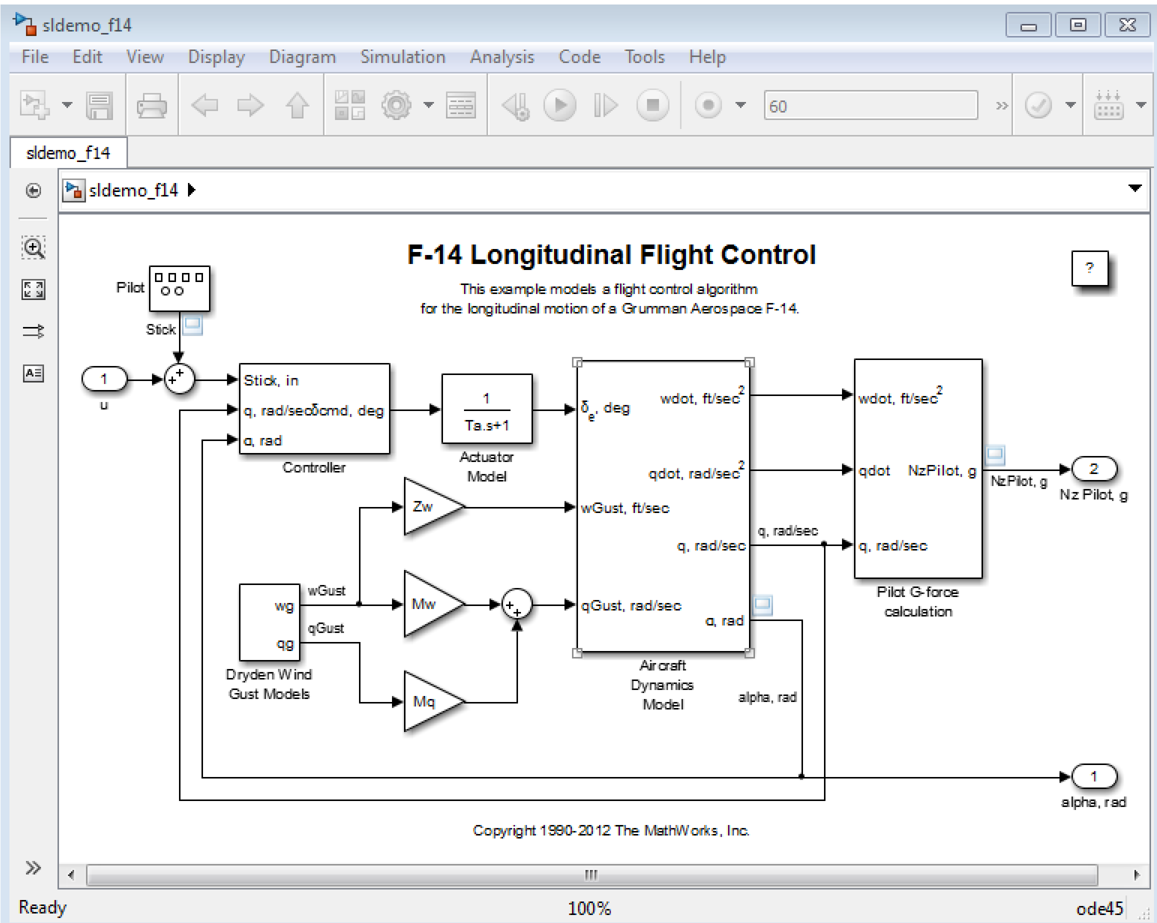
Note: Selecting the **Use printframe** deactivates the `Custom` and `Zoom` options and automatically scales the image to the print frame size.

Properties Options

- **Include callouts to describe visible objects**: Displays descriptive callouts for visible objects

- **Use printframe**: Prints a frame around the image. You can use the default frame, `rptdefaultframe.fig`, or use the Frame Editor to build a custom frame. For more information, see the `frameedit` function in Simulink documentation.

  The default frame is five inches wide and four inches high. It includes the name of the system and the model folder. This frame is optimized for use with a `portrait` paper orientation. The Flight Control Model in the `f14` Simulink model appears here with the default Simulink Report Generator frame option.



Display Options

To access the display options, click the **Advanced** button.

- **Scaling**: Controls size of the image, as displayed in a browser. Making an image larger using this option does not affect the storage size of the image, but the quality of the displayed image may decrease as you increase or decrease the size of the displayed image.

  Generally, to achieve the best and most predictable display results, use the default setting of `Use image size`.

  - `Use image size`: Causes the image to appear the same size in the report as on screen (default).
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
- **Size**: Specifies the size of the snapshot in a browser, using the format `w h` (width, height). This field is active only if you choose `Fixed size` in the **Scaling** selection list.
- **Max size**: Specifies the maximum size of the snapshot in a browser, using the format `w h` (width, height). This field is active only if you choose `Zoom` in the **Scaling** selection list.
- **Units**: Specifies the units for the size of the snapshot in a browser. This field is active only if you choose `Fixed size` in the **Image size** list box.
- **Alignment**: Only reports in `PDF` or `RTF` format support this property.
  - `Auto`
  - `Right`
  - `Left`
  - `Center`
- **Image title**:
  - `None` (Default)
  - `System name`: Uses the system name as the image name.
  - `Full system name`: Uses the system name, with path information, as the image name.
  - `Custom`: Specifies a custom title.

- **Caption**:
  - None (Default)
  - Description (use system description)
  - Custom: Specifies a custom caption.

Insert Anything into Report?

Yes. Image.

Class

rptgen_sl.csl_sys_snap

See Also

System Loop

To Workspace Plot

Capture plot figure created in the MATLAB workspace

Description

This component captures a plot figure created in the MATLAB workspace, and then inserts one or both of the following into the report:

- A table that includes input and output numeric values.
- A figure that plots the values included in the table.

Print Options

- **Image file format**: Specifies the image file format (for example, `JPEG` or `TIFF`) from this list. Select `Automatic HG Format` (the default) to choose the format best suited for the specified report output format automatically. Otherwise, choose an image format that your output viewer can read. Other options are:
  - `Automatic HG Format` (Uses the file format selected in the Preferences dialog box)
  - `Bitmap (16m-color)`
  - `Bitmap (256-color)`
  - `Black and white encapsulated PostScript`
  - `Black and white encapsulated PostScript (TIFF)`
  - `Black and white encapsulated PostScript2`
  - `Black and white encapsulated PostScript2 (TIFF)`
  - `Black and white PostScript`
  - `Black and white PostScript2`
  - `Color encapsulated PostScript`
  - `Color encapsulated PostScript (TIFF)`
  - `Color encapsulated PostScript2`
  - `Color encapsulated PostScript2 (TIFF)`

    - `Color PostScript`
    - `Color PostScript2`
    - `JPEG high quality image`
    - `JPEG medium quality image`
    - `JPEG low quality image`
    - `PNG 24-bit image`
    - `TIFF - compressed`
    - `TIFF - uncompressed`
    - `Windows metafile`
- **Paper orientation**:
    - `Landscape`
    - `Portrait`
    - `Rotated`
    - `Use figure orientation`: Uses the orientation for the figure, which you set with the `orient` command.
    - `Full page image (PDF only)`: In PDF reports, scales images to fit the full page, minimizes page margins, and maximizes the size of the image by using either a portrait or landscape orientation.

  For more information about paper orientation, see the `orient` command in the MATLAB documentation.
- **Image size**:
    - `Use figure PaperPositionMode setting`: Uses the `PaperPositionMode` property of the Handle Graphics figure to set the image size in the report. For more information about paper position mode, see the `orient` command in the MATLAB documentation.
    - `Automatic (same size as on screen)`: Sets the image in the report to the same size as it appears on the screen.
    - `Custom`: Specifies a custom image size. Specify the image size in the `size` and `units` fields.

- **Size**: Specifies the size of the Handle Graphics figure snapshot in the format `wxh` (width times height). This field is active only if you choose `Custom` in the **Image size** list box.
- **Units**: Specifies units for the size of the Handle Graphics figure snapshot. This field is active only if you choose `Set image size` in the **Custom** list box.
- **Invert hardcopy**: Uses the Handle Graphics figures `InvertHardcopy` property, which inverts colors for printing; it changes dark colors to light colors, and light colors to dark colors.
  - `Automatic`: Automatically changes dark axes colors to light axes colors. If the axes color is a light color, it is unchanged.
  - `Invert`: Changes dark axes colors to light axes colors, and light axes colors to dark axes colors.
  - `Don't invert`: Retains image colors displayed on screen in the printed report.
  - `Use figure's InvertHardcopy setting`: Uses the `InvertHardcopy` property set in the Handle Graphics image.
  - `Make figure background transparent`: Makes the image background transparent.

Display Options

- **Scaling**: Controls size of the image, as displayed in a browser. Making an image larger using this option does not affect the storage size of the image, but the quality of the displayed image may decrease as you increase or decrease the size of the displayed image.

  Generally, to achieve the best and most predictable display results, use the default setting of `Use image size`.
  - `Use image size`: Causes the image to appear the same size in the report as on screen (default).
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
- **Size**: Specifies the size of the snapshot in the format `w h` (width, height). This field is active only if you choose `Fixed size` in the **Scaling** list.

- **Max size**: Specifies the maximum size of the snapshot in the format `w h` (width, height). This field is active only if you choose `Zoom` from the **Scaling** list.
- **Units**: Specifies units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** list box.
- **Alignment**: Only reports in `PDF` or `RTF` format support this property.
  - `Auto`
  - `Right`
  - `Left`
  - `Center`
- **Title**: Specifies text to appear above the snapshot.
- **Caption**: Specifies text to appear under the snapshot.

Insert Anything into Report?

Yes. Figure.

Class

`rptgen_sl.csl_blk_toworkspace`

See Also

`Figure Loop`

Truth Table

Report on truth tables in Simulink and Stateflow models

Description

The Truth Table component reports on truth tables in Simulink and Stateflow models. It displays both the condition table and the action table. The parent component of the Truth Table determines its behavior.

- `Model Loop`: Reports on all truth tables in the current model.
- `System Loop`: Reports on all truth tables in the current system.
- `Block Loop` : Reports on all truth tables in the current block.
- `Signal Loop`: Reports on all truth tables in the current signal.

Title

Title: Specifies a title for the truth table.

- `No title`
- `Use Stateflow name`
- `Custom`

Condition Table

Specify display parameters for the condition table.

- **Show header**: Displays the column headers in the table.
- **Show number**: Displays the condition number column in the table.
- **Show condition**: Displays the condition column in the table.
- **Show description**: Displays the description column in the table.
- **Wrap if column count**: Specifies how many columns to display before creating a table continuation. If the specified number is greater than the number of columns that can appear on the page, some columns do not appear in the report.

Action Table

- **Show header**: Displays the column headers in the table.
- **Show number**: Displays the condition number column in the table.
- **Show condition**: Displays the condition column in the table.
- **Show description**: Displays the description column in the table. If you do not select this option, no action table appears in the report.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_sf.csf_truthtable`

See Also

`Block Loop`, `Model Loop`, `Signal Loop`, `System Loop`

Functions – Alphabetical List

report

Generate report from specified Simulink system

Syntax

```
report
report (filename,...)
report (___,-oOPATH)
report (___,-fFORMAT)
report (___,-genOption1,...)
[report1, report2, ...] = report (rptfile1, rptfile2, ...)
```

Description

- `report` with no arguments opens the Report Explorer. For more information on the Report Explorer, see "Report Explorer"
- `report (filename,...)` generates a report from the specified report setup files. You can specify one or more report setup files. When specifying the name of the report setup file, omit the `.rpt` file name extension.
- `report (___,-oOPATH)` sets the name of the generated report. You can specify a path or a single file name for the `OPATH` path argument.
- `report (___,-fFORMAT)` sets the output format and file name extension of the generated report. Supported formats include:

  - Adobe Acrobat PDF (`.pdf`)
  - HTML (`.html`)
  - Microsoft Word (`.doc`)
  - Rich Text format (`.rtf`)

  For example, `report('simple-report','-fPDF)` generates a PDF file.
- `report (___,-genOption1,...)` specifies one or more of the following report generation options:

  - `-noview` — Prevents launching the file viewer

- `-graphical` — Shows hierarchy in Report Explorer
- `-debug` — Enables debug mode
- `-quiet` — Sets error echo level to `0`
- `-sSTYLESHEETNAME` — Sets stylesheet name (not required when choosing format)

- `[report1, report2, ...] = report (rptfile1, rptfile2, ...)` returns the names of the generated reports. If the MATLAB Report Generator software cannot generate a given report, its returned name is empty.

Examples

Example 1: Setting the format of the generated report

- Generate the report `testrpt` in `PDF` format:

  ```
  report testrpt -fpdf
  ```
- Generate the report `testrpt` in `RTF` format:

  ```
  report testrpt -frtf
  ```
- Generate the report `testrpt` in Microsoft Word format:

  ```
  report testrpt -fdoc
  ```

  Note: Only Microsoft Windows platforms support this option.
- Generate a multipage `HTML` report from the `figloop-tutorial` report setup file:

  ```
  report figloop-tutorial -fhtml -shtml-!MultiPage
  ```

Example 2: Specifying the file and path of the generated report

Generate a report named `simple-report` in the folder `/tmp/index.html`:

```
report ('simple-report','-o/tmp/index.html')
```

More About

- "Generate Reports"

See Also

`setedit` | `rptconvert` | `rptlist` | `compwiz`

rptlist

Return list of all report s in MATLAB path

Syntax

```
rptlist
rptlist ('system_name')
list = rptlist
```

Description

`rptlist` with no arguments opens the Report Explorer, which lists available report setup files in the MATLAB path. You can open, run, or associate these files with the current Simulink system.

`rptlist ('system_name')` opens the Report Explorer with the Simulink system's `ReportName` property selected.

`list = rptlist` returns a list of report setup files in the MATLAB path.

See Also

`report` | `setedit` | `rptconvert` | `compwiz`

slwebview

Export Simulink models to Web views

Syntax

```
slwebview
filename = slwebview(system_name)
filename = slwebview(system_name,Name,Value)
```

Description

`slwebview` starts the Web View dialog box in the Report Explorer.

`filename = slwebview(system_name)` exports the subsystem `system_name` and its child systems to the file filename.

`filename = slwebview(system_name,Name,Value)` provides additional options specified by one or more Name,Value pairs.

Examples

Export Web View for a Subsystem and Systems that Contain that Subsystem

Open the fuel rate controller subsystem.

```
open_system('fuelsys')
```

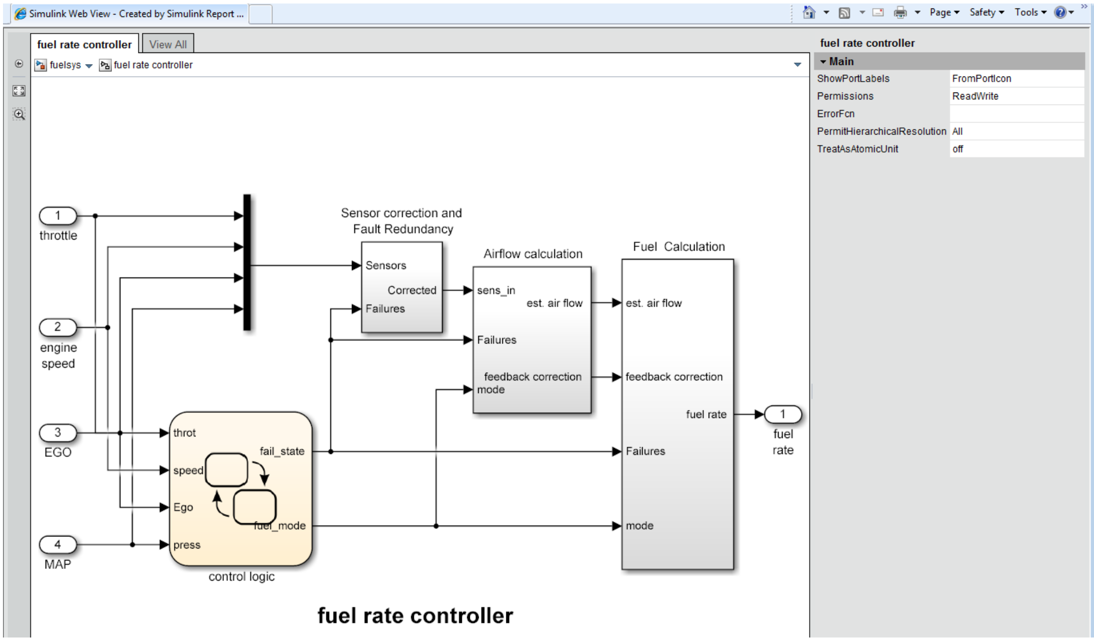
Export to a Web view the fuel rate controller subsystem and the system that contains it. Do not export the subsystems that it contains. This example assumes the current folder is the `H:` drive.

```
fuelsys_web_view = slwebview...
('fuelsys/fuel rate controller','SearchScope','CurrentAndAbove')
```

```
fuelsys_web_view =
```

```
H:\fuel_rate_controller\webview.html
```

The Web view displays in the system browser.



Export Web View with Access to Referenced Models

Open the `sldemo_mdlref_depgraph` model.

```
open_system('sldemo_mdlref_depgraph')
```

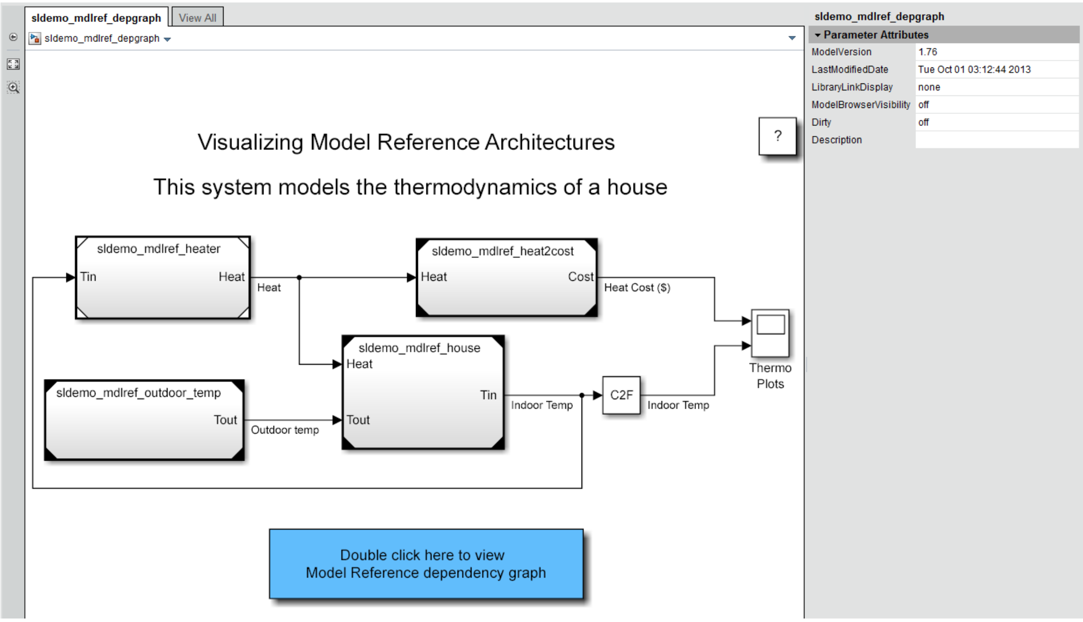
Export to a Web view the `sldemo_mdlref_depgraph` model and allow access to the models it references.

```
depgraph_web_view = slwebview...
('sldemo_mdlref_depgraph','FollowModelReference','true')
```

```
depgraph_web_view =

H:\sldemo_mdlref_depgraph\webview.html
```

The Web view displays in the system browser. In the Web view, you can open the models referenced by the Model blocks.



