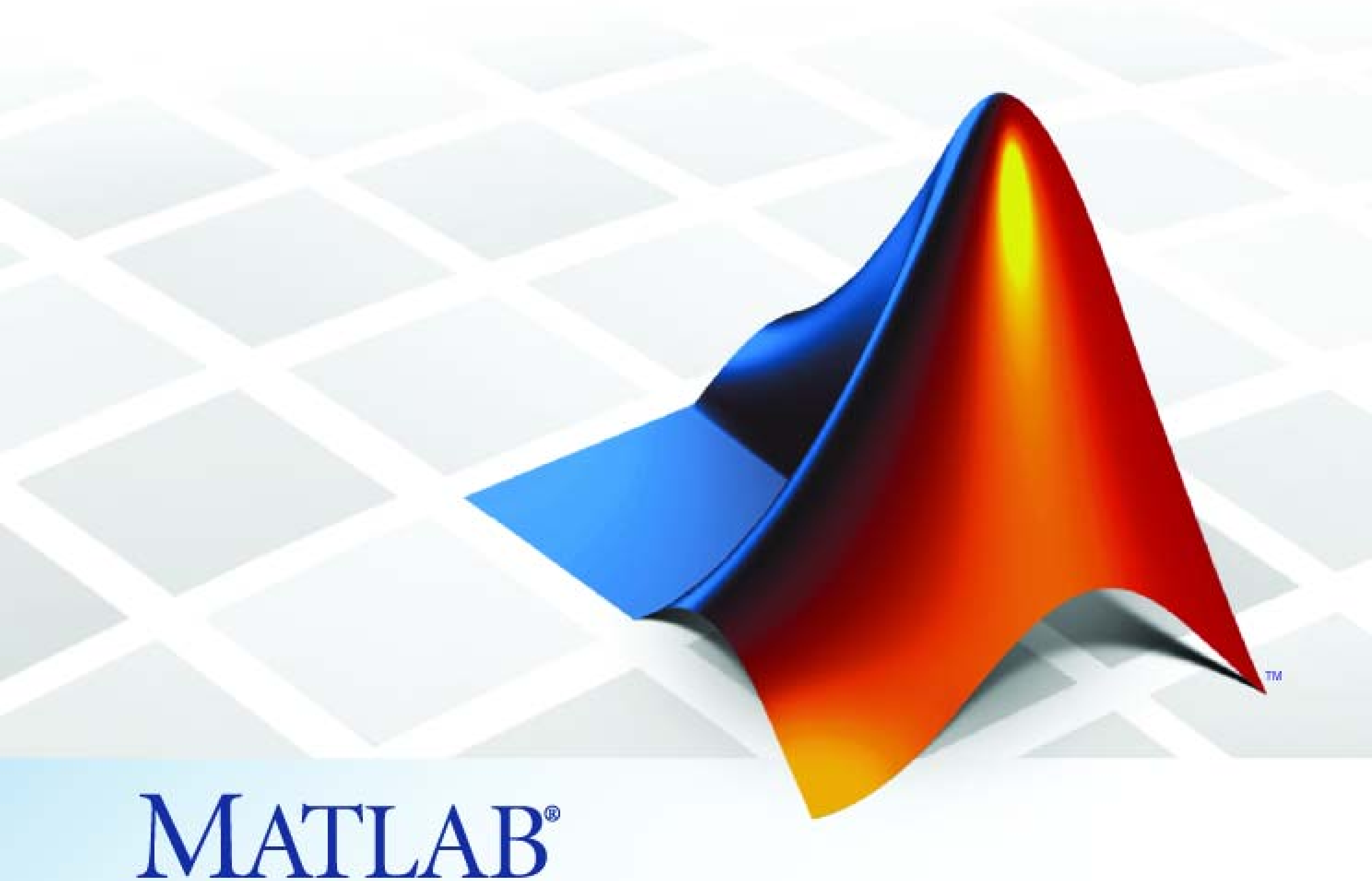
MATLAB® Report Generator™ 3

User's Guide



MATLAB®

The MathWorks™

Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

☎ 508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

MATLAB® Report Generator™ User's Guide

Trademarks

Patents

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 MATLAB® 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)