Click a Model block to see its properties. Double-click a Model block to display the referenced model.

- "Create and Use a Web View"

Input Arguments

`system_name` — The system to export to a Web view file

string 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. By default, child systems of the `system_name` system are also exported. Use the `SearchScope` name-value pair to export other systems, in relation to `system_name`.

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: `htmlFileName = slwebview(gcs,'LookUnderMasks','all',... 'FollowLinks','on')` Export to a Web view all layers of the model hierarchy to which the current system belongs, including the ability to interact with library links and masks.

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

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

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

`CurrentAndAbove` exports the Simulink system or the Stateflow chart specified by the `system_name` 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 `system_name`.

Data Types: `char`

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

`false` 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.

`true` exports all masked blocks.

`none` is supported for compatibility with earlier user code. Do not use this value in new code. This value is equivalent to `false`.

`all` is supported for compatibility with earlier user code. Do not use this value in new code. This value is equivalent to `true`.

`graphical` is supported for compatibility with earlier user code. Do not use this value in new code. This value is equivalent to `all`.

`functional` is supported for compatibility with earlier user code. Do not use this value in new code. This value is equivalent to `all`.

Data Types: `char`

`'FollowLinks'` — Specifies whether to follow links into library blocks
`false` (default) | `true`

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

`true` 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
`false` (default) | `true`

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

`true` 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
`true` (default) | `false`

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

`false` 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
`true` (default) | `false`

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

`false` 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
string

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

More About

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 SVG, either natively or with an Adobe SVG plugin. The MATLAB Web browser does not support Web views.

Template-Based Report Formatting

Report Generation Using Templates

Report Templates

A report template can contain:

- Style sheet with style definitions for report elements

  A style is a collection of formats for a report element.
- Fixed content

  The default templates do not include fixed content. You can add fixed content.
- Holes (blanks) that the DOM API fills with generated content

You can create custom Word and HTML templates to:

- Tailor report formatting to meet your specific formatting requirements.
- Add fixed content.
- Rearrange the order of the holes for generated content.

Benefits of Using Templates

Using a template when generating a report provides several benefits, compared to generating a report without using a template.

- Report generation is faster.
- Report generation does not use Java memory. Generating reports without using a template can cause Java to run out of memory.
- You can customize report formatting using standard techniques for specifying Word and HTML styles.

Custom Templates

The default templates produce reports that look similar to reports generated without using templates. You can create custom templates.

Component Formatting

For these Report Explorer components, you can specify an applicable style from a template style sheet. The style controls formatting for the component when you generate a report using the template that contains the style definition.

- Chapter/Section
- Text
- Paragraph
- List
- Title Page (the Abstract and the Legal Notice sections)
- Table
- Table Body
- Table Footer
- Table Header

For some components, you can specify a style for the title (for example, the title of a paragraph) and a style for the content.

To change the default formatting for instances of one of these components, in the template edit the style definition for the default style for that component (or component title). You can change the default format for components that have associated style in the template style sheet.

To change the style or format for an instance of a component, open the component property dialog box and use one of these approaches.

- Use the **Style Name** field to associate a different style with that instance.
- Use the component property dialog box to set a specific format. For example, the Paragraph component provides format options such as bold and underline.

Formats that you specify in a component property dialog box have priority over formatting specified using the **Style Name** field.

Related Examples

- “Generate a Report Using a Template” on page 10-5
- “Create Custom Microsoft Word Report Templates” on page 10-7
- “ Create Custom HTML Report Templates” on page 10-11

More About

- “Report Generation Using Templates” on page 10-2

Generate a Report Using a Template

In this section...

Generate a Report Using a Template for the File Format

1 In Report Explorer, in the **Outline** pane, select the report.

2 In the Report Options dialog box that appears in the **Properties** pane, set the **File format** field to one of these options:

   - `HTML (from template)`
   - `PDF (from template)`
   - `Word (from template)`

3 Optionally, from the list of style sheets available for the current file format, select a selecting a different style sheet than the default style sheet.

4 If you select `HTML (from template)`, choose a packaging options for the output files.

   - **Unzipped** — Generate the report files in a subfolder of the current folder. The subfolder has the report name.
   - **Zipped** — Package report files in a single compressed file that has the report name, with a `.zip` extension.
   - **Both Zipped and Unzipped**

5 In the toolbar, click the **Report** button ().

Generate the System Design Report Using an HTML Template

You can use a template when you generate the System Design Description report. This example shows how to generate a report using the PDF template.

1 Open a model, such as `vdp`.

2 In the Simulink Editor, select **File** > **Reports** > **System Design Description**.

3 In the Design Description dialog box, set **File format** to `PDF (from template)`.

4 Click **Generate**.

Related Examples

- “Create Custom Microsoft Word Report Templates” on page 10-7
- “ Create Custom HTML Report Templates” on page 10-11

More About

- “Report Generation Using Templates” on page 10-2

Create Custom Microsoft Word Report Templates

In this section...

Copy a Word Template

To customize the styles used in the default Word template, you need to copy that template (or a template that was copied from the default template) and modify or add style definitions in the copy. You cannot edit the default template.

1. In Report Explorer, select **Tools** > **Edit Document Conversion Template**.
2. In the Library pane (middle pane), select a template from the list of Word templates. For example, select the `DefaultWordTemplate`.
3. In the Properties pane, click **Copy template**.
4. In the file browser, navigate to the location where you want to save the template file.

   Select a path that is on the MATLAB path (for example, in the `MATLAB` folder in your home directory).

   Specify the file name, using the default file extension for a Word template (`.dotx`). Click **Save**.
5. In the list of templates in the middle pane, select the template copy you just created.
6. In the Properties pane, in the **Template id** and **Display name** fields, specify a unique ID and display name for the template.

   The display name is the name that appears in the Report Explorer list of templates. Commands that act on templates use the template ID to specify the template.
7. Optionally, in the Properties pane, fill in the **Description** and **Creator** fields.
8. To save the template properties you entered, move the cursor outside of the Properties pane and click.

Edit Existing Word Styles in a Template

You can customize or add format styles in a custom Word template.

1 In the list of templates in the middle pane, select the custom template that you want to edit.

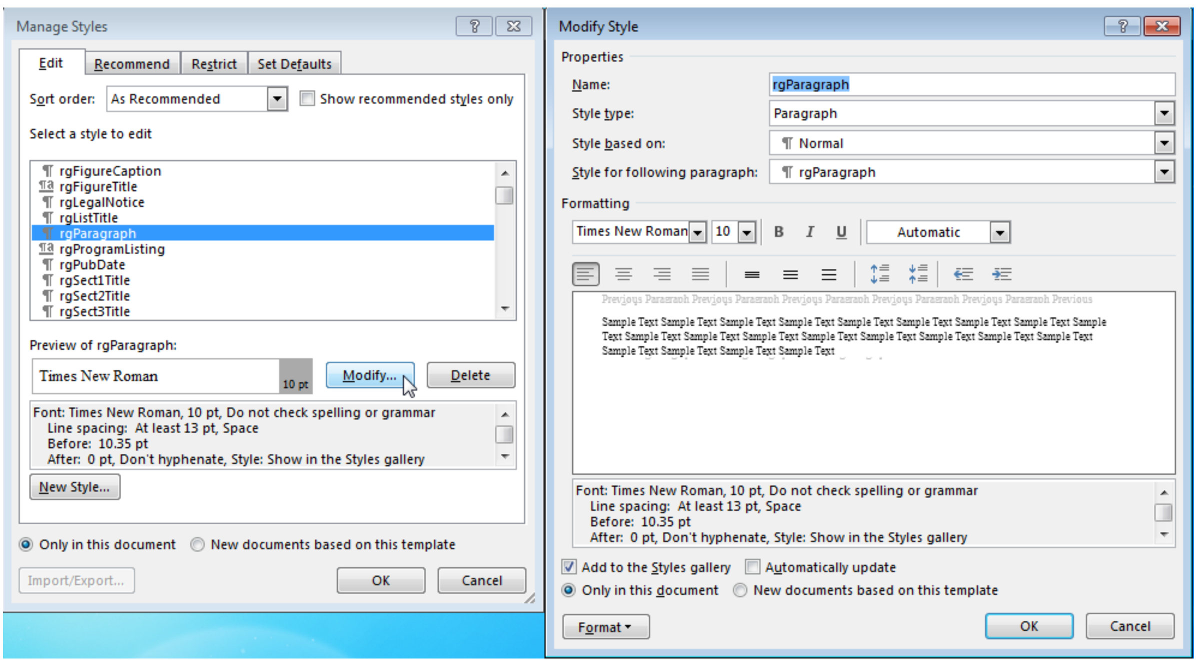
Tip If the Report Explorer middle pane does not show a list of templates, then select **Tools > Edit Document Conversion Template**.

2 In the Properties pane, click **Open stylesheet**.

3 In the Manage Styles dialog box, select a style to edit.

Styles that begin with `rg` (for example, `rgParagraph`) are the default styles used for reports. A default style applies to all instances of a component with which it is associated, unless you override the style.

4 Click **Modify**. The Modify Style dialog box appears.



5 In the Modify Style dialog box, change style definitions to meet your formatting requirements. For example, you can change the font family, font size, indentation, etc. To save your changes, click **OK** and close the dialog box.

6 In Word, save and close the template.

For more information about using Word styles, see the Microsoft Word documentation.

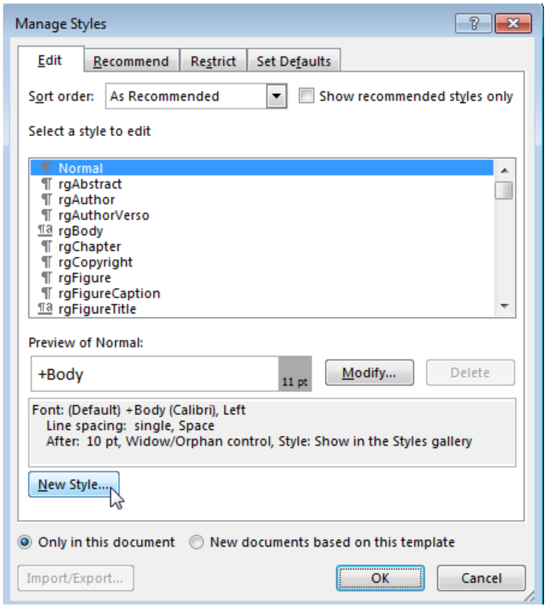
Add a Style to a Word Template

To add a new style to a Word template:

1 Open the Manage Styles dialog box, as described in "Edit Existing Word Styles in a Template" on page 10-8.

2 If applicable, select an existing style to use as a starting point for the new style.

3 Click the **New Style** button.



4 Specify a name for the new style and define the style characteristics. To save the new style definition, click **OK** and close the dialog box.

5 In Word, save and close the template.

Modify or Add Fixed Content

In the Report Content hole for generated content, you can add fixed text. When you generate the report, the fixed text appears before the content generated for the Report Content hole.

Change the Order of Holes

You can change the order of the holes in a template. However, do not:

- Remove any holes from the template.

  The Report Explorer requires that a template includes the holes to successfully generate a report.
- Add any holes.

  The Report Explorer ignores any additional holes.

Related Examples

- “ Create Custom HTML Report Templates” on page 10-11
- “Generate a Report Using a Template” on page 10-5

More About

- “Report Generation Using Templates” on page 10-2

Create Custom HTML Report Templates

In this section...

Copy an HTML Template

To customize the format styles used in the default HTML template, copy that template (or a template that was copied from the default template) and modify or add style definitions in the copy. You cannot edit the default HTML template.

1. In Report Explorer, select **Tools** > **Edit Document Conversion Template**.
2. In the middle pane, select a template from the list of HTML templates.
3. In the **Properties** pane, click **Copy template**.
4. In the file browser, navigate to the location where you want to save the template file.

   Select a path that is on the MATLAB path (for example, in the `MATLAB` folder in your home directory).

   Specify the file name, using the default file extension for a HTML template (`.htmtx`) and click **Save**.
5. In the list of templates in the middle pane, select the template copy.
6. In the **Properties** pane, in the **Template id** and **Display name** fields, specify a unique ID and display name for the template.

   The display name is the name that appears in the Report Explorer list of templates. Commands that act on templates use the template ID to specify the template.
7. Optionally, in the **Properties** pane, fill in the **Description** and **Creator** fields.
8. To save the template properties you entered, move the cursor outside of the Properties pane and click.

Select an HTML Editor

By default, when you edit a Report Explorer HTML style sheet, the style sheet appears in the MATLAB Editor.

To use a different editor for editing an HTML template:

1 In Report Explorer, select **File** > **Preferences**.
2 In the **Edit HTML Command** text box, enter a MATLAB expression that opens the HTML editor you want to use. For example:

```
system('Dreamweaver %<FileName> &')
```

When you open an HTML stylesheet, the Report Explorer automatically replaces `FileName` with the template that you selected.

Edit HTML Styles in a Template

You can customize or add format styles in a custom HTML template.

1 In the list of templates in the middle pane, select the custom template that you want to edit.

Tip If the Report Explorer middle pane does not show a list of templates, then select **Tools** > **Edit Document Conversion Template**.

---

2 In the **Properties** pane, click **Open stylesheet**.
3 In the HTML editor, edit the cascading style sheet (CSS).

For information about editing a cascading style sheet, see documentation such as the W3Schools.com CSS tutorial.

4 Save the style sheet.

Related Examples

- “Create Custom Microsoft Word Report Templates” on page 10-7
- “Generate a Report Using a Template” on page 10-5

More About

- “Report Generation Using Templates” on page 10-2

Create a Report Program

Create a Report Program

The MATLAB Report Generator includes a set of functions, called the DOM (Document Object Model) API, that allows you to generate Word, HTML, and PDF reports programmatically. For example, the following MATLAB script uses the API to generate and display an HTML report displaying today's date.

```
import mlreportgen.dom.*;
report = Document('today');
append(report, ['Today is ', date, '.']);
close(report);
rptview(report.OutputPath);
```

To get started learning about creating reports with the DOM API, see "Document Object Model" on page 11-4.

More About

- "Document Object Model" on page 11-4

Document Object Model

The DOM API creates a representation of a report document in your system's memory. Such a representation is often referred to as a Document Object Model (DOM). Hence, the DOM API's name.

The DOM API's document object model consists of a hierarchical set of data structures, known as objects, that represent the document and its contents. At the top of the hierarchy is an object representing the document itself. The document object maintains a list of objects, called its children, that represent its contents (such as paragraphs, images, tables, lists, and so on). Each child object, in turn, maintains a list of its contents. For example, a table lists its rows, a row lists its table entries, a table entry lists its contents, and so on.

The DOM API contains functions that allow you to create and assemble DOM objects, such as paragraphs, images, and tables, into a model of a specific document. You can then use the API to write the model out to disk as an HTML or Microsoft Word document file.

DOM Object Help and Documentation

For a list of the DOM objects, type the following at the MATLAB prompt.

```
help mlreportgen.dom
```

To get help for a specific object, such as a `Paragraph`, use a `help` command such as this.

```
help mlreportgen.dom.Paragraph
```

To get a complete list of DOM API classes and functions in the MATLAB Report Generator documentation, open the **Functions** pane.

To see the documentation reference page for an object, search in documentation or in MATLAB use a `doc` command such as this.

```
doc mlreportgen.dom.Paragraph
```

Related Examples

- “Construct a DOM Object” on page 11-6
- “Get and Set DOM Object Properties” on page 11-8

- "Import the DOM API Package" on page 11-7

Construct a DOM Object

The DOM API includes a special set of MATLAB functions, called constructors, for creating DOM objects of various types, or classes.

The name of an object constructor is the name of the MATLAB class from which the DOM creates an object. For example, the name of the constructor for a DOM paragraph object is `mlreportgen.dom.Paragraph`. Some constructors do not require any arguments. Other constructors can take one or more arguments that typically specify its initial content and properties. For example, the following line creates a paragraph whose initial content is `Chapter 1.`

```
p = mlreportgen.dom.Paragraph('Chapter 1.');
```

A constructor returns a handle to the object it creates. Assigning the handle to a variable allows you to subsequently append content to the object or set its properties. For example, the following line appends content to the paragraph object `p` created in the previous example.

```
append(p,'In the Beginning');
```

Note that you can assign an object handle to multiple variables and hence access the same object via multiple variables.

Related Examples

- "Import the DOM API Package" on page 11-7
- "Get and Set DOM Object Properties" on page 11-8

More About

- "Document Object Model" on page 11-4

Import the DOM API Package

All DOM class names, and hence constructor names, include the prefix `mlreportgen.dom`. To avoid the need to include the prefix in your code, insert the following statement at the beginning of any script or function that uses the DOM API.

```
import mlreportgen.dom.*;
```

The documentation frequently refers to DOM API objects and functions without the `mlreportgen.dom` prefix, assuming that you have already imported the DOM API package.

Related Examples

- "Create a Report Program" on page 11-3

More About

- "Document Object Model" on page 11-4

Get and Set DOM Object Properties

To get or set the property of a document object, use dot notation, which involves appending a period to the name of a variable that references the object, followed by the property name. For example, the following line saves the current font family of a paragraph referenced by p and sets it to a new font family.

```
saveFont = p.FontFamily;
p.FontFamily = 'Arial';
```

Related Examples

- "Construct a DOM Object" on page 11-6
- "Use Format Properties" on page 11-24

More About

- "Document Object Model" on page 11-4

Create a Document Object to Hold Content

Every report program must create an `mlreportgen.dom.Document` object to hold report content. Use the `mlreportgen.dom.Document` constructor to create a `Document` object.

If you use the constructor with no arguments, the DOM API creates an HTML document named `Untitled.htmx` in the current folder.

You can specify the file system path of the report as the first argument of the constructor.

You can specify the type of report to be generated by using a second argument. You can specify the type to be `'html'`, `'docx'` (for Microsoft Word), or `'pdf'`. If you specify `'pdf'`, the DOM API generates the report as a Word document. You can then use the `rptview` function to convert it to PDF, or you can open the report in Word and save it as PDF.