Contents

User's Guide

1 Getting Started

Examples

2 Creating MATLAB Reports

3 Generating Reports

4 Working with Components

5 Creating Custom Components

6 Creating Custom Stylesheets

7 Comparing XML Files

8 Component Reference

9 Components — Alphabetical List

10 Function Reference

11 Functions – Alphabetical List

A Examples

User’s Guide

Getting Started

Product Overview

You can use the MATLAB Report Generator software to:

- Document MATLAB software tasks, such as analyzing and visualizing data and developing algorithms
- Create a reusable report template that reflects your own styles and standards
- Keep your documentation and specifications up to date with your workflow
- Create user manuals that accurately capture your application capabilities and specifications
- Create comparison reports for XML files. See Chapter 7, “Comparing XML Files”.

The report generation process creates documentation (called *reports*) about MATLAB data and workflow. You can customize report templates to include:

- MATLAB code and its output
- Workspace data
- MATLAB graphics
- Logical components like `IF`, `THEN`, `ELSE`, and `WHILE` that allow conditional report generation
- Loops that perform a series of operations on multiple objects
- User-defined text, styles, and components

What Is the Report Explorer?

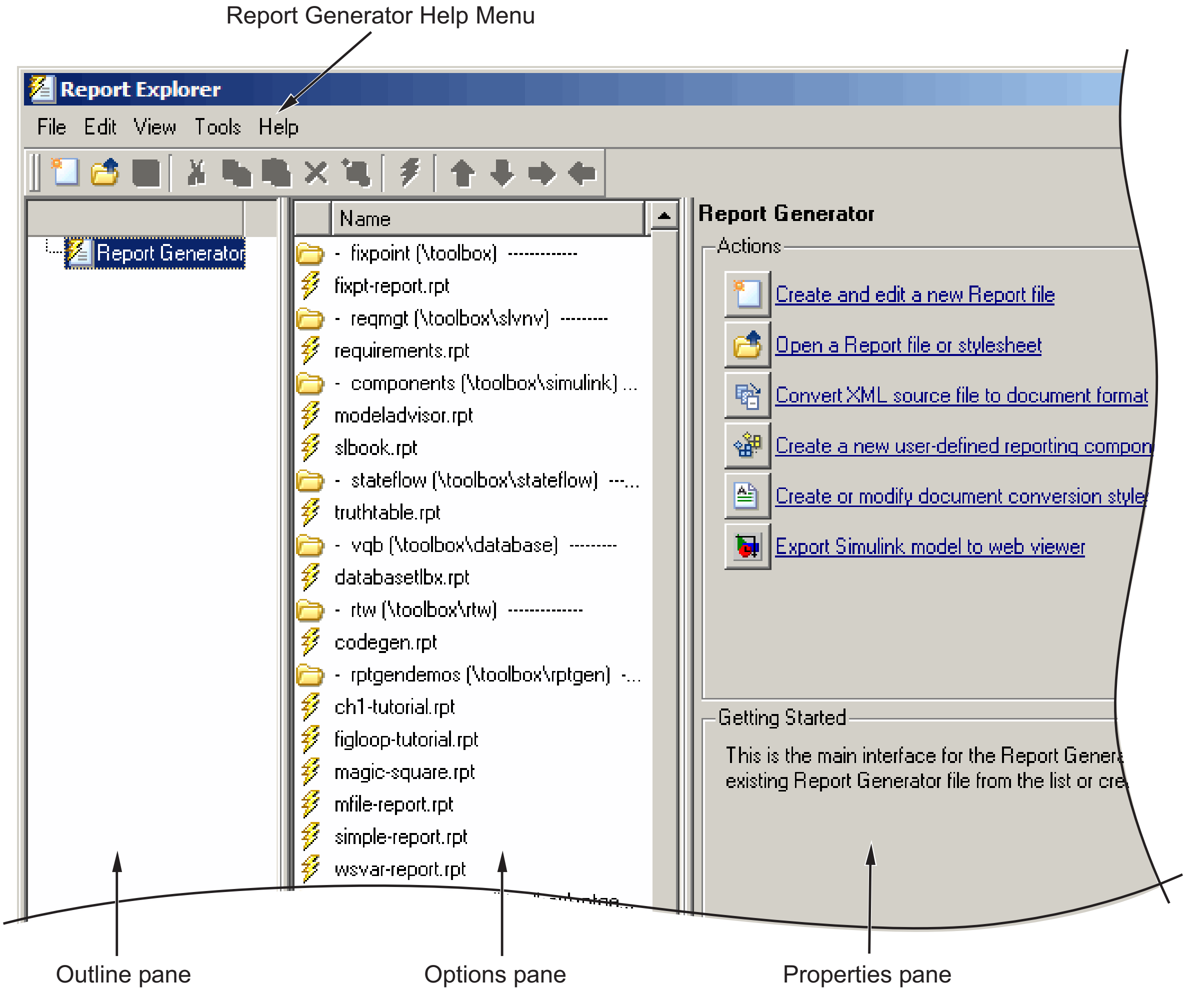
In this section...