This `Document` constructor creates an HTML report called `myReport`.

```
d = Document('myreport','html');
```

Using a third argument, you can specify the file system path of a Word or HTML template to be used as a basis for creating the report. You need to specify a template only if you are using template-based formatting (using style sheets) or form-based report generation. If you specify a template, it must be a Word template (`.dotx`) for Word or PDF reports or an HTML template (`.htmtx`) for HTML reports. For example, this `Document` constructor creates a Word report using the Word template `myWordTemplate.dotx`.

```
d = Document('myreport','dotx','myWordTemplate');
```

See Also

Functions
`rptview`

Classes
`mlreportgen.dom.Document`

Related Examples

- "Create a Report Program" on page 11-3

- “Use Style Sheets” on page 11-21
- “Construct a DOM Object” on page 11-6

More About

- “Form-Based Reporting” on page 11-26
- “Document Object Model” on page 11-4

Add Content to a Report

The DOM append function allows you to add content to documents, paragraphs, tables, and other DOM objects that serve as containers for report content. The append function takes two arguments. The first argument is the object to which the content is to be appended. The second is the content to be appended. In this example, the text Hello World is appended to the document.

```
d = Document('MyReport');
append(d,'Hello World');
```

The append function throws an error if the second argument (the content to be appended), is incompatible with the first argument (the object to which the content is to be appended). For example, the append method in the following script throws an error.

```
% This code throws an error
image = Image('membrane.png');
append(image,Paragraph('Hello World'));
```

This is because you cannot add a paragraph to an image. The reference documentation for classes lists the types of objects that you can append to instances of the classes. To get a complete list of DOM API classes and functions in the MATLAB Report Generator documentation, open the **Functions** pane. To see the documentation reference page for an object, search in documentation or in MATLAB use a doc command such as this.

```
doc mlreportgen.dom.Paragraph
```

As shown in the preceding examples, the append method, depending on the target object type, allows you to append strings, doubles, arrays, and other basic MATLAB data types, without first converting the data to DOM objects. The function converts the appended data to a DOM object before appending it to the target object. For example, the following script appends a two-dimensional array of strings to a document as a table.

```
d = Document('MyDoc');
tableArray = {'a','b';'c','d'};
append(d,tableArray);
```

Many constructors also allow you to specify basic MATLAB data types as the initial content of the object when you construct the object. This example is equivalent to the preceding example.

```
d = Document('MyDoc');
tableArray = {'a','b';'c','d'};
```

```
append(d,Table(tableArray));
```

See Also

Functions
`mlreportgen.dom.Paragraph.append`

Related Examples

- “Construct a DOM Object” on page 11-6
- “Clone a DOM Object” on page 11-13
- “Add Content as a Group” on page 11-14
- “Stream a Report” on page 11-16

More About

- “Document Object Model” on page 11-4

Clone a DOM Object

If you attempt to append an object more than once to the same object or to append an object to multiple objects, the `append` function throws an error. If you need to append an object multiple times, use the `clone` function to create copies of the object.

```
d = Document('MyDoc');
text = append(d,'Hello World');
text.Color = 'magenta';
text = clone(text);
text.Color = 'cyan';
append(d,text);
```

See Also

Functions
`mlreportgen.dom.Paragraph.clone`

Related Examples

- “Add Content to a Report” on page 11-11
- “Construct a DOM Object” on page 11-6

More About

- “Document Object Model” on page 11-4

Add Content as a Group

You can use a group to include the same content in different parts of a report. The DOM API clones the members of a group before appending them to another object.

This example shows the key code to include. After describing the steps involved in using a group, this example includes code for a complete report that uses a group.

1 Define the DOM objects that you want to include repeatedly in a report.

```
disclaimerHead = Heading(2,'Results May Vary');
disclaimerIntro = Paragraph('The following results assume:');
disclaimerList = UnorderedList(...
    {'Temperature between 30 and 70 degrees F',...
    'Wind less than 20 MPH','Dry road conditions'});
```

2 Define a `Group` object that includes the DOM objects for the group. For example:

```
disclaimer = Group();
append(disclaimer,disclaimerHead);
append(disclaimer,disclaimerIntro);
append(disclaimer,disclaimerList);
```

3 Append the `Group` object in the place in the report where you want to repeat the content. For example, if the document object is `doc`:

```
append(doc,disclaimer);
```

This code builds on the code shown above.

```
import mlreportgen.dom.*;
doc = Document('groupReport','html');
disclaimerHead = Heading(2,'Results May Vary');
disclaimerIntro = Paragraph('The following results assume:');
disclaimerList = UnorderedList(...
    {'Temperature between 30 and 70 degrees F',...
    'Wind less than 20 MPH','Dry road conditions'});
disclaimer = Group();
append(disclaimer,disclaimerHead);
append(disclaimer,disclaimerIntro);
append(disclaimer,disclaimerList);
append(doc,disclaimer);
p1 = Paragraph('First set of results...');
p1.Bold = true;
p2 = Paragraph('more report content...');
```

```
p2.Bold = true;
append(doc,p1);
append(doc,p2);
append(doc,disclaimer);
close(doc);
rptview('groupReport','html');
```

See Also

Functions
mlreportgen.dom.Paragraph.append

Classes
mlreportgen.dom.Group

Related Examples

- "Add Content to a Report" on page 11-11

Stream a Report

The DOM API supports two modes of appending content to a document:

- In-memory — Creates the document entirely in memory. In-memory is the default mode.
- Streaming — Streaming mode writes objects to disk as they are appended to a document. Streaming mode allows you to create large reports on systems with modest memory.

To enable streaming mode, set the `StreamOutput` property of the `Document` object for the report to `true`.

```
d = Document('MyDoc');
d.StreamOutput = true;
```

See Also

Classes
`mlreportgen.dom.Document`

Related Examples

- “Add Content to a Report” on page 11-11

Report Packages

A Microsoft Word document packages all of its contents, text, images, style sheets, and so on, in a single compressed file having a `.docx` extension.

For HTML documents, the DOM API defines an analogous packaging scheme, with an `.htmtx` compressed file extension. By default, the DOM API generates HTML reports as `.htmx` files. To generate an HTML report in unzipped format or both zipped and unzipped format, set the `PackageType` property of the `Document` object for a report to `'unzipped'` or `'both'`, respectively.

See Also

Functions
`unzipTemplate` | `zipTemplate`

Classes
`mlreportgen.dom.Document`

More About

- “Document Object Model” on page 11-4

Close a Report

The last step in creating a report with the DOM API is to close the report. Closing a report writes out any content that remains in memory and closes the report file. Use the `close` function.

```
d = Document('MyDoc');
append(d,'Hello World');
close(d);
```

See Also

Functions
`mlreportgen.dom.Document.close`

Related Examples

- "Create a Report Program" on page 11-3

More About

- "Document Object Model" on page 11-4

Display a Report

The DOM API `rptview` function allows you to display a generated report in a viewer appropriate to its document type: the Microsoft Word editor for Word documents, an HTML browser for HTML reports, and Adobe Acrobat for PDF reports.

The `rptview` function takes two arguments:

- The path of the report
- The output type: `'html'`, `'docx'`, or `'pdf'`

If you omit the second argument (the output type), `rptview` uses the output type from the report's extension.

If an HTML report is in zipped format, `rptview` creates a copy of the report in your temporary directory and displays the temporary copy. If you specify `'pdf'`, the function uses Word to convert the report to PDF format. It then displays the report in Adobe Acrobat.

See Also

Functions
`rptview`

Related Examples

- "Create a Report Program" on page 11-3

Report Formatting Approaches

The DOM API supports three approaches to formatting document objects in reports:

- Style sheets — Assign to a document object (such as a paragraph, table, or list) a style from a Microsoft Word or HTML template, using the `StyleName` property of a document object. A style is a collection of formats.
- Format objects — Use format objects, such as a `FontFamily` object, with the `Style` property of a document object.
- Format properties — Use format properties of a document object. For example, for a `Paragraph` object `p`, you can specify `p.Color = 'red'`.

Related Examples

- "Use Style Sheets" on page 11-21
- "Use Format Objects" on page 11-23
- "Use Format Properties" on page 11-24

More About

- "Format Inheritance" on page 11-25

Use Style Sheets

A style is a collection of formats that together define the appearance of a document object, such as a paragraph, table, or list. Microsoft Word allows you to define styles and then assign them to paragraphs, tables, and other documents by name. The assigned style then determines how Word renders the document object on the screen or printed page. Word stores the styles in a document as an object called a style sheet. HTML browsers support a similar capability.

DOM API objects that have a `StyleName` property allow you to leverage Word and HTML style sheets to format reports as follows.

1. Create a Word or HTML template, using the DOM API or a Word or HTML editor, depending on the report type.
2. Optionally, you can change a template style definition or add a new style. For details, see “Modify Styles in a Microsoft Word Template” on page 11-96 or “Modify Styles in an HTML Template” on page 11-104.
3. In a DOM report, create a `Document` object that uses the template.
4. Assign the names of styles defined in the style sheet to the `StyleName` property of objects that you want to have the specified style.

For example, the following script assigns a style named `Warning` to a paragraph object. It assumes that you have defined the `Warning` style previously in a Word template named `MyTemplate.dotx`.

```
d = Document('MyDoc','docx','MyTemplate');
p = Paragraph('Danger');
p.StyleName = 'Warning';
append(d,p);
close(d);
```

Assigning the `Warning` style to the DOM paragraph object causes Word to use the `Warning` style to render the generated paragraph when you open the generated report.

Tip Some document object constructors allow you to specify the value of the `StyleName` property as an argument. For example, this paragraph has the text `Danger` and uses a style defined for the template style named `Warning`.

```
p = Paragraph('Danger','Warning');
```

Related Examples

- "Create a Microsoft Word Template" on page 11-92
- "Modify Styles in a Microsoft Word Template" on page 11-96
- "Create an HTML Template" on page 11-101
- "Modify Styles in an HTML Template" on page 11-104

More About

- "Report Formatting Approaches" on page 11-20
- "Format Inheritance" on page 11-25

Use Format Objects

A format object is a MATLAB program entity that defines the properties and functions of a specific type of document format, such as a font family or font size. The DOM API provides a set of constructors for creating format objects corresponding to most of the formatting options available in HTML and Word documents. Most DOM document objects include a `Style` property that you can set to a cell array of format objects. Together, format objects and the document object `Style` property allow you to format a document object by creating an array of format objects that define the appearance (style), of the object and assigning this array to the `Style` property of the document object. For example, the following script uses format objects to specify the style of a warning paragraph.

```
p = Paragraph('Danger!');
p.Style = {Color('red'),FontFamily('Arial'),FontSize('18pt')};
```

You can assign the same array of format objects to more than one DOM document object. This allows you to create a programmatic equivalent of a template style sheet. For example:

```
warning = {Color('red'), FontFamily('Arial'), FontSize('18pt')};
p = Paragraph('Danger!');
p.Style = warning;
p = Paragraph('Caution!');
p.Style = warning;
```

The DOM API allows you to assign any format object to any document object, regardless of whether the format applies. If the format does not apply, it is ignored.

More About

- “Report Formatting Approaches” on page 11-20
- “Format Inheritance” on page 11-25

Use Format Properties

Most DOM objects have a set of properties corresponding to the format options most commonly used for an object of that class. You can use dot notation with format properties to specify formats for an object. For example, the following code sets the font and color of text in a paragraph, using the `Color`, `FontFamily`, and `FontSize` format properties of a `Paragraph` object.

```
p = Paragraph('Danger!');
p.Color = 'red';
p.FontFamily = 'Arial';
p.FontSize = '18pt';
```

Assigning a value to a format property causes the API to create an equivalent format object and assign it to the `Style` property of the document object. Similarly, assigning a format object to an object's `Style` property causes the API to assign an equivalent value to the corresponding format property if it exists. In this way, the API keeps format properties for an object in sync with the `Style` property of the object.

Note: When you change the value of a format property, the DOM API:

- Creates a clone of the corresponding format object
- Changes the value of the clone's corresponding format object property
- Replaces the original format object with the clone in the array of format objects assigned to the document object

In this way, the DOM prevents changing a format property in one object from changing a style originally assigned to other objects as well.

More About

- "Report Formatting Approaches" on page 11-20
- "Format Inheritance" on page 11-25

Format Inheritance

The DOM API allows you to use both template-based styles and format object-based styles (or equivalent format properties) to specify the appearance of an object. If you set both the `StyleName` and the `Style` property of an object, the formats in the `Style` property override corresponding formats specified by the template-based style of the `StyleName` property. Consider, for example, the following script.

```
d = Document('MyDoc','docx','MyTemplate');
p = Paragraph('Danger!');
p.StyleName = 'Warning';
p.Style = {Color('red')};
append(d,p);
close(d);
```

Suppose that the `Warning` style defines the color of a warning as yellow. In that case, this example overrides the color specified by the `Warning` style.

If a document object does not specify a `StyleName` (a template-based style), it inherits from its container any formats that it does not itself specify. The container itself inherits any formats that it does not specify from its container, and so on, all the way to the top of a container hierarchy. Format inheritance allows you to use a single statement to assign a format for all the objects contained by a container. For example, the following script uses a single `Style` property to assign a color to all the entries in a table.

```
d = Document('MyDoc');
tableArray = {'a', 'b'; 'c', 'd'};
table = append(d, tableArray);
table.Style = {Color('blue')};
append(d, table);
close(d);
```

Related Examples

- “Use Style Sheets” on page 11-21
- “Use Format Objects” on page 11-23

More About

- “Report Formatting Approaches” on page 11-20

Form-Based Reporting

The DOM API supports a form-based approach to report generation. You can use Microsoft Word or an HTML editor to create a template that defines the fixed content of the form, interspersed with holes (blanks), that your DOM report program fills with generated content.

Related Examples

- “Fill in the Blanks in a Report Form” on page 11-27
- “Use Subforms in a Report” on page 11-29
- “Create a Microsoft Word Template” on page 11-92
- “Add Holes in a Microsoft Word Template” on page 11-93
- “Create an HTML Template” on page 11-101
- “Add Holes in an HTML Template” on page 11-102

Fill in the Blanks in a Report Form

Navigate Holes in the Form

When you create a form template, you associate an ID with each hole in the template. This allows you to navigate the holes in a form, using the DOM `moveToNextHole` function. The first time you execute this function, the DOM API copies to the output document all of the text up to the first hole in the template. At this point, you can start adding content to the output document, using this DOM `append` function, thereby filling in the first hole. The next time you execute this function, the DOM API copies all the text between the first and second hole in the template to the output document. You can then fill in the second hole by appending content to the output document. In this way, you generate the output document by copying the content from the template and filling in all its holes.

For example, this function generates a report from a Word template that has three holes named `Title`, `Author`, and `Content`. The arguments `title`, `author`, and `content`, are assumed to be strings.

```
function makerpt(title,author,content,rptname,rpttemplate)
      import mlreportgen.dom.*
      rpt = Document(rptname,'docx',rpttemplate);

      while ~strcmp(rpt.CurrentHoleId,'#end#')
          switch rpt.CurrentHoleId
               case 'Title'
                   append(rpt,title);
               case 'Author'
                   append(rpt,author);
               case 'Content'
                   append(rpt,content);
               end
           end
           moveToNextHole(rpt);
       end

      close(rpt);
```

See Also

Functions

`mlreportgen.dom.Document.moveToNextHole`

Related Examples

- "Use Subforms in a Report" on page 11-29
- "Create a Microsoft Word Template" on page 11-92
- "Add Holes in a Microsoft Word Template" on page 11-93
- "Create an HTML Template" on page 11-101
- "Add Holes in an HTML Template" on page 11-102

More About

- "Form-Based Reporting" on page 11-26

Use Subforms in a Report

A document part is a form that you can add to a document or to another document part. Document parts simplify generation of sections of a report that have the same form, such as sections that report on the results of a series of tests or the performance of a series of financial portfolios. Use a similar approach as you do for main document forms.

1 Create a template that defines the form of the document part.
2 For each section:
    a Create an `mlreportgen.dom.DocumentPart` object based on the template.
    b Fill in the holes.
    c Append the part to the main document.

For an example of a report that uses subforms, open the Functional Report example.

Tip The DOM API allows you to store the templates for document parts in the main template for a report. This allows you to use a single template file to supply all the templates required for a report. For details, see “Create Document Part Template Libraries” on page 11-31.

See Also

Functions
`mlreportgen.dom.Document.moveToNextHole`

Classes
`mlreportgen.dom.DocumentPart`

Related Examples

- “Fill in the Blanks in a Report Form” on page 11-27
- “Create a Microsoft Word Template” on page 11-92
- “Add Holes in a Microsoft Word Template” on page 11-93
- “Create an HTML Template” on page 11-101
- “Add Holes in an HTML Template” on page 11-102

More About

- "Form-Based Reporting" on page 11-26

Create Document Part Template Libraries

In this section...

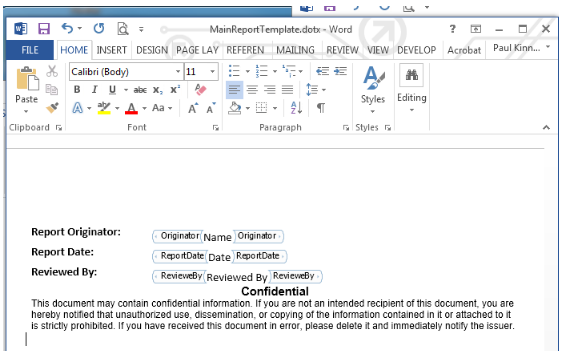
A document part template library is a set of document part templates stored by name in another template. You can create a document part based on a template stored in a library by specifying the name of the template in the document part constructor. Document part template libraries allow you to store all the templates for a report in a single template file, for example, the main template file of a report.

Create a Document Part Template Library in a Microsoft Word Template File

You can use the Quick Part Gallery in Word to create a document part template library in the main template of a report. A Quick Part Gallery is a collection of reusable pieces of preformatted content, called quick parts, that is stored in the document. You can use quick parts as templates for DOM `DocumentPart` objects.

1 Open the Word template in which you want to create the document part template.

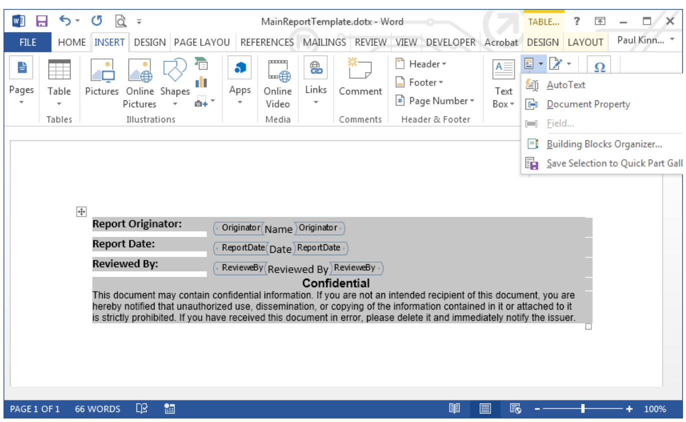
2 In the template, create the Word content to serve as a prototype for the document part template. (You will delete the prototype after copying it to the parent template Quick Part Gallery.) The document part template content that you create can contain holes and page layout sections, as well as other types of Word content. For example:



3 On the Word ribbon, select the **Insert** tab.

4 Select the content that you have created for the document part template.

5 On the **Insert** ribbon, click the **Explore Quick Parts** button. Select **Save Selection to the Quick Part Gallery** to save a copy of the selected prototype in the Quick Part Gallery of the template file. The Create New Building Block dialog box appears.



6 In the Create New Building Block dialog box, in the **Name** field, enter a unique name for the template. Use this name in the constructor of a `DocumentPart` object to be based on this quick part.

7 For the first document part template you create in the template file, in the **Category** list, click `Create New Category`. Create a category named `mlreportgen`. Then select `mlreportgen` from the **Category** list.

Otherwise, select `mlreportgen` from the **Category** list.

8. In the **Description** field, enter a template description and click **OK**.
9. Delete the content that served as the prototype for the document part template.
10. Save the template file.

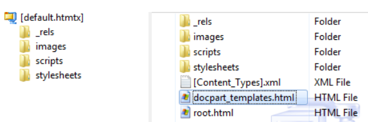
Modify a Document Part Template in a Quick Part Gallery

You can modify a document part template located in a Quick Part Gallery.

1. Open the Word template that contains the document part template.
2. Click in the template where you want to create an instance of the document part template.
3. In the Word ribbon, select the **Insert** tab.
4. On the **Insert** ribbon, click the **Explore Quick Parts** button to display the Quick Part Gallery.
5. To create an instance of the template in the parent template file, in the Quick Part Gallery, select the document part template to modify.
6. Edit the instance.
7. Select and save the modified instance to the Quick Part Gallery using the same name as the original template.
8. Delete the instance from the parent template file.
9. Save the parent template file.

Create a Document Part Template Library in an HTML Template File

HTML template packages created by copying the DOM API default HTML template package contains a document part template library file named `docpart_templates.html`.



The `docpart_templates.html` file contains default document part templates whose names and content are indicated by HTML markup conventions defined by the DOM

API. You can modify these templates and add your own templates by editing this file, using the markup conventions.

You can also create a template part library file in an HTML template that you create from scratch. In this case, you must ensure that the file observes the HTML markup conventions that the DOM API uses to indicate the name and content of a document part template in a document part library. To ensure this, copy the default template part library file into the template you create from scratch.

Add a Template to an HTML Document Part Template Library File

1. Unzip the template package containing the part template library file.
2. Open the file, named `docpart_templates.html` by default, in an HTML or text editor.
3. Create the following HTML markup in the `<body>` element of the file.

```
<div class="Template">
  <div class="TemplateName">
     <span class="TemplateName">TEMPLATE_NAME</span>
  </div>
  <div class="TemplateBody">
     TEMPLATE_BODY
  </div>
</div>
```

4. Replace `TEMPLATE_NAME` with a unique name for the template. Use this name in the constructor of a DOM `DocumentPart` object to be based on this template.
5. Replace `TEMPLATE_BODY` with HTML markup that defines the fixed content and holes of the template.
6. Save the library file.
7. Repackage the template.

See Also

Classes

`mlreportgen.dom.DocumentPart`

Related Examples

- “Fill in the Blanks in a Report Form” on page 11-27

- "Create a Microsoft Word Template" on page 11-92
- "Add Holes in a Microsoft Word Template" on page 11-93
- "Create an HTML Template" on page 11-101
- "Add Holes in an HTML Template" on page 11-102

More About

- "Form-Based Reporting" on page 11-26

Object-Oriented Report Creation

Note: This section assumes that you are familiar with object-oriented programming in MATLAB. For information on object-oriented programming in MATLAB, see “Object-Oriented Programming” in the MATLAB documentation.

The DOM API supports an object-oriented approach to creating report programs. With this approach, you subclass the DOM `Document` and `DocumentPart` classes to create document and document part classes tailored to your report application. You then create instances of these classes to generate a report.

Related Examples

- “Simplify Filling in Forms” on page 11-37

Simplify Filling in Forms

The object-oriented approach allows you to exploit the DOM `fill` method to simplify form-based reporting. The `fill` method is intended to be used with instances of classes derived from `mlreportgen.dom.Document` or `mlreportgen.dom.DocumentPart` class. It assumes that for each hole in a document or document part template the derived class defines a method having the following signature:

```
fillHoleId(obj)
```

The `HoleID` part of the signature is the ID of a hole defined by the document or document part template. The `obj` argument is an instance of the derived class. For example, supposed that a template defines a hole named `Author`. Then the derived class would define a method name `fillAuthor` to fill the `Author` hole. Assuming that the derived class defines methods for filling the holes, the `fill` method moves from the first hole in the document or part to the last, invoking the corresponding `fillHoleId` method to fill each hole.

The `fill` method eliminates the need for a report program to loop explicitly through the holes in a document or document part's template. The report need only invoke the document or part `fill` method. For example, suppose that you have derived a report class, name `MyReport`, from the `mlreportgen.dom.Document` class and that this derived class defines methods for each of the holes defined by the report template, based on data supplied in its constructor. Then, you would need only three lines to generate an instance of `MyReport`:

```
function makeReport(rptdata)
rpt = MyReport(rptdata);
fill(rpt);
close(rpt);
```

For an example of a forms-based, object-oriented report program, in the **Examples** pane of the MATLAB Report Generator documentation, open the Object-Oriented Report example.

See Also

Functions
`mlreportgen.dom.Document.moveToNextHole`

Classes
`mlreportgen.dom.DocumentPart`

Related Examples

- “Use Subforms in a Report” on page 11-29
- “Fill in the Blanks in a Report Form” on page 11-27
- “Create a Microsoft Word Template” on page 11-92
- “Add Holes in a Microsoft Word Template” on page 11-93
- “Create an HTML Template” on page 11-101
- “Add Holes in an HTML Template” on page 11-102

More About

- “Form-Based Reporting” on page 11-26

Create and Format Text

In this section...

Create Text

You can create text by appending a string to a document, paragraph, table entry, or list item. The DOM `append` function converts the string to a `Text` object, appends it, and returns the `Text` object. Use the `Text` object to format the text. You can also create a text object directly and append it to a document. This example:

- Creates the `Text` object `t1` by appending the string `'Hello'` to the document
- Uses a `Text` constructor to create a `Text` object and append the text `'World'` to the document

```
import mlreportgen.dom.*
doc = Document('mydoc','html');

t1 = append(doc,'Hello');

append(doc, Text('World'));

close(doc);
rptview('mydoc','html');
```

Create Special Characters

You can define special characters, such as the British pound symbol, to include in a report by creating an `mlreportgen.dom.CharEntity` object. Specify a name of a character entity listed at http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references. For example:

```
import mlreportgen.dom.*;
d = Document('test','html');
```

```
p = Paragraph(CharEntity('pound'));
append(d,p);
append(p,'3');

close(d);
rptview('test','html');
```

Append HTML or XML Markup

To append HTML markup to an HTML document or Microsoft Word XML markup to a Word document, use an `mlreportgen.dom.RawText` object. This is useful for creating HTML or Word elements that the DOM API does not support directly, such as the HTML `div` element. This example shows how to create a `RawText` object to append HTML markup.

```
import mlreportgen.dom.*;
d = Document('test','html');

append(d,RawText('<div id = toc> </toc>'));

close(d);
rptview('test','html');
```

Format Text

You can format text programmatically, using either DOM format objects or `Text` object format properties. You can also use Word and HTML template styles. For information about these formatting techniques and format inheritance, see “Report Formatting Approaches” on page 11-20.

Format Text Programmatically

You can use format objects to format `Text` objects or format properties to specify commonly used text formats. This example uses:

- A `FontFamily` format object to specify the primary and backup font
- The `Bold` format property to specify text weight

```
import mlreportgen.dom.*;
d = Document('test','html');

t = append(d, 'Bold Arial text');
```

```
fontFamily = FontFamily('Arial');
fontFamily.BackupFamilyNames = {'Helvetica'};
t.Style = {fontFamily};

t.Bold = true;

close(d);
rptview('test','html');
```

Use these format objects and format properties to format text.

Formatting	Format Object	Format Property
Font	`FontFamily`	`FontFamilyName`
Backup font (HTML only)	`FontFamily`	n/a
Complex script font (for example, Arabic)	`FontFamily`	n/a
East Asian font	`FontFamily`	n/a
Font size	`FontSize`	`FontSize`
Foreground color	`Color`	`Color`
Background color	`BackgroundColor`	`BackgroundColor`
Bold	`Bold`	`Bold`
Italic	`Italic`	`Italic`
Subscript or superscript	`VerticalAlign`	n/a
Strike through	`Strike`	`Strike`
Underline type (single, double, etc.)	`Underline`	`Underline`
Underline color	`Underline`	n/a
Preserve white space	`WhiteSpace`	`WhiteSpace`

Format Text Using Microsoft Word Style Sheets

You can format text using styles defined in the Word template used to generate the report.

To define a text style in a Word template, start by using these steps.

1. Open the Word template used with the report.
2. Open the **Styles** pane.
3. Click the **Manage Styles** button  .
4. Click **New Style**.
5. In the Create New Style from Formatting dialog box, set **Style type** to `Character` or `Linked (paragraph and character)`.

For more information about working with Word styles, see “Modify Styles in a Microsoft Word Template” on page 11-96.

Format Text Using HTML Style Sheets

You can format text using a style defined in the HTML template used to generate the report. Apply a template style to a `Text` object either as the second argument in a `Text` object constructor or by setting the `StyleName` property to a template style.

For an HTML report, use a `span` style. For example:

```
span.Pass {
  font-family: "Times New Roman", Times, serif;
  color: green;
}
```

For more information about using HTML styles with DOM objects, see “Modify Styles in an HTML Template” on page 11-104 .

Apply a Style to a Text Object

Apply a template style to a `Text` object either as the second argument in a `Text` object constructor or by setting the `StyleName` property to a template style. For example, suppose you have defined styles named `Body`, `Pass`, and `Fail` in the template for your report. You can then apply the styles as follows.

```
import mlreportgen.dom.*;
passed = rand(1) >= 0.5;
rpt = Document('MyReport','html','MyTemplate');

t1 = Text('Test status: ');
t1.StyleName = 'Body';
t1.WhiteSpace = 'preserve';
```

```
if passed
  status = 'Passed';
  statusStyle = 'Pass';
else
  status = 'Failed';
  statusStyle = 'Fail';
end

t2 = Text(status,statusStyle);
statusPara = Paragraph(t1);
append(statusPara,t2);
append(rpt, statusPara);

close(rpt);
rptview(rpt.OutputPath);
```

Override Template Formats

You can use programmatic formats to override the formats defined in a template-based style. For example, suppose you define a style named `AlertLevel` in your template and set the color to be green by default. You can override the style in your report program to set a color based on the current alert level. For example:

```
t = Text('Danger!','AlertLevel');
t.Color = 'red';
```

See Also

Classes

`mlreportgen.dom.Bold` | `mlreportgen.dom.CharEntity` | `mlreportgen.dom.FontFamily` | `mlreportgen.dom.FontSize` | `mlreportgen.dom.Italic` | `mlreportgen.dom.Strike` | `mlreportgen.dom.Text` | `mlreportgen.dom.Underline`

Related Examples

- “Add Content to a Report” on page 11-11

More About

- “Report Formatting Approaches” on page 11-20

Create and Format Paragraphs

In this section...

Create a Paragraph

You can create a paragraph by using an `mlreportgen.dom.Paragraph` constructor with a text string. For example:

```
p = Paragraph('Text for a paragraph');
```

You can also specify these DOM objects in a `Paragraph` object constructor.

- `mlreportgen.dom.Text`
- `mlreportgen.dom.ExternalLink`
- `mlreportgen.dom.InternalLink`
- `mlreportgen.dom.LinkTarget`
- `mlreportgen.dom.Image`

Create a Heading

You can use an `mlreportgen.dom.Heading` object to create a paragraph that you want to appear in the table of contents of a document (see "Create a Table of Contents" on page 11-74). Specify the heading level as the first argument in the `Heading` object constructor, followed by the heading content. Optionally, as a third argument, you can specify the name of a paragraph style defined in the template used to generate your report.

This example creates a heading with the text `Chapter 1: System Overview` and specifies the heading to appear at the top level in a table of contents.

```
h1 = Heading(1,'Chapter 1: System Overview');
```

Format a Paragraph

You can format a paragraph programmatically, using DOM format objects or format properties. You can also use Word and HTML template styles. For information about these formatting techniques and format inheritance, see “Report Formatting Approaches” on page 11-20.

Note: You can use the same format objects and properties for `Heading` objects as you do for `Paragraph` objects.

Format a Paragraph Programmatically

You can use format objects to format `Paragraph` objects or format properties to specify commonly used paragraph formats. This example uses:

- An `OuterMargin` format object to specify the margins for the paragraph
- The `HAlign` format property to center the paragraph

```
import mlreportgen.dom.*;
doc = Document('test','html');

p = Paragraph('Indent a half inch and space after 12 points.');
p.Style = {OuterMargin('0.5in','0in','0in','12pt')};
append(doc,p);

p = Paragraph('Centered paragraph');
p.HAlign = 'center';
append(doc,p);

close(doc);
rptview('test','html');
```

Use these format objects and format properties to format a paragraph.

Formatting	Format Object	Format Property
Font	`FontFamily`	`FontFamilyName`
Backup font (HTML only)	`FontFamily`	n/a
Complex script font (for example, Arabic)	`FontFamily`	n/a

Formatting	Format Object	Format Property
East Asian font	FontFamily	n/a
Font size	FontSize	FontSize
Foreground color	Color	Color
Background color	BackgroundColor	BackgroundColor
Bold	Bold	Bold
Italic	Italic	Italic
Subscript or superscript	VerticalAlign	n/a
Strike through	Strike	Strike
Underline type (single, double, etc.)	Underline	Underline
Underline color	Underline	n/a
Create border around paragraph	Border	n/a
Preserve white space	WhiteSpace	WhiteSpace
Indent a paragraph	OuterMargin	OuterLeftMargin
Indent first line of paragraph	FirstLineIndent	FirstLineIndent
Hanging indent	FirstLineIndent	n/a
Space before and after paragraph	OuterMargin	n/a
Space to right of paragraph	OuterMargin	n/a
Space between paragraph and its bounding box	InnerMargin	n/a
Space between paragraph lines	LineSpacing	n/a
Align paragraph left, center, right	HAlign	HAlign
Start paragraph on next page	PageBreakBefore	n/a
Keep with next paragraph	KeepWithNext	n/a

Formatting	Format Object	Format Property
Keep paragraph on same page	`KeepLinesTogether`	n/a
Eliminate widows and orphans	`WidowOrphanControl`	n/a
Table of contents level of paragraph	`OutlineLevel`	`OutlineLevel`

Format a Paragraph Using Microsoft Word Style Sheets

You can format a paragraph using a style defined in the Word template used to generate the report.

To define a paragraph style in a Word template, start by using these steps.

1. Open the Word template used with the report.
2. Open the **Styles** pane.
3. Click the **Manage Styles** button .
4. Click **New Style**.
5. In the Create New Style from Formatting dialog box, set **Style type** to `Character` or `Linked (paragraph and character)`.

For more information about working with Word styles, see “Modify Styles in a Microsoft Word Template” on page 11-96.

Format a Paragraph Using HTML Style Sheets

You can format using a style in defined in the HTML template used to generate the report.

For an HTML report, define the style as a `p` style. For example:

```
p.BodyPara {
  font-family: "Times New Roman", Times, serif;
  font-style: normal;
  font-size: 11pt;
  color: black;
  margin-left: 0.5in;
```

```
}
```

For more information about using HTML styles with DOM objects, see “Modify Styles in an HTML Template” on page 11-104.

Apply a Style to a Paragraph Object

Apply a template style to a `Paragraph` object either as the second argument in a `Paragraph` object constructor or by setting the `StyleName` property to a template style. For example, suppose you have defined styles named `BodyPara` and `TableTitle` in the template for your report. This example specifies a style name in a `Paragraph` constructor and in a `Paragraph` object `StyleName` format property, using the `TableTitle` style defined in `MyTemplate`.

```
import mlreportgen.dom.*;
rank = 5;
rpt = Document('MyReport','html','MyTemplate');

p = Paragraph('Here is a magic square or rank 5:','BodyPara');
append(rpt,p);

p = Paragraph(sprintf('Rank %d MagicSquare',rank));
p.StyleName = 'TableTitle';
append(rpt,magic(rank));

close(rpt);
rptview(rpt.OutputPath);
```

Override Template Formats

You can use programmatic formats to override the paragraph formats defined in a template-based paragraph style. For example, suppose you define a paragraph style named `BodyPara` in your Word template and set the `KeepWithNext` property to off. You can override the style in your report program to keep a particular paragraph on the same page with the next paragraph. For example:

```
import mlreportgen.dom.*;
rpt = Document('MyReport','docx','MyTemplate');

p = Paragraph('Keep this body paragraph with next.','BodyPara');
p.Style = {'KeepWithNext'};
append(rpt,p);

p = Paragraph('Next paragraph.');
```

```
append(rpt, p);

close(rpt);
rptview(rpt.OutputPath);
```

See Also

Classes

`mlreportgen.dom.Bold` | `mlreportgen.dom.FontFamily` | `mlreportgen.dom.FontSize` | `mlreportgen.dom.Italic` | `mlreportgen.dom.KeepLinesTogether` | `mlreportgen.dom.KeepWithNext` | `mlreportgen.dom.LineSpacing` | `mlreportgen.dom.PageBreakBefore` | `mlreportgen.dom.Paragraph` | `mlreportgen.dom.Strike` | `mlreportgen.dom.Text` | `mlreportgen.dom.Underline`

Related Examples

- “Add Content to a Report” on page 11-11

More About

- “Report Formatting Approaches” on page 11-20

Create and Format Lists

In this section...

You can add two kinds of lists to a report:

- Unordered (bulleted)
- Ordered (numbered)
- Multilevel (lists that contain ordered or unordered lists in any combination)

Create an Unordered List

You can create an unordered list from a numeric or cell array or from scratch.

- Creating a list from a cell array allows you to include items of different types in the list.
- Creating a list from scratch is useful for including multiple objects in a list item.