About the Report Explorer

The *Report Explorer* is the MATLAB Report Generator and Simulink® Report Generator™ graphical user interface (GUI). It allows you to:

- Create and modify report templates.
- 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.


Report Generator Help Menu
Report Explorer
File Edit View Tools Help
Report Generator
Name
- fixpoint (\toolbox) ------------
fixpt-report.rpt
- reqmgt (\toolbox\slvnv) --------
requirements.rpt
- components (\toolbox\simulink) ...
modeladvisor.rpt
slbook.rpt
- stateflow (\toolbox\stateflow) ---...
truthtable.rpt
- vqb (\toolbox\database) --------
databasetlbx.rpt
- rtw (\toolbox\rtw) -------------
codegen.rpt
- rptgendemos (\toolbox\rptgen) -...
ch1-tutorial.rpt
figloop-tutorial.rpt
magic-square.rpt
mfile-report.rpt
simple-report.rpt
wsvar-report.rpt
Report Generator
Actions
Create and edit a new Report file
Open a Report file or stylesheet
Convert XML source file to document format
Create a new user-defined reporting compon
Create or modify document conversion style
Export Simulink model to web viewer
Getting Started
Outline pane
Options pane
Properties pane

This is the main interface for the Report Genera
existing Report Generator file from the list or cre

The Report Explorer has three panes:

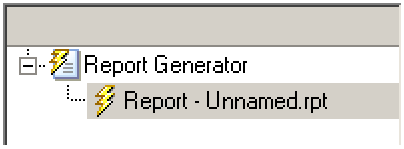
- The *Outline pane* on the left shows the hierarchy of components in currently opened report templates. Report components can reside within other report components, creating parent, child, and sibling relationships. For more information, see “The Outline Pane” on page 1-5.

- The *Options pane* in the middle lists the options available in the context of the Outline pane. If no report template is open, the Options pane lists available reports. When a report template is open, the Options pane lists components available to insert into it. When a stylesheet is open, the Options pane lists available attributes. For more information, see "The Options Pane" on page 1-6 .
- If no report template is open, the *Properties pane* on the right displays tasks the Report Explorer can perform. If a report template is open, the Properties pane displays the properties for the item that is currently selected in the Options pane. For more information, see "The Properties Pane" on page 1-7.

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.

The Outline Pane

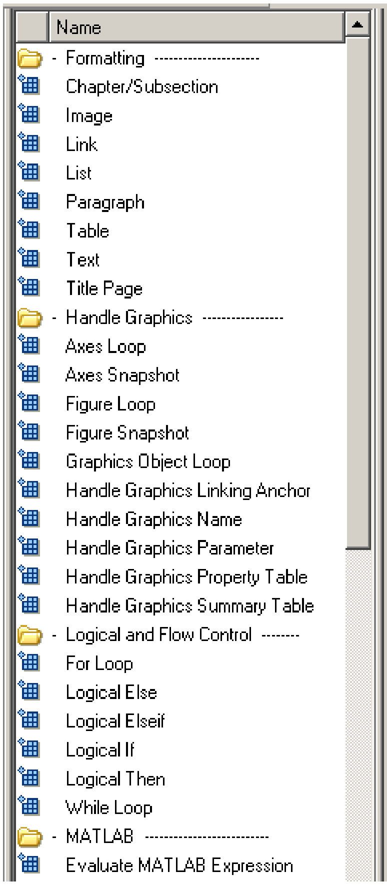
The Outline pane initially contains the top level of the report template.