Create an Unordered List from an Array

You can create an unordered list by appending a one-dimensional numeric or cell array to a document (or document part). The `append` function converts the array to an `mlreportgen.dom.UnorderedList` object, appends the object to the document, and returns the object, which you can then format. In the cell array, you can include strings, numbers, and some DOM objects, such as a `Text` object. For a list of DOM objects you can include, see `mlreportgen.dom.ListItem`.

```
import mlreportgen.dom.*;
d = Document('myListReport','html');

t = Text('third item');
append(d,{'first item',6,t,'fourth item'});
```

```
close(d);
rptview('myListReport','html');
```

You can also create an unordered list from an array by including the array in an `UnorderedList` object constructor.

```
import mlreportgen.dom.*;
d = Document('unorderedListReport','html');

ul = UnorderedList({Text('item1'),'item 2',3});
append(d,ul);

close(d);
rptview('unorderedListReport','html');
```

Create an Unordered List from Scratch

You can create an unordered list from scratch by creating `mlreportgen.dom.ListItem` objects and appending them to an `UnorderedList` object.

```
import mlreportgen.dom.*;
d = Document('unorderedListReport','html');

li1 = ListItem('Rank 3 magic square:');
table = append(li1,Table(magic(3)));
table.Border = 'inset';
table.Width = '1in';
li2 = ListItem('second item');
li3 = ListItem('third item');

ul = UnorderedList();
append(ul,li1);
append(ul,li2);
append(ul,li3);

append(d,ul);

close(d);
rptview('unorderedListReport','html');
```

Create an Ordered List

You can create an ordered list from a numeric or cell array or from scratch.

- Creating an ordered list from a cell array allows you to include items of different types in the list.
- Creating a list from scratch is useful for including multiple objects in a list item.

Create an Ordered List from an Array

You can create an unordered list from a numeric array or cell array by including the array in an `mlreportgen.dom.OrderedList` object constructor. In the cell array, you can include strings, numbers, and some DOM objects, such as a `Text` object. For a list of DOM objects you can include, see `mlreportgen.dom.ListItem`.

```
import mlreportgen.dom.*;
d = Document('orderedListReport','html');

t = Text('step 1');
ol = OrderedList({t,'step 2','step 3'});
append(d,ol);

close(d);
rptview('orderedListReport','html');
```

Create an Ordered List from Scratch

You can create an unordered list from scratch by creating `mlreportgen.dom.ListItem` objects and appending them to an `OrderedList` object.

```
import mlreportgen.dom.*;
d = Document('orderedListReport','html');

li1 = ListItem('Create a rank 3 magic square:');
p = append(li1,Paragraph('>> magic(3)'));
p.FontFamilyName = 'Courier New';
li2 = ListItem('step 2');
li3 = ListItem('step 3');

ol = OrderedList();
append(ol,li1);
append(ol,li2);
append(ol,li3);

append(d,ol);

close(d);
rptview('orderedListReport','html');
```

Create a Multilevel List

A multilevel list is an ordered or unordered list whose list items contain ordered or unordered lists. You can create lists that have as many as nine levels.

You can create multilevel lists either from cell arrays or from scratch. Creating a multilevel list from scratch is useful for creating list items that contain multiple paragraphs, paragraphs and tables, and other combinations of document elements.

Create a Multilevel List from a Cell Array

You can use any of these approaches to create a multilevel list from a cell array.

- Nest one-dimensional cell arrays representing sublists in a one-dimension cell array representing the parent list.

  ```
  import mlreportgen.dom.*;
  d = Document('orderedListReport','html');

  ol = OrderedList({'step 1','step 2',...
       {'option 1','option 2'},...
       'step 3'});
  append(d,ol);

  close(d);
  rptview('orderedListReport','html');
  ```
- Include list objects as members of a one-dimensional cell array representing the parent list. Use this approach to create ordered sublists from cell arrays.

  ```
  d = Document('myListReport','html');

  append(d,{'first item',OrderedList({'step 1','step 2'}),'second item'});

  close(d);
  rptview('myListReport','html');
  ```
- Combine the nested cell array and nested list object approaches.

Create a Multilevel List from Scratch

You can create a multilevel list from scratch by appending child lists to parent lists.

```
import mlreportgen.dom.*;
```

```
d = Document('orderedListReport','html');

ol = OrderedList({'Start MATLAB', ...
     'Create a rank 3 or 4 magic square:'});
optionList = UnorderedList;
li = ListItem('>> magic(3)');
table = append(li,Table(magic(3)));
table.Width = '1in';
append(optionList, li);
li = ListItem('>> magic(4)');
table = append(li,Table(magic(4)));
table.Width = '1in';
append(optionList,li);
append(ol, optionList);
append(ol, ListItem('Close MATLAB'));,
append(d,ol);
close(d);
rptview('orderedListReport','html');
```

Format Lists

You can use list styles defined in a report style sheet to specify the indentation of each level of a list and the type of bullet or the number format used to render list items. To use a template-defined list style to format a list, set the `StyleName` property of the list to the name of the style. For example:

```
import mlreportgen.dom.*;
d = Document('myListReport','html','MyTemplate');

list = append(d,{'first item',...
    OrderedList({'step 1','step 2'}),'second item'});
list.StyleName = 'MyListStyle';

close(d);
rptview('myListReport','html');
```

Note: A list style determines how list items are rendered regardless of the list type. If you do not specify a list style, the DOM API uses a default list style that renders the list according to type. For example, the default list style for unordered lists uses bullets to render list items. If you specify a list style for an `UnorderedList` object that numbers top-level items, the top-level items are numbered, even though the object type is unordered (bulleted).

Create a Word List Style

To define a list style in a Word template, select `List` as the style type in the Create New Style from Formatting dialog box (see “Add Styles to a Word Template” on page 11-97).

Create an HTML List Style

To define a list style in an HTML template cascading style sheet (CSS), use the `ul` element for unordered list styles and the `ol` element for ordered list styles. You can use the parent element selector (>) to define multilevel list styles. For example, this CSS code defines the appearance of a two-level unordered list that can contain ordered or unordered sublists.

```
ul.MyUnorderedList {
 list-style-type:disc;
}

ul.MyUnorderedList > ul {
 list-style-type:circle;
}

ul.MyUnorderedList > ol {
 list-style-type:decimal;
}
```

For information about editing a cascading style sheet (CSS), see documentation such as the W3Schools.com CSS tutorial.

See Also

Classes
`mlreportgen.dom.ListItem` | `mlreportgen.dom.OrderedList` | `mlreportgen.dom.UnorderedList`

Functions
`mlreportgen.dom.OrderedList.append`

Related Examples

- “Use Style Sheets” on page 11-21

Create and Format Tables

In this section...

Two Types of Tables

You can use the DOM API to create two types of tables that differ in structure.

- An informal table (i.e., a table) consists of rows that contain table entries.
- A formal table contains a header, a body, and a footer section. Each section contains rows that contain table entries.

Informal tables are useful for most of your reporting needs. Use formal tables for tables whose headers or footers contain multiple rows.

For details about informal tables, see:

- "Create a Table from a Two-Dimensional Array" on page 11-57
- "Create a Table Using the Table entry Function" on page 11-57
- "Create a Table from Scratch" on page 11-58
- "Format a Table" on page 11-59

For details about formal tables, see:

- "Create a Formal Table" on page 11-64
- "Format a Formal Table" on page 11-64

Create a Table from a Two-Dimensional Array

You can create a table by appending a two-dimensional numeric array or a cell array containing built-in MATLAB data (strings and numbers) and DOM objects (`Text`, `Table`, `Image`, etc.) to a document. The `append` function converts the array to a `Table` object, appends it to the document, and returns the `Table` object, which you can then format. You can also create a `Table` object directly by including a two-dimensional array in its constructor.

This example shows how to create a table from a numeric array and another table from a cell array of various object types. The cell array contains a magic square, which is rendered as an inner table. The cell array also includes a `Text` object constructor that uses the `AlertLevel` template style.

```
import mlreportgen.dom.*;
doc = Document('test');

table1 = append(doc,magic(5));
table1.Border = 'single';
table1.ColSep = 'single';
table1.RowSep = 'single';

ca = {'text entry',Paragraph('a paragraph entry'); ...
       Text('Danger!','AlertLevel'),magic(4)};
table2 = Table(ca);
append(doc,table2);

close(doc);
rptview(doc.OutputPath);
```

Create a Table Using the Table entry Function

You can use the `entry` function with a `Table` object to add content to a table entry or to format an entry. This approach is useful when you need to format table entries individually. For example:

```
import mlreportgen.dom.*;
doc = Document('test');

a = magic(5);
[v,i] = max(a);
[v1,i1] = max(max(a));
```

```
table = Table(a);

text = table.entry(i(i1),i1).Children(1);
text.Color = 'red';
append(doc,table);

close(doc);
rptview(doc.OutputPath);
```

Create a Table from Scratch

You can create a table from scratch by creating `TableEntry` objects, appending them to `TableRow` objects, and appending the `TableRow` objects to a `Table` object. This approach is useful when you need to create table entries that span multiple columns or rows that have a different number of entries. This example shows how to create a table with four columns and two rows. In the first table row, the second entry spans the second and third columns.

```
import mlreportgen.dom.*;
doc = Document('test');

table = Table(4);
table.Border = 'single';
table.ColSep = 'single';
table.RowSep = 'single';

row = TableRow;
append(row, TableEntry('entry 11'));
te = TableEntry('entry 12-13');
te.ColSpan = 2;
te.Border = 'single';
append(row, te);
append(row, TableEntry('entry 14'));
append(table,row);

row = TableRow;
for c = 1:4
  append(row, TableEntry(sprintf('entry 2%i', c)));
end
append(table,row);

append(doc,table);
```

```
close(doc);
rptview(doc.OutputPath);
```

Format a Table

You can format a table programmatically, using DOM format objects or format properties. You can also use Word and HTML template styles. For information about these formatting techniques and format inheritance, see “Report Formatting Approaches” on page 11-20.

Format a Table Programmatically

You can use format objects to format tables or use `Table` format properties to specify commonly used table formats. This example uses:

- `Border`, `ColSep`, and `RowSep` format objects to specify a red table border and the green column and row separators
- The `Width` format property to specify the table width

```
import mlreportgen.dom.*;
doc = Document('test','html');

table = Table(magic(5));
table.Style = {Border('inset','red','3px'), ...
               ColSep('single','green','1px'), ...
               RowSep('single','green','1px')};

table.Width = '50%';

append(doc, table);

close(doc);
rptview(doc.OutputPath);
```

Use these format objects and format properties to format a table.

Formatting	Format Object	Format Property
Width of table	`Width`	`Width`
Color of table background	`BackgroundColor`	`BackgroundColor`
Create border around table	`Border`	`Border`

Formatting	Format Object	Format Property
Color of border	`Border`	`BorderColor`
Thickness of border	`Border`	`BorderWidth`
Create left, right, top, or bottom table border	`Border`	n/a
Collapse table and table entry borders (HTML)	`BorderCollapse`	`BorderCollapse`
Create column separator	`ColSep`	`ColSep`
Column separator color	`ColSep`	`ColSepColor`
Column separator thickness	`ColSep`	`ColSepWidth`
Create row separator	`RowSep`	`RowSep`
Row separator color	`RowSep`	`RowSepColor`
Row separator thickness	`RowSep`	`RowSepWidth`
Indent table from left margin	`OuterMargin`	`OuterLeftMargin`
Space before or after table	`OuterMargin`	n/a
Space to right of table	`OuterMargin`	n/a
Align table left, right, or center	`HAlign`	`HAlign`
Specify table entry flow direction (left-to-right or right-to-left)	`FlowDirection`	`FlowDirection`
Resize table columns to fit contents	`ResizeToFitContents`	n/a

Format Table Entries

A `Table` object has properties that allow you to specify the same format or set of formats for all of its entries.

Formatting	Table Object Property
Align entries vertically (top, middle, bottom)	`TableEntriesValign`

Formatting	Table Object Property
Align entries horizontally (left, right, center)	`TableEntriesValign`
Create space (padding) between entry boundary and content	`TableEntriesInnerMargin`
Apply a set of format objects to all table entries	`TableEntriesStyle`

Keep a Table and Its Title on the Same Page

Use the `KeepLinesTogether` and `KeepWithNext` paragraph formats to keep a table title and the table together on the same page. This example creates a table title, creates table content, and makes the table header row bold, using table entry indexing. To keep the table on the same page, the code specifies `KeepLinesTogether` and `KeepWithNext` for all rows except the last row. The last row has only `KeepLinesTogether` set and not `KeepWithNext`. This prevents the table from being forced to stay with the paragraph that follows.

```
import mlreportgen.dom.*
rpt = Document('test','docx');

p = Paragraph('Table 1');
p.Style = {Bold,KeepLinesTogether,KeepWithNext};
append(rpt, p);

ca = {Paragraph('Col 1'),Paragraph('Col 2'); ...
      Paragraph('data 11'),Paragraph('Data 12'); ...
      Paragraph('data 21'),Paragraph('Data 22')};

ca{1,1}.Children(1).Bold = true;
ca{1,2}.Children(1).Bold = true;

for r = 1:2
    for c = 1:2
        ca{r, c}.Style = {KeepLinesTogether,KeepWithNext};
    end
end

for c = 1:2
    ca{3, c}.Style = {KeepLinesTogether};
end
```

```
 append(rpt, ca);

close(rpt);
rptview(rpt.OutputPath);
```

Format a Table Using Microsoft Word Style Sheets

You can format tables using an existing Word styles in a template or a template style that you modify or add.

To define a table style in a Word template, start by using these steps.

1. Open the Word template used with the report.
2. Open the **Styles** pane.
3. Click the **Manage Styles** button .
4. Click **New Style**.
5. In the Create New Style from Formatting dialog box, set **Style type** to `Table`.

For more information about using Word styles with DOM objects, see “Modify Styles in a Microsoft Word Template” on page 11-96.

Format a Table Using an HTML Style Sheet

You can format tables using an HTML style defined in the template used to generate the report.

To define a table style in an HTML template, define the style as a `table` style. For example:

```
table.MyTable {
  border-bottom-color: rgb(128, 128, 128);
  border-bottom-width: thin;
  border-collapse: collapse;
}
```

Tip Use the CSS parent selector (>) to specify the format of the children of a table to be formatted with your table style. For example, this CSS code specifies the format of the table entries (`td` elements) of a table whose style is `MyTable`.

```
table.MyTable > tr > td {
```

```
  font-family: Arial, Helvetica, sans-serif;
  font-size: 11pt;
  text-align: center;
}
```

Apply a Table Style to a Table

Once you have defined a table style in a template, you can apply it to a `Table` object in your report program either as the second argument in the `Table` object constructor or by setting it to the `StyleName` property of the `Table` object. For example, suppose you have defined styles named `BodyPara`, `TableTitle`, and `RuledTable` in the template for your report. This example specifies style names in a `Paragraph` constructor, in the `StyleName` property of a Paragraph object, and in a `Table` constructor.

```
import mlreportgen.dom.*;
rank = 5;
rpt = Document('MyReport','html','MyTemplate');

p = Paragraph('Here is a magic square or rank 5:','BodyPara');
append(rpt,p);

p = Paragraph(sprintf('Rank %d MagicSquare',rank));
p.StyleName = 'TableTitle';

append(rpt,Table(magic(rank),'RuledTable'));

close(rpt);
rptview(rpt.OutputPath);
```

You can use programmatic formats to override the styles defined in a template-based table style. For example, suppose you define a table style named `UnruledTable` in your template to create tables without any borders or column or row separators. You can then override the style in your report program to draw a frame around a table.

```
import mlreportgen.dom.*;
rpt = Document('MyReport','html','MyTemplate');

table = Table(magic(5),'UnruledTable');
table.Border = 'single';
append(rpt,table);

close(rpt);
rptview(rpt.OutputPath);
```

Create a Formal Table

To create a formal table, use the same basic approaches as with an informal table, except that you must use an mlreportgen.domlFormalTable constructor to construct a formal table. The constructor optionally accepts a two-dimensional numeric array or a cell array of MATLAB data for the body, header, and footer sections.

If you choose to build a formal table completely or partially from scratch, you can use the FormalTable object functions appendHeaderRow and appendBodyRow to append rows to the table header and footer sections. The FormalTable.append function appends a row to the body section. Alternatively, you can access a section using the Header, Body, or Footer properties of the FormalTable object.

```
import mlreportgen.dom.*
d = Document('test');

t = FormalTable({'a','b';'c','d'});

r = TableRow();
append(r,TableEntry('Column 1'));
append(r,TableEntry('Column 2'));
append(t.Header,r);

append(d,t);

close(d);
rptview(d.OutputPath);
```

Format a Formal Table

You can format a formal table programmatically, using DOM format objects or format properties. You can also use Word and HTML template styles. For information about these formatting techniques and format inheritance, see "Report Formatting Approaches" on page 11-20.

Format a Formal Table Programmatically

You can format a formal table programmatically the same way you format an informal table. The format objects and properties that apply to an informal table also apply to formal tables. In addition, you can format the header, body, and footer sections of an informal table programmatically. If you specify a format for the table and one of its sections, the value you specify for the section overrides the value you specify for the table as a whole. Not all formal table formats apply to formal table sections. For example, you cannot indent a header, body, or footer section independently of the containing table. In other words, the `OuterLeftMargin` property does not apply to formal table sections.

Apply Table Styles to a Formal Table and Its Sections

Use the same procedure for defining formal table styles in Word and HTML templates as you use for defining informal table styles.

You can apply a table style to a formal table and to each of its sections. If you apply a table style to the table itself and to one of its section (for example, the header), the section style overrides the table style.

Note: If you apply a table style to one or more sections of a Word formal table, you must specify the widths of each of the table columns. Otherwise, the columns of the sections may not line up.

Create and Format Table Rows

If you need to build a table from scratch, you can use the `TableRow` constructor to create the rows. Format the rows and then append the rows to a table that you are building.

Create a Table Row

The `mlreportgen.dom.TableRow` constructor takes no arguments and returns a `TableRow` object. You can then create and append `TableEntry` objects to the object to complete the row construction. Once you construct the row, you can add the row to the table, using the `append` function. This example creates a two-column table with two rows.

```
import mlreportgen.dom.*
rpt = Document('test');

table = Table(2);

row = TableRow();
append(row,TableEntry('Col1'));
append(row,TableEntry('Col2'));
append(table,row);
```

```
row = TableRow();
append(row,TableEntry('data11'));
append(row,TableEntry('data12'));
append(table,row);

append(rpt,table);

close(rpt);
rptview(rpt.OutputPath);
```

Specify the Format of a Table Row

Use these format objects and format properties to format a table row.

Row Height Formatting	Format Object	Format Property
Specify the exact height of a row	`RowHeight`	`Height`
Specify the minimum height of row (Word only)	`RowHeight`	n/a
Cause this row to repeat as header row when a table flows across pages	`RepeatAsHeaderRow`	n/a
Allow this row to straddle a page boundary	`AllowBreakAcrossPages`	n/a

Format Table Columns

To format table columns, you can use `mlreportgen.dom.TableColSpecGroup` objects, either alone or with `mlreportgen.dom.TableColSpecGroup` objects. Use a `TableColSpecGroup` object to specify the format of a group of adjacent table columns. Use a `TableColSpec` object to override, for some table columns, some or all of the formats of a column group. In this example, the `TableColSpecGroup` property specifies a column width of 0.2 inches and green text. The `TableColSpec` overrides those formats for the first column, specifying a width of 0.5 inches and bold, red text.

```
import mlreportgen.dom.*
rpt = Document('test');

rank = 5;
table = Table(magic(rank));
```

```
table.Border = 'single';
table.BorderWidth = '1px';

grps(1) = TableColSpecGroup;
grps(1).Span = rank;
grps(1).Style = {Width('0.2in'),Color('green')};

specs(1) = TableColSpec;
specs(1).Span = 1;
specs(1).Style = {Width('0.5in'),Bold,Color('red')};

grps(1).ColSpecs = specs;

table.ColSpecGroups = grps;
append(rpt,table);

close(rpt);
rptview('test','html');
```

Create and Format Table Entries

If you need to build a table from scratch, you can use the `mlreportgen.dom.TableEntry` constructor to create table entries. You can then format the table entries and add them to table rows, which you can then add to the table you are building. If you need to format entries in a table that you have created from a cell array, you can use the `TableEntry` or `TableRow` function `entry` to gain access to an entry, which you can then format.

Create a Table Entry

Use a `TableEntry` constructor to create a table entry. You can optionally use the constructor to specify these kinds of entry content:

- Char array
- Any of these kinds of DOM objects:
  - `Paragraph`
  - `Text`
  - `Image`
  - `Table`
  - `OrderedList`

- `UnorderedList`
- `CustomElement`

Format Table Entries Programmatically

You can use format objects or `TableEntry` format properties to format a table entry programmatically.

Formatting	Format Object	Format Property
Create border around entry	`Border`	`Border`
Color of border	`Border`	`BorderColor`
Thickness of border	`Border`	`BorderWidth`
Create left, right, top, or bottom entry border	`Border`	n/a
Align entry content top, bottom, middle	`VAlign`	`VAlign`
Space between entry boundary and entry content	`InnerMargin`	`InnerMargin`
Space between entry content and its top, bottom, right, or left boundaries	`InnerMargin`	n/a
Cause entry to span multiple columns	`ColSpan`	`ColSpan`
Cause entry to span multiple rows	`RowSpan`	`RowSpan`

Format Table Entries Using Style Sheets

For HTML reports, you can use styles defined in an HTML template style sheet to format table entries. When defining a table entry style, use a `td` element selector. For example:

```
td.TableEntryWithBorder {
  border:5px solid red;
}
```

To apply a template-defined style to a table entry, set the `TableEntry` object `StyleName` property to the name of the style or specify the style name as the second argument to the `TableEntry` constructor. For example:

```
te = TableEntry('Hello World','TableEntryWithBorder');
```

See Also

Classes

mlreportgen.dom.AllowBreakAcrossPages | mlreportgen.dom.ColSep | mlreportgen.dom.FlowDirection | mlreportgen.dom.FormalTable | mlreportgen.dom.RepeatAsHeaderRow | mlreportgen.dom.ResizeToFitContents | mlreportgen.dom.RowHeight | mlreportgen.dom.RowSep | mlreportgen.dom.Table | mlreportgen.dom.TableBody | mlreportgen.dom.TableColSpec | mlreportgen.dom.TableColSpecGroup | mlreportgen.dom.TableEntry | mlreportgen.dom.TableFooter | mlreportgen.dom.TableHeader | mlreportgen.dom.TableHeaderEntry | mlreportgen.dom.TableRow

Functions

mlreportgen.dom.FormalTable.appendFooterRow | mlreportgen.dom.FormalTable.appendHeaderRow | mlreportgen.dom.TableRow.append

Related Examples

- "Add Content to a Report" on page 11-11

More About

- "Report Formatting Approaches" on page 11-20

Create Links

In this section...

Links

You can add these kinds of links to a report:

- External — Link to a location outside of the report, such as an HTML page or a PDF file. Use an `mlreportgen.dom.ExternalLink` object.
- Internal — Link to locations in the report. Use an `mlreportgen.dom.InternalLink` object.

Create a Link Target

To specify the link target for an `InternalLink` object, use value in the `Name` property of an `mlreportgen.dom.LinkTarget` object. When you construct an `ExternalLink` object, you can use an `LinkTarget` object `Name` value or a URL.

This example creates a link target called `home`, and uses `home` as the target for an internal link.

```
import mlreportgen.dom.*
d = Document('mydoc');

append(d,LinkTarget('home'));
append(d,InternalLink('home','Go to Top'));

close(d);
rptview('mydoc', 'html');
```

Create an External Link

Use an `mlreportgen.dom.ExternalLink` object to create an external link, specifying the link target and the link text.

```
import mlreportgen.dom.*
d = Document('mydoc');

append(d,ExternalLink('http://www.mathworks.com/','MathWorks'));

close(d);
rptview('mydoc','html');
```

Create an Internal Link

To set up links to a location in a report, append an `mlreportgen.dom.InternalLink` object to the document or document element. Use an `mlreportgen.dom.LinkTarget` object with the document element to link to. For example, you can include an `About the Author` link to a section that has the heading `Author's Biography`.

```
import mlreportgen.dom.*
d = Document('mydoc');

append(d,InternalLink('bio','About the Author'));
h = Heading(1,LinkTarget('bio'));
append(h,'Author''s Biography');
append(d,h);

close(d);
rptview('mydoc','html');
```

See Also

`mlreportgen.dom.ExternalLink` | `mlreportgen.dom.InternalLink` | `mlreportgen.dom.LinkTarget`

Related Examples

- "Create Image Maps" on page 11-81
- "Add Content to a Report" on page 11-11

More About

- "Report Formatting Approaches" on page 11-20

Create and Format Images

In this section...

Create an Image

To create an image to a report, create an `mlreportgen.dom.Image` object. You can append it to one of these document element objects:

- `Document`
- `Group`
- `Paragraph`
- `ListItem`
- `TableEntry`

For example, you can create a MATLAB figure, save it as an image, and add the image to a report.

```
import mlreportgen.dom.*
d = Document('imageArea','html');

p = Paragraph('Plot 1');
p.Bold = true;
append(d,p);

x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y);

saveas(gcf,'myPlot_img.png');

plot1 = Image('myPlot_img.png');
append(d,plot1);
```

```
close(d);
rptview(d.OutputPath);
```

For a list of supported image formats, see `mlreportgen.dom.Image`.

Resize an Image

To resize an image object, you can:

- Set the `Image.Height` and `Image.Width` properties.
- Use an `mlreportgen.dom.Height` or `mlreportgen.dom.Width` object in an `Image.Style` property definition.

For Microsoft Word reports, you can use an `mlreportgen.dom.ScaleToFit` object to scale an image so that it fits within the page margins.

Image Storage

Keep the original file until it has been copied into the document. The DOM API copies the contents of the source image file into the output document either when you append the `Image` object to the document (if you set the `Document.StreamOutput` property to `true`) or when you close the document.

Links from an Image

You can specify an area in an image to be a link. Clicking a link area in an image in an HTML browser opens the link. For details, see "Create Image Maps" on page 11-81.

See Also

`mlreportgen.dom.Height` | `mlreportgen.dom.Image` | `mlreportgen.dom.ScaleToFit` | `mlreportgen.dom.Width`

Related Examples

- "Add Content to a Report" on page 11-11

More About

- "Report Formatting Approaches" on page 11-20

Create a Table of Contents

In this section...
“Create a Microsoft Word Table of Contents” on page 11-74
“Create an HTML Table of Contents” on page 11-76
“Set Outline Levels of Section Heads” on page 11-78

Create a Microsoft Word Table of Contents

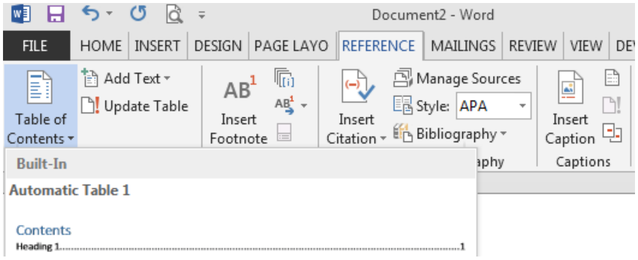
The DOM API relies on an automatic table-of-contents (TOC) generation feature of Word to generate a table of contents in a DOM Word report. With the Word TOC generation feature, you create an item called a TOC reference in a Word document where you want a TOC to appear. You create and set the outline line level of the paragraphs (typically section heads) that you want to include in the generated TOC. Finally, you have Word update the TOC to include the content of the paragraphs at the indicated outline level.

You use a very similar procedure for Word reports you create using the DOM API, except that you create the section heads programmatically instead of interactively. To generate a table of contents in a DOM Word report, perform these steps.

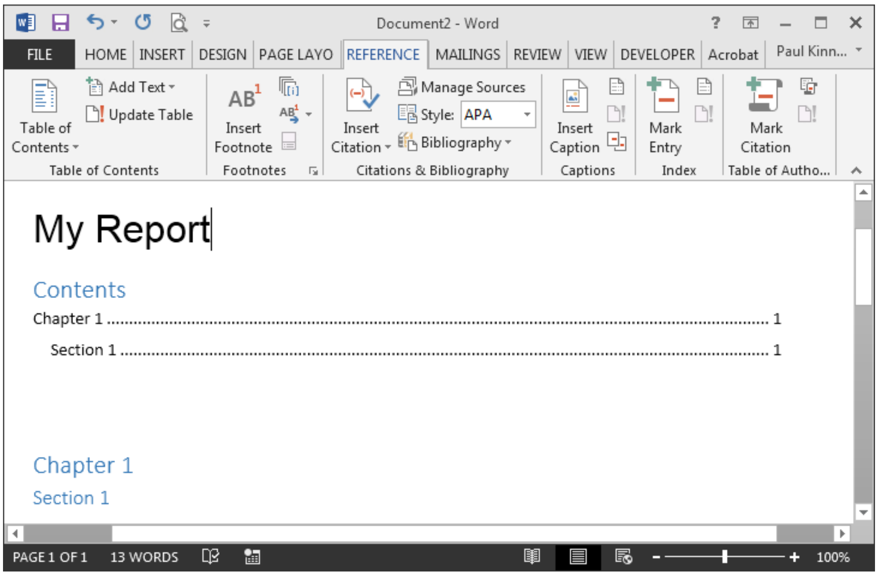
1. Create a table of contents reference in the Word template to specify where in the report to generate the TOC. See “Create a Word Table of Contents Reference” on page 11-74.
2. Set the outline levels of the section heads that you want to appear in the table of contents. See “Set Outline Levels of Section Heads” on page 11-78.
3. Update the generated document. See “Update the TOC in a Word Report” on page 11-75.

Create a Word Table of Contents Reference

1. Open the template in Word.
2. Click where you want to create the table of contents.
3. In the Word ribbon, select the **References** pane.
4. Select the **Table of Contents** button.



5 Select a TOC format option to generate a table of contents. For example, select the **Built-In** format option. The TOC appears.



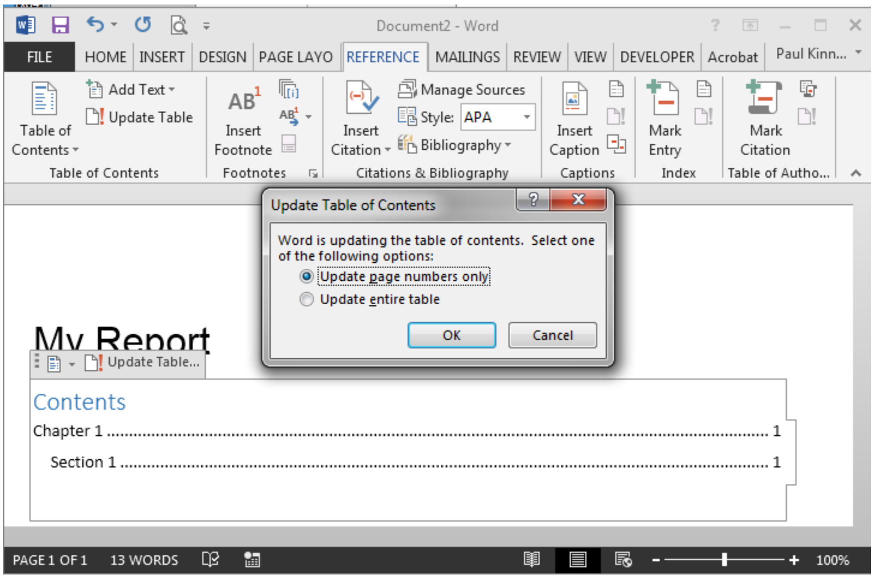
6 Save the template.

Update the TOC in a Word Report

After you generate a Word report with the DOM API, open the report in Word and update the document. Updating a document refers to the process of updating the parts of a Word document that Word itself generates, such as a TOC, page numbers, and so on, to reflect changes in the document content. The DOM `rptview` function causes Word to update a report after opening it. If you use `rptview` to display your document in Word, you do not need to do anything else to generate a TOC in your report.

However, if you open a newly generated report yourself in Word, without first using `rptview`, perform these steps to get the TOC to appear.

1. In the Word template, select the TOC reference.
2. Open the **Reference** pane and click the **Update Table** button.
3. In the Update Table of Contents dialog box, select **Update entire table** and click **OK**.



4. Save the report.

Create an HTML Table of Contents

The DOM API provides a JavaScript® script that you can include in an HTML report to generate a table of contents when you open the report in an HTML browser. To use this script to generate a TOC in an HTML report, perform these steps.

1. Create an HTML TOC placeholder element in the template. See “Create an HTML TOC Placeholder” on page 11-77
2. Set the outline levels of the section heads that you want to appear in the table of contents. See “Set Outline Levels of Section Heads” on page 11-78.
3. Open the generated report in a browser.

Create an HTML TOC Placeholder

An HTML TOC placeholder element is an HTML `div` element with an `id` attribute set to `toc`.

```
<div id="toc"/>
```

You can create a TOC placeholder in any of the following ways.

- “Use a Template That Contains a Placeholder” on page 11-77
- “Insert a Placeholder Programmatically Using a Custom Element” on page 11-77
- “Insert a Placeholder Programmatically Using a Document Part” on page 11-78

Use a Template That Contains a Placeholder

1. Create a copy of the DOM default HTML template.
2. Unzip the template using the `unzipTemplate` command.
3. In a text or HTML editor, open the template main document (typically named `root.html`).
4. In the root document, in the report location you want the TOC to appear, insert the following HTML markup:

   ```
   <div id="toc"></div>
   ```
5. Save the main document.
6. Zip the template, using the `unzipTemplate` command.
7. Set the outline levels of the section heads that you want to appear in the table of contents. See “Set Outline Levels of Section Heads” on page 11-78 .
8. Use the template to generate the report.

Next,

Insert a Placeholder Programmatically Using a Custom Element

If you use the DOM default HTML template (or a template based on the default template), you can create a placeholder programmatically in your report. For example:

```
import mlreportgen.dom.*;
d = Document('MyReport','html');
append(d,'My Report');
```

```
toc = CustomElement('div');
toc.CustomAttributes = {CustomAttribute('id', 'toc')};
append(toc, CustomText()); % Workaround for browser bug
append(d, toc);

append(d, Heading(1, 'Chapter 1'));
append(d, Heading(2, 'Section 1'));

close(d);
rptview(d.OutputPath);
```

Insert a Placeholder Programmatically Using a Document Part

The document part template library of the DOM default template contains a document part for creating a TOC placeholder. If you use this template or a template based on it, you can use the document part to insert a TOC placeholder in your report. For example:

```
import mlreportgen.dom.*;
d = Document('MyReport','html');
append(d,'My Report');

append(d, DocumentPart(d,'ReportTOC'));
append(d, Heading(1,'Chapter 1'));
append(d, Heading(2,'Section 1'));

close(d);
rptview(d.OutputPath);
```

Set Outline Levels of Section Heads

To generate a table of contents in a Word or HTML report, your program must set the outline levels of the section heads that you want to appear in the table. An outline level is a paragraph format property that specifies whether and at what level a paragraph's contents appear in a table of contents. For example, if a paragraph has an outline level of 1, its content appears at the top level of the generated table of contents. You can use up to nine distinct outline levels.

To set the outline level of paragraphs, use one of these approaches.

- "Use Template-Defined Styles to Set Outline Levels" on page 11-79
- "Use Format Objects to Set Outline Levels" on page 11-79
- "Use Heading Objects to Set Outline Levels" on page 11-79

Use Template-Defined Styles to Set Outline Levels

You can use styles defined in the report's template to set the outline level of a paragraph. For example, by default Word documents include a set of styles, `Heading 1`, `Heading 2`, and so on, that define outline levels for paragraphs you want to appear in a TOC. Your program can use these built-in styles to specify that paragraphs that serve as section heads appear in the TOC. The following example illustrates the use of template-defined styles to set the outline levels of section heads. This example assumes that the template `MyTemplate` includes a TOC reference.

```
import mlreportgen.dom.*;
d = Document('MyReport','docx','MyTemplate');

append(d, Paragraph('Chapter 1','Heading 1'));
append(d, Paragraph('Section 1','Heading 2'));

close(d);
rptview(d.OutputPath); % Updates the TOC
```

You can also use Word or an HTML editor to define your own heading styles and then use them to generate a report.

Use Format Objects to Set Outline Levels

You can use format objects to set outline levels. This example assumes that the template `MyTemplate` includes a TOC reference.

```
import mlreportgen.dom.*;
d = Document('MyReport','docx','MyTemplate');

h1 = {FontFamily('Arial'),FontSize('16pt'),OutlineLevel(1)};
h2 = {FontFamily('Arial'),FontSize('14pt'),OutlineLevel(2)};
p = append(d,Paragraph('Chapter 1'));
p.style = h1;
p = append(d, Paragraph('Section 1'));
p.style = h2;

close(d);
rptview(d.OutputPath); % Updates the TOC
```

Use Heading Objects to Set Outline Levels

You can use `mlreportgen.dom.Heading` objects to specify outline levels. A `Heading` object is a paragraph whose constructor specifies its outline level. You can use a `Heading` object alone or in combination with template-based styles or format object-based styles. This example assumes that the template `MyTemplate` includes a TOC reference.

```
import mlreportgen.dom.*;
d = Document('MyReport','docx','MyTemplate');

h1 = {FontFamily('Arial'),FontSize('16pt')};
h2 = {FontFamily('Arial'),FontSize('14pt')};
h = append(d, Heading(1,'Chapter 1'));
h.style = h1;
h = append(d, Heading(2,'Section 1'));
p.style = h2;

close(d);
rptview(d.OutputPath); % Updates the TOC
```

The `Heading` objects generate HTML `h1`, `h2` (and so on) elements. By using `Heading` objects in an HTML report, you can ensure that your report uses the default styles for headings implemented by the browser in which you display the report.

See Also

Functions
`rptview` | `unzipTemplate` | `zipTemplate`

Classes
`mlreportgen.dom.Heading`

Related Examples

- “Create a Microsoft Word Template” on page 11-92
- “Create an HTML Template” on page 11-101

Create Image Maps

You can specify areas of an image to be links. Clicking the link area in an image in an HTML browser opens the target. You can map different areas in an image to different link targets.

1 Create an `mlreportgen.dom.ImageArea` object for each image area that is to serve as a link. You can specify text to display if the image is not visible.

   You can specify an image area to have one of these shapes:

   - Rectangle
   - Circle
   - Polygon

   For details, see `mlreportgen.dom.ImageArea`.

2 Create an `mlreportgen.dom.ImageMap` object to associate the link areas with the image. Append the `ImageArea` objects to the `ImageMap` object.

For example, you can create a link from a plot image to documentation about plotting.

```
import mlreportgen.dom.*
d = Document('imageArea','html');

x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y);
annotation('textbox',[0.2,0.4,0.1,0.1],...
           'String','Help for plot function');
saveas(gcf,'plot_img.png');

plot1 = Image('plot_img.png');
append(d,plot1);

area1 = ImageArea(...
    ['http://www.mathworks.com/help/matlab/ref/' ...
    'plot.html?searchHighlight=plot'], ...
    'plot function help',240,450,463,492);

map = ImageMap();
plot1.Map = map;
append(map,area1);
```

```
close(d);
rptview(d.OutputPath);
```

See Also

Classes

`mlreportgen.dom.Image` | `mlreportgen.dom.ImageArea` | `mlreportgen.dom.ImageMap`

Functions

Related Examples

- “Add Content to a Report” on page 11-11

More About

- “Report Formatting Approaches” on page 11-20

Automatically Number Document Content

In this section...

You can automatically number document content, such as chapter, section, table, and figure headings. Append automatic numbering objects to the document where you want numbers to appear. Each automatic number is associated with a numbering stream that determines the value of each number in a sequence. Report generation replaces an automatic numbering object with a number based on its position in the document relative to other automatic numbers in the same stream. For example, the first automatic numbering object in a stream can be replaced by 1, the second by 2, and so on. You can use automatic numbering to create hierarchical numbering schemes such as Section 1.1., Section 1.2, and so on.

You can automatically number document content programmatically or by inserting automatic numbering fields in a Word template or numbering properties in an HTML template. For example, you can insert automatic numbering in a template for a document part that is appended repeatedly to a report.

Automatically Number Content Programmatically

To automatically number document content programmatically, do the following at each point in a document where you want an automatically generated number to appear.

1 Create an automatic numbering object, using the `mlreportgen.dom.AutoNumber` constructor. Specify the name of the associated automatic numbering stream in the constructor. For example, this line creates an automatic number belonging to the stream named `chapter`.

```
chapterNumber = AutoNumber('chapter');
```

Note: If the specified automatic numbering stream does not exist, the `AutoNumber` constructor creates a numbering stream having the specified name. The implicitly constructed stream renders automatic numbers as Arabic numerals. To use a stream with different properties, create the stream explicitly, using a `createAutoNumberStream` function of a `Document` object.

2 Append the `AutoNumber` to a `Text`, `Paragraph`, or `Heading` object that contains the text that precedes the automatic number.

```
append(chapHead,chapterNumber);
```

3 Append an `mlreportgen.dom.CounterInc` format object to the `Style` property of the content object that you want to automatically number. Appending a `CounterInc` object increments the stream associated with the automatic number when the paragraph or heading is output. The updated value replaces the `AutoNumber` object.

```
chapHead.Style = {CounterInc('chapter'), WhiteSpace('preserve')};
```

This script automatically numbers the chapter headings in a document.

```
import mlreportgen.dom.*;
d = Document('MyReport','html');

for rank = 3:5
    chapHead = Heading(1,'Chapter ','Heading 1');
    append(chapHead,AutoNumber('chapter'));
    append(chapHead,sprintf('. Rank %i Magic Square',rank));
    chapHead.Style = {CounterInc('chapter'), ...
                      WhiteSpace('preserve')};
    append(d,chapHead);
    table = append(d,magic(rank));
    table.Width = '2in';
end

close(d);
rptview(d.OutputPath);
```

Create Hierarchical Automatic Numbering

You can create hierarchical numbering schemes, such as 1.1, 1.2, 1.3, 2.1, 2.2, and so on. Use an `mlreportgen.dom.CounterReset` format object to reset a child automatic number to its initial value when its parent number changes. For example, this script uses a `CounterReset` format object to reset the chapter table number stream at the beginning of each chapter.

```
import mlreportgen.dom.*;
d = Document('MyReport','html');

for rank = 3:2:9
    chapHead = Heading(1,'Chapter ');
    append(chapHead, AutoNumber('chapter'));
```

```
    chapHead.Style = {CounterInc('chapter'), ...
                      CounterReset('table'), ...
                      WhiteSpace('preserve')};
    append(d,chapHead);

    for i = 0:1;
        tableHead = Paragraph('Table ');
        append(tableHead,AutoNumber('chapter'))
        append(tableHead,'.');
        append(tableHead, AutoNumber('table'));
        append(tableHead, ...
           sprintf('. Rank %i Magic Square',rank+i));
        tableHead.Style = {CounterInc('table'), ...
                           Bold, ...
                           FontSize('11pt'), ...
                           WhiteSpace('preserve')};
        append(d,tableHead);
        table = append(d,magic(rank+i));
        table.Width = '2in';
    end
end

close(d);
rptview(d.OutputPath);
```

Automatically Number Content Using Part Templates

You can automatically number a document by creating document parts based on templates containing Microsoft Word or HTML automatic numbering and repeatedly appending the parts to a document.

For example, suppose that you add a chapter part template `Chapter` to the part template library of the Word `MyReportTemplate.dotx` report template. This template uses a Word sequence (`SEQ`) field to number the chapter heading. The template also contains holes for the chapter title and the chapter content.

This script uses the chapter part template to create numbered chapters. The last statement in this script opens the report in Word and updates it. Updating the report causes Word to replace the `SEQ` fields with the chapter numbers.

```
import mlreportgen.dom.*
d = Document('MyReport','docx','MyReportTemplate');

for rank = 3:5
    chapterPart = DocumentPart(d,'Chapter');
    while ~strcmp(chapterPart.CurrentHoleId,'#end#')
       switch chapterPart.CurrentHoleId
           case 'ChapterTitle'
              append(chapterPart, ...
                 sprintf('Rank %i Magic Square',rank));
           case 'ChapterContent'
              table = append(chapterPart,magic(rank));
              table.Width = '2in';
       end
       moveToNextHole(chapterPart);
    end
    append(d, chapterPart);
end

close(d);
rptview(d.OutputPath);
```

You can use a similar approach to automatically number HTML reports, using the CSS `counter-increment` and `content` properties in the template for your report.

See Also

Functions

`mlreportgen.dom.Document.createAutoNumberStream` | `mlreportgen.dom.Document.getAutoNumberStream`

Classes

`mlreportgen.dom.AutoNumber` | `mlreportgen.dom.AutoNumberStream` | `mlreportgen.dom.CounterInc` | `mlreportgen.dom.CounterReset`

Display Report Generation Messages

In this section...

Report Generation Messages

The DOM API includes a set of messages that can display when you generate a report. The messages are triggered every time a document element is created or appended during report generation.

You can define additional messages to display during report generation. The DOM API provides these classes for defining messages:

- `ProgressMessage`
- `DebugMessage`
- `WarningMessage`
- `ErrorMessage`

The DOM API provides additional classes for handling report message dispatching and display. It uses MATLAB events and listeners to dispatch messages. A message is dispatched based on event data for a specified DOM object. For an introduction to events and listeners, see "Events and Listeners — Concepts".

Display DOM Default Messages

This example shows how to display the default DOM debug messages. Use a similar approach for displaying other kinds of DOM report messages.

1 Create a message dispatcher, using the `MessageDispatcher.getTheDispatcher` method. Use the same dispatcher for all messages.

```
dispatcher = MessageDispatcher.getTheDispatcher;
dispatcher.Filter.DebugMessagesPass = true;
```

2 Use the `MessageDispatcher.Filter` property to specify to display debug messages.

```
dispatcher.Filter.DebugMessagesPass = true;
```

3 Add a listener using the MATLAB `addlistener` function. Specify the dispatcher object, the source and event data, and a `disp` function that specifies the event data and format to use for the message.

```
l = addlistener(dispatcher,'Message', ...
      @(src, evtdata) disp(evtdata.Message.formatAsText));
```

4 Include a code to delete the listener. Place it after the code that generates the report.

```
delete(l);
```

This report displays debug messages.

```
import mlreportgen.dom.*;
d = Document('test','html');

dispatcher = MessageDispatcher.getTheDispatcher;
dispatcher.Filter.DebugMessagesPass = true;

l = addlistener(dispatcher,'Message', ...
      @(src, evtdata) disp(evtdata.Message.formatAsText));

open(d);

p = Paragraph('Chapter ');
p.Tag = 'chapter title';
p.Style = { CounterInc('chapter'),...
    CounterReset('table'),WhiteSpace('pre') };
append(p, AutoNumber('chapter'));
append(d,p);

close(d);
rptview('test','html');

delete(l);
```

Create and Display a Progress Message

This example shows how to create and dispatch a progress message. You can use a similar approach for other kinds of messages, such as warnings.

1. Create a message dispatcher.

   ```
   dispatcher = MessageDispatcher.getTheDispatcher;
   ```

2. Add a listener using the MATLAB `addlistener` function.

   ```
   l = addlistener(dispatcher,'Message', ...
         @(src, evtdata) disp(evtdata.Message.formatAsText));
   ```

3. Dispatch the message, using the `Message.dispatch` method. Specify the dispatcher object and the message to dispatch. Here the message is a debug message called `starting chapter`, and the `Document` object `d` is the source of the message.

   ```
   dispatch(dispatcher,ProgressMessage('starting chapter',d));
   ```

4. Include code to delete the listener, after the code that generates the report.

   ```
   delete(l);
   ```

This report uses this progress message.

```
import mlreportgen.dom.*;
d = Document('test','html');

dispatcher = MessageDispatcher.getTheDispatcher;

l = addlistener(dispatcher,'Message', ...
      @(src, evtdata) disp(evtdata.Message.formatAsText));

open(d);
dispatch(dispatcher,ProgressMessage('starting chapter',d));

p = Paragraph('Chapter ');
p.Tag = 'chapter title';
p.Style = { CounterInc('chapter'),...
    CounterReset('table'),WhiteSpace('pre') };
append(p, AutoNumber('chapter'));
append(d,p);

close(d);
rptview('test','html');

delete(l);
```

The MATLAB Command Window displays progress messages, including the `starting chapter` message, as well as the messages the DOM API dispatches by default.

See Also

Functions

`mlreportgen.dom.MessageDispatcher.dispatch` | `mlreportgen.dom.MessageDispatcher.getTheDispatcher` | `mlreportgen.dom.ProgressMessage.formatAsHTML` | `mlreportgen.dom.ProgressMessage.formatAsText` | `mlreportgen.dom.ProgressMessage.passesFilter`

Classes

`mlreportgen.dom.DebugMessage` | `mlreportgen.dom.ErrorMessage` | `mlreportgen.dom.MessageDispatcher` | `mlreportgen.dom.MessageEventData` | `mlreportgen.dom.MessageFilter` | `mlreportgen.dom.ProgressMessage` | `mlreportgen.dom.WarningMessage`

Compile a Report Program

If the MATLAB Compiler™ product is installed on your system, you can use it to compile your DOM-based report generation program. This allows you to share your report generation program with others who do not have MATLAB installed on their systems.

To enable someone who does not have MATLAB installed to run your compiled program, your program must execute the following statement before executing the first line of DOM code that it executes to generate a report:

```
makeDOMCompilable();
```

Create a Microsoft Word Template

Use one of these approaches to create a Word template for generating a report.

- Use `mlreportgen.dom.Document.createTemplate` to create a copy of the DOM API default Word template that you can then customize. For example,

  ```
  mlreportgen.dom.Document.createTemplate('mytemplate','docx');
  ```
- Use an existing Word template (for example, a report template for your organization) and customize the template to use with the DOM API.
- Create a Word template from scratch.

If you copy an existing template that is not based on the DOM API default Word template, apply any standard Word styles such as `Title`, `Heading 1`, `TOC 1`, `List 1`, `Emphasis`, etc. to an element in the template. You can apply the styles to placeholder content and then remove the content. That process creates instances of the standard styles in the template style sheet.

See the Word documentation for information about how to create templates and to copy styles from one template to another.

Related Examples

- "Add Holes in a Microsoft Word Template" on page 11-93
- "Modify Styles in a Microsoft Word Template" on page 11-96
- "Create an HTML Template" on page 11-101

Add Holes in a Microsoft Word Template

In this section...

Template holes are places in a template that a report script fills with generated content, supporting a forms-based report.

Note: To create holes in a Word template, use the Word **Developer** ribbon. If the **Developer** tab is not showing in your Word ribbon, add it to the ribbon.

1 In Word, select **File** > **Options**.

2 In the Word Options dialog box, select **Customize Ribbon**.
3 In the **Customize the Ribbon** list, select the **Developer** check box.

Inline and Block Holes

The DOM API supports two types of holes: inline and block.

- An inline hole is for document elements that a paragraph element can contain: `Text`, `Image`, `LinkTarget`, `ExternalLink`, `InternalLink`, `CharEntity`, `AutoNumber`.
- A block hole can contain the same kinds of document elements as an inline hole, as well as `Paragraph`, `Table`, `OrderedList`, `UnorderedList`, `DocumentPart`, and `Group` document elements.

Create an Inline Hole

1 Open the template in Word.
2 On the Word ribbon, select the **Developer** tab.
3 Click **Design Mode** to see the hole marks with the title tag after creating the hole.
4 Position the Word insertion mark at the point in the paragraph where you want to create an inline hole.

Tip If the hole is the only hole in a paragraph or is at the end of a paragraph:

a. Add several blank spaces at the end of the paragraph.
b. Insert the hole before the spaces.
c. Delete the extra spaces.

5. Click the **Rich Text Control** button Aa. Word inserts a rich text control at the insertion point.
6. Click the **Properties** button.
7. In the dialog box, in the **Title** field enter an ID for the hole and in **Tag** field enter `Hole`. Click **OK**. The hole ID appears on the rich text control.

Create a Block-Level Hole

Creating a block-level hole in a Word document is essentially the same as creating an inline hole. The main difference is that rich text content control must contain an (empty) paragraph instead of residing in a paragraph.

1. Open the template in the Word editor.
2. On the Word tool ribbon, select the **Developer** ribbon.
3. Click **Design Mode** to see the hole marks with the title tag after creating the hole.
4. Create an empty paragraph at the point where you want to create a block-level hole. If you are at the end of a document, create a second empty paragraph.
5. Select the empty paragraph.
6. Select the **Rich Text Control** button Aa. Word inserts a rich text control at the insertion point.
7. Click the **Properties** button.
8. In the dialog box, in the **Title** field enter an ID for the hole and in **Tag** field enter `Hole`. Click **OK**. The hole ID appears on the rich text control.

Set Default Text Style for a Hole

Your template can specify the name of a default style to use to format `Text` and `Paragraph` objects appended to a hole. If such an object does not specify a style name, the DOM API sets its `StyleName` property to the name of the default style, which must be a character or linked character and paragraph style defined in the template. Defining a default hole style eliminates the need to format hole content programmatically.

1. Open the template in Microsoft Word.
2. In the Word ribbon, select the **Developer** tab.
3. Click the hole whose default style name you want to specify.

   This step assumes that you have already created the hole. If have not create a hole, see "Inline and Block Holes" on page 11-93.
4. In the **Developer** ribbon, click **Properties**.
5. In the Content Control Properties dialog box, select the **Use a style to format text typed into the empty control** check box.
6. From the **Style** list, select a style to use an existing style or select **New Style** to create a new style to be used as the default style. In either case, the style must be a `Character` or a `Linked (paragraph and character)` style.
7. Click **OK** and save the template.

Related Examples

- "Modify Styles in a Microsoft Word Template" on page 11-96
- "Create an HTML Template" on page 11-101
- "Create and Format Tables" on page 11-56

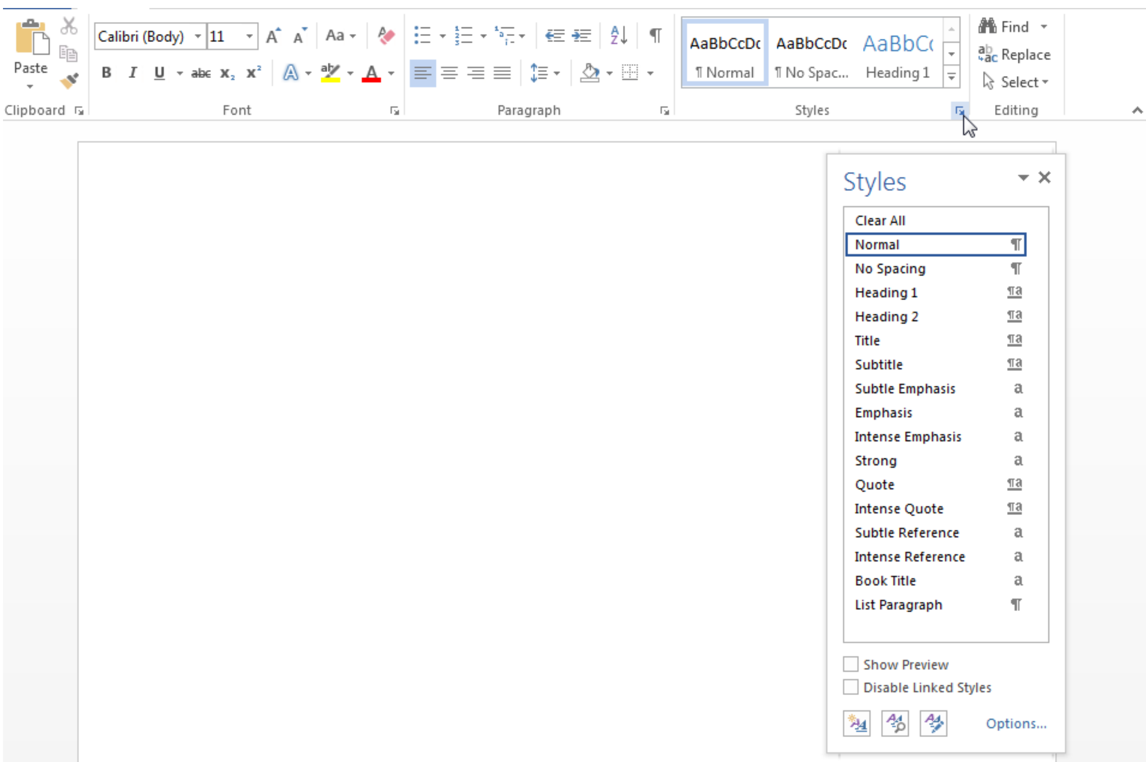
Modify Styles in a Microsoft Word Template

In this section...

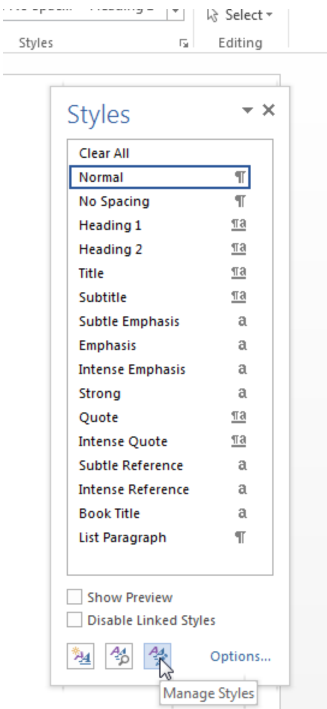
Edit Styles in a Word Template

You can customize or add format styles in a custom Word template.

1 In Word, open the Styles dialog box.



2 In the Style dialog box, click the **Manage Styles** button.



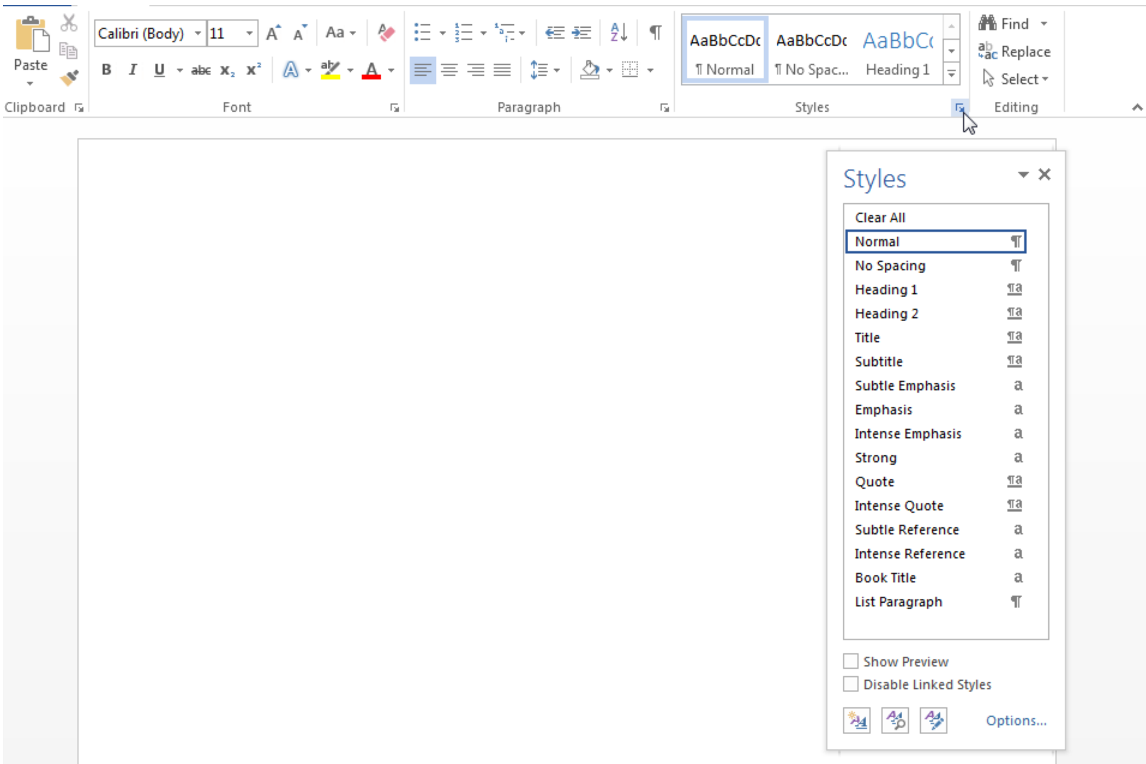
3. In the Manage Styles dialog box, click **Modify**. The Modify Style dialog box appears.
4. In the Modify Style dialog box, change any of the style definitions. For example, you could change the font family, font size, indentation, etc. To save your changes, click **OK** and close the dialog box.
5. In Word, save and close the template.

For more information about using Word styles, see the Microsoft Word documentation.

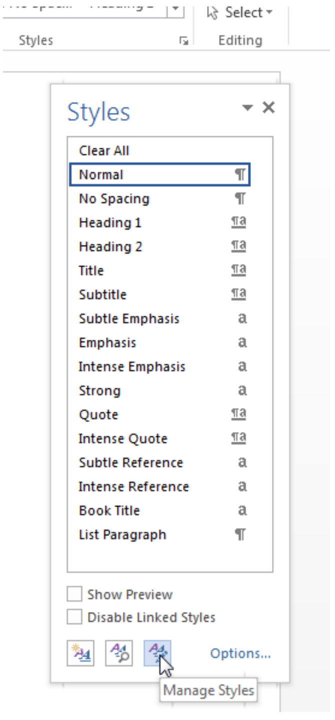
Add Styles to a Word Template

To add a new style to a template:

1. In Word, open the Styles dialog box.

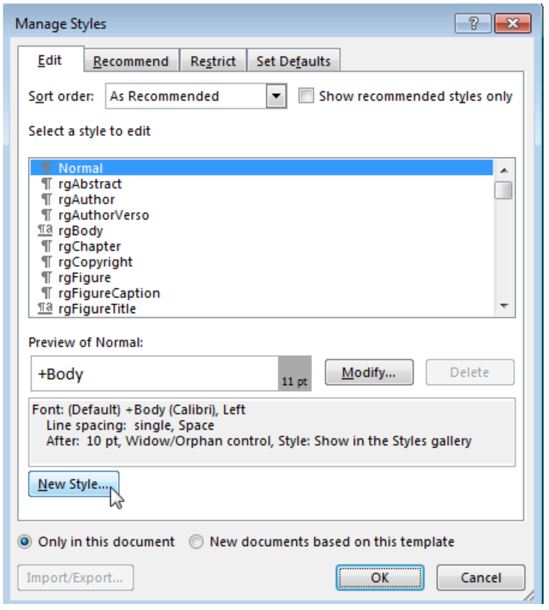


2 In the Style dialog box, click the **Manage Styles** button.



3 If applicable, select an existing style to use as a starting point for the new style.

4 Click the **New Style** button.



5 Specify a name for the new style and define the style characteristics. To save the new style definition, click **OK** and close the dialog box.

6 In Word, save and close the template.

Related Examples

- "Add Holes in a Microsoft Word Template" on page 11-93
- "Create an HTML Template" on page 11-101

Create an HTML Template

Use one of these approaches to create an HTML template for generating a report.

- Use `mlreportgen.dom.Document.createTemplate` to create a copy of the DOM API default HTML template that you can then customize. For example:

  ```
  mlreportgen.dom.Document.createTemplate('mytemplate','html');
  ```
- Use an existing HTML template (for example, a report template for your organization) and customize the template to use with the DOM API.
- Create an HTML template from scratch.

Edit a Zipped HTML Template

To edit a zipped HTML template, unzip it into a subfolder of the current folder, using the `unzipTemplate` function. For example, to unzip files for a template called `mytemplate`:

```
unzipTemplate('mytemplate')
```

To repackage a template after you edit it, use the `zipTemplate` function. For example, to package the `mytemplate.htmx` template in a subfolder called `mytemplate`, in the current folder:

```
zipTemplate('mytemplate.htmtx')
```

If you do not want to use the current folder, you can specify a path with the `unzipTemplate` and `zipTemplate` functions.

Related Examples

- “Add Holes in an HTML Template” on page 11-102
- “Modify Styles in an HTML Template” on page 11-104
- “Create a Microsoft Word Template” on page 11-92

Add Holes in an HTML Template

In this section...

Template holes are places in a template that a report script fills with generated content, supporting a forms-based report.

Inline and Block Holes

The DOM API supports two types of holes: inline and block.

- An inline hole is for document elements that a paragraph element can contain: `Text`, `Image`, `LinkTarget`, `ExternalLink`, `InternalLink`, `CharEntity`, `AutoNumber`.
- A block hole can contain a `Paragraph`, `Table`, `OrderedList`, `UnorderedList`, `DocumentPart`, and `Group`.

Create an Inline Hole

1. Unzip the template.
2. Open the `root.html` document of the template in an HTML or text editor.
3. Place the insertion point in the paragraph where you want to create a hole.
4. Add code that uses this pattern:

```
<p>
  <span>
    <div class=Hole" data-default-hole-style-name="STYLE_NAME"
     data-hole-id="HOLEID">
       <span class="HoldId:>HOLEID</span>
       <span class="HoleDesc">HOLE_DESCRIPTION</span>
     </div>
  </span>
</p>
```

Replace `STYLE_NAME` with the name of a default text (HTML `span` element) style to use for formatting `Text` objects appended to this hole. If a `Text` object appended to this hole does not specify a style name, the DOM API sets the text object `StyleName` property to the default style name. The style must be defined in the style sheet of the template. Defining in the template a default text style for hole content eliminates the need to format hole content programmatically.

Set `HOLEID` to the ID of the hole, and use `HOLE_DESCRIPTION` to describe the hole.

Templates based on the DOM API default HTML template contain a style sheet for holes that highlights the hole IDs when you display the template in a browser.

Create a Block Hole

1. Unzip the template.
2. Open the `root.html` document of the template in an HTML or text editor.
3. Position the insertion point at the desired location for the hole. You cannot set he insertion point inside a paragraph.
4. Add code that uses this pattern:

```
<div>
  <span>
    <div class=Hole" data-default-hole-style-name="STYLE_NAME"
     data-hole-id="HOLEID">
       <span class="HoldId:>HOLEID</span>
       <span class="HoleDesc">HOLE_DESCRIPTION</span>
     </div>
  </div>
```

   Replace `STYLE` with the name of a default paragraph (HTML `p` element) style to use for formatting text content appended to this hole. If you do not specify a style name for a `Text` object appended to this hole, the DOM API sets the text object `StyleName` property to the default style name. The template style sheet must define the default style. Defining a default paragraph style for hole content eliminates the need to format hole content programmatically.

   Set `HOLEID` to the ID of the hole, and use `HOLE_DESCRIPTION` to describe the hole.

Related Examples

- "Modify Styles in an HTML Template" on page 11-104
- "Create a Microsoft Word Template" on page 11-92

Modify Styles in an HTML Template

You can customize or add format styles in a custom HTML template.

1. In a text or HTML editor, open the `TEMPLATEROOT/Stylesheet/root.css` file.
2. In the Properties pane, click **Open stylesheet**.
3. In a text or HTML editor, edit the cascading style sheet (CSS).

   For information about editing a cascading style sheet, see documentation such as the W3Schools.com CSS tutorial.
4. Save the style sheet.

Related Examples

- "Add Holes in an HTML Template" on page 11-102
- "Create a Microsoft Word Template" on page 11-92

Create Microsoft Word Page Layout Sections

In this section...

You can divide a Word document into one or more sections, each with its own page layout. Page layout includes page margins, page orientation, and headers and footers.

Define Page Layouts in a Template

Every Word template has at least one page layout section. You can use Word to create as many additional sections as you need. For example, you may want to create in the main template of a report sections for your report's title page, table of contents, and chapters. See the Word documentation for information on how to create page layout sections in a Word template.

Navigate Template-Defined Sections

When you open a `Document` or `DocumentPart` object in a report program, the DOM API creates a hole and an associated `DOCXSection` property object for each section defined in the document or document part template. The hole ID for the first section is `#start`. The hole ID for the second section is `sect2`, and so on.

You can use the `moveToNextHole` function to move from section to section and from hole to hole within a section. At each section hole, the DOM API sets a document or document part `CurrentDOCXSection` property to the `DOCXSection` object associated with that object. The `DOCXSection` object reflects the properties of the current section, as defined in the template. For example, if you have defined the page orientation of that section to be portrait, the page orientation is set as portrait in the current `DOCXSection` object.

You can change the template-defined section properties programmatically. For example, the page orientation of the DOM default Word template is portrait. This example shows how to change the orientation to landscape to accommodate wide tables. The code swaps the height and width of the page to reflect the new page orientation.

```
import mlreportgen.dom.*
```

```
rpt = Document('test','docx');
open(rpt);

sect = rpt.CurrentDOCXSection;
pageSize = sect.PageSize;
pageSize.Orientation = 'landscape';

saveHeight = pageSize.Height;
pageSize.Height = pageSize.Width;
pageSize.Width = saveHeight;

table = append(rpt,magic(22));
table.Border = 'solid';
table.ColSep = 'solid';
table.RowSep = 'solid';

close(rpt);
rptview(rpt.OutputPath);
```

Create Sections Programmatically

You can use the `append` function of a `Document` or `DocumentPart` to create sections programmatically. To use the `append` function to add a section to a report, use this `append` syntax:

```
paraObj = append(rptObj,paraObj,docxSectionObj)
```

This us of the append function appends a paragraph to the report as the last paragraph of the current section and then starts a new section whose properties are defined by a `DOCXSection` object. For example, this script adds a landscape section to a report to accommodate a large magic square.

```
import mlreportgen.dom.*
rpt = Document('test','docx');

append(rpt,Heading(1,'Magic Square Report','Heading 1'));

sect = DOCXSection;
sect.PageSize.Orientation = 'landscape';
sect.PageSize.Height = '8.5in';
sect.PageSize.Width = '11in';
append(rpt,Paragraph('The next page shows magic square.'),sect);
```

```
table = append(rpt,magic(22));
table.Border = 'solid';
table.ColSep = 'solid';
table.RowSep = 'solid';

close(rpt);
rptview(rpt.OutputPath);
```

See Also

Classes
mlreportgen.dom.DOCXPageMargins | mlreportgen.dom.DOCXPageSize | mlreportgen.dom.DOCXSection

Related Examples

- “Create a Microsoft Word Template” on page 11-92

Create Page Footers and Headers

In this section...

You can create as many as three page headers and three page footers for a Word report section:

- One for the first page of the section
- One for even pages
- One for odd pages

You can create report page headers and footers programmatically or in the template used to create a report or report part. You can append content to both template-defined and programmatically defined headers and footers.

Create Page Headers and Footers in a Template

You can use Word to create page headers and footers in the main template of a report. For information on creating headers and footers, see the Word documentation.

When you open a report, the DOM API:

1. Reads the headers and footers from the template and converts them to `DOCXPageHeader` and `DOCXPageFooter` objects, respectively
2. Associates the headers and footer objects with the `DOCXSection` object that defines the properties of the section that contains the headers and footers
3. Adds the headers and footers to your report as your code navigates the sections defined by the template

As your report generation program navigates the sections, it can append content to the template-defined headers and footers.

Access Template-Defined Headers and Footers

To append content to a template-defined header or footer, you need to access it. Use the `CurrentDOCXSection` property of a `Document` or `DocumentPart` object to access the template-defined headers and footers for the current section of a document or document part.

The value of the CurrentDOCXSection property is a DOCXSection object whose PageHeaders and PageFooters properties contain a cell array of DOCXPageHeader and DOCXPageFooter objects corresponding to the template-defined headers and footers of the current section. The header cell array can contain as many as three header objects, depending on how many of the possible types of headers (first page, even page, odd page) you define for the section. The footers cell array similarly can contain as many as three footer objects. The objects can appear in any order in the cell array. Thus, to access a header or footer of a particular type, search the cell array to find the one you want to access.

Append Content to a Template-Defined Header or Footer

You can use the DOM API to append content to a template-defined header or footer that appears on every page in a section. To append content to a header or footer in the current section of a document or document part, first use the document or document part CurrentDOCXSection property to access the DOCXPagerHeader or DOCXPageFooter object. Then use the append method of a DOCXPageHeader or DOCXPageFooter object to append content to the header or footer. Because header and footer objects are a kind of document part object, you can append any kind of content to a page header or footer that you can append to a document part, for example, images and tables as well as paragraphs.

You can use holes in the header and footers of your main template to control the positioning of content that you append to the headers and footers. For example, this script appends today's date to a hole named Date on the first template-defined page header of the first section of a report. This example assumes that the Word template MyReportTemplate has one section that defines a first page, odd page, and even page header and footer.

```
import mlreportgen.dom.*;
d = Document('MyReport','docx','MyReportTemplate');
open(d);

sect = d.CurrentDOCXSection;

for i = 1:numel(sect.PageHeaders)
    if strcmpi(sect.PageHeaders(i).PageType,'first')
        firstPageHeader = sect.PageHeaders(i);
        while ~strcmp(firstPageHeader.CurrentHoleId,'#end#')
```

```
            switch firstPageHeader.CurrentHoleId
                case 'Date'
                    append(firstPageHeader,date);
            end
            moveToNextHole(firstPageHeader);
        end
        break;
    end
end

close(d);
rptview(d.OutputPath);
```

Generate Header and Footer Content That Varies from Page to Page

You cannot programmatically append to headers and footers content that varies from page to page, such as page numbers or running heads. To create content that varies, use Word fields, which enable automatic content generation. For example, to include a page number on each page of a section of your report, insert a page number field in the report template in the page footer of a section. For more information, see the Microsoft Word documentation.

Create Page Headers and Footers Programmatically

Perform these steps to create programmatically a page header or footer in the current section of a report.

1. Use the `DOCXPageHeader` or `DOCXPageFooter` constructor to create a page header or footer of the desired type (first page, odd page, even page, or odd and even page) based on a template that defines template form (the fixed content and holes for variable content).
2. Fill the holes in the header or footer with content.
3. Insert the header or footer in the array of page headers or footers of the current `DOCXSection` object.

This script creates a first page header from a template stored in the document part template library of a report.

```
import mlreportgen.dom.*;
d = Document('MyReport','docx','MyReportTemplate');
open(d);
```

```
pageHeaders(1) = DOCXPageHeader('first',d,'FirstPageHeader');

while ~strcmp(pageHeaders(1).CurrentHoleId,'#end#')
    switch pageHeaders(1).CurrentHoleId
        case 'Date'
            append(pageHeaders(1),date);
    end
    moveToNextHole(pageHeaders(1));
end

d.CurrentDOCXSection.PageHeaders = pageHeaders;

close(d);
rptview(d.OutputPath);
```

See Also

Functions

mlreportgen.dom.Document.createTemplate

Classes

mlreportgen.dom.Document | mlreportgen.dom.DocumentPart | mlreportgen.dom.DOCXPageFooter | mlreportgen.dom.DOCXPageHeader | mlreportgen.dom.DOCXSection

Related Examples

- “Create a Microsoft Word Template” on page 11-92
- “Create an HTML Template” on page 11-101