The Options Pane

The Options pane lists by category components that are available for adding to report templates. In the following figure, the visible component categories are `Formatting`, `Handle Graphics`, `Logical and Flow Control`, and `MATLAB`. For details about report components, see Chapter 8, “Component Reference”.

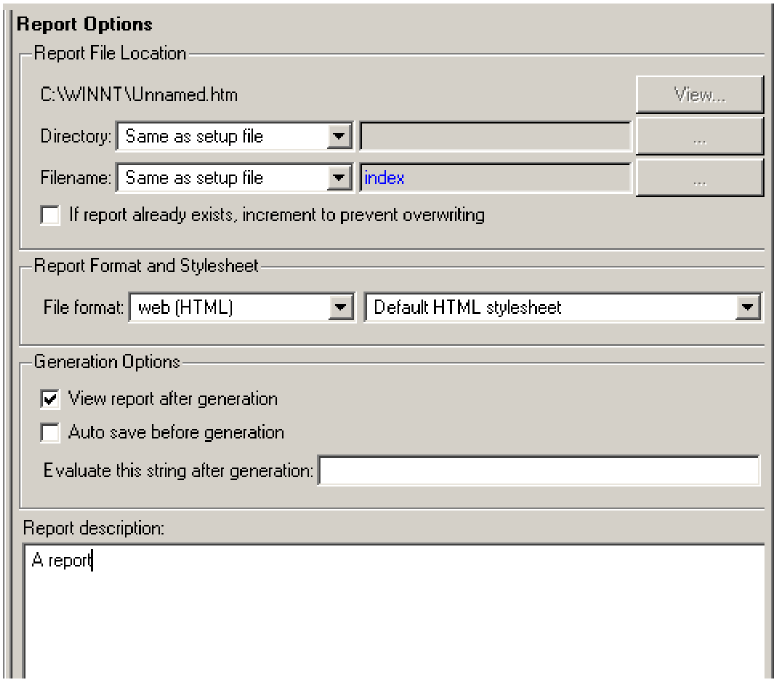
The folder icon indicates a component category. The blue square icon indicates a component.



Name
- Formatting ---------------------
Chapter/Subsection
Image
Link
List
Paragraph
Table
Text
Title Page
- Handle Graphics ----------------
Axes Loop
Axes Snapshot
Figure Loop
Figure Snapshot
Graphics Object Loop
Handle Graphics Linking Anchor
Handle Graphics Name
Handle Graphics Parameter
Handle Graphics Property Table
Handle Graphics Summary Table
- Logical and Flow Control --------
For Loop
Logical Else
Logical Elseif
Logical If
Logical Then
While Loop
- MATLAB -------------------------
Evaluate MATLAB Expression

The Properties Pane

After you create a report template, the Properties pane initially displays properties for the report template as a whole.



Report Options

Report File Location

C:\WINNT\Unnamed.htm

Directory: Same as setup file

Filename: Same as setup file index

□ If report already exists, increment to prevent overwriting

Report Format and Stylesheet

File format: web (HTML) Default HTML stylesheet

Generation Options

☑ View report after generation

□ Auto save before generation

Evaluate this string after generation:

Report description:

A report

MATLAB Report Generator Workflow

1. Open the Report Explorer.

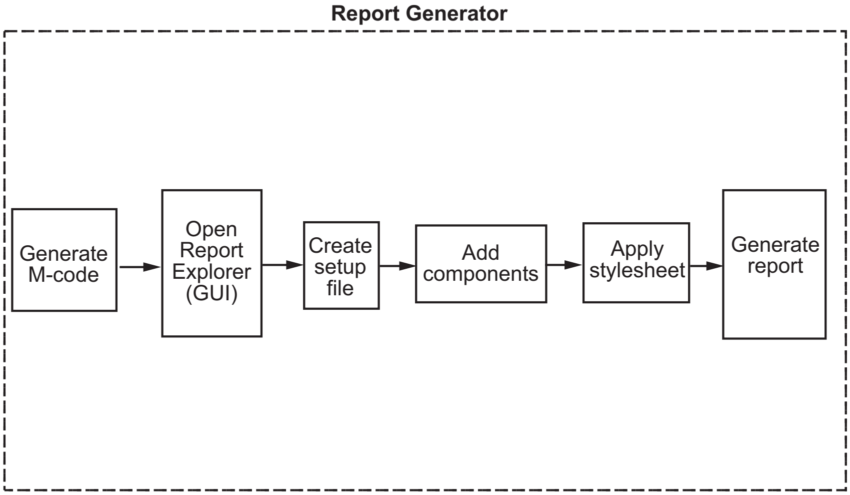
2. Create a report template. For details about report templates, see Chapter 2, "Creating MATLAB Reports" in this documentation.

3. Add existing components to the report template or create your own custom components. For details on using components, see Chapter 4, "Working with Components" in this documentation.

4. Choose an existing stylesheet or create a custom stylesheet to apply styles and standards to the report. For details on stylesheets and attributes, see Chapter 6, "Creating Custom Stylesheets" in this documentation.

5. Generate the report.

The following figure illustrates a typical MATLAB Report Generator workflow.



How MATLAB Report Generator and MATLAB Software Interact

The MATLAB Report Generator and MATLAB software interact to create reports. You can access the Report Explorer GUI from the MATLAB command line.

The following table describes these interactions in detail.

User Interface	MATLAB Report Generator Interaction	Description
Report Explorer	The Report Explorer is the MATLAB Report Generator graphical user interface (GUI). For more information, see "What Is the Report Explorer?" on page 1-3.	Use the Report Explorer to edit existing report templates, components, stylesheets, and attributes, or to customize your own.
MATLAB command line	Enter commands at the MATLAB command line to: • Start the Report Explorer • Create and modify report template files • Apply stylesheets • Specify output formats for reports • Generate reports	The following MATLAB commands work with the MATLAB Report Generator software: • `report` — Start the Report Explorer. • `setedit` — Edit a report template with the Report Explorer. • `rptconvert` — Convert a source file created by the report-generation process to the desired output format. • `rptlist` — List `.rpt` files in the current path.

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 on how to customize DocBook XML, see the OASIS DocBook TC Web page at `http://www.oasis-open.org`.

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

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

Note PDF reports only support bitmap (`.bmp`) and jpeg (`.jpg`) images.

Examples

Use this list to find examples in the documentation.

Generating Reports from MATLAB Files

Working with Components

Customizing Components

Customizing Stylesheets

Generating Reports from XML files

Creating MATLAB Reports

About This Tutorial

Tasks demonstrated in this tutorial include the following:

- Evaluating MATLAB expressions
- Formatting reports with multiple chapters
- Running loops and flow control
- Handling errors
- Inserting workspace data
- Inserting MATLAB images

This tutorial takes you through the steps required to design a report template file and generate a report from it in the MATLAB software.

The report template that you design does the following:

1. Uses MATLAB code to create several magic squares – matrices whose columns, rows, and diagonals all add up to the same number.

2. Displays the magic squares in the generated report as matrices or images, depending on their size.

For more information on magic squares, see “Matrices and Magic Squares” in the MATLAB documentation.

Note You do not need to know the MATLAB software to use this tutorial. However, knowledge of MATLAB is helpful for understanding the MATLAB code that executes during report generation.

Setting Report Options in the Report Template

To create and configure a report template:

1. Start a MATLAB software session.
2. Open the Report Explorer by entering `report` at the MATLAB command line.
3. Select **File > New** to create a report template. The new report template has the default name `Unnamed.rpt`.
4. In the Properties pane on the right:

   a. To save the report in the current working folder, select `Present Working Directory` from the **Directory** list.

   b. Set **File format** to `web (HTML)` to create the report as an HTML file.

   c. In the **Report description** text box, replace the existing text with the following text.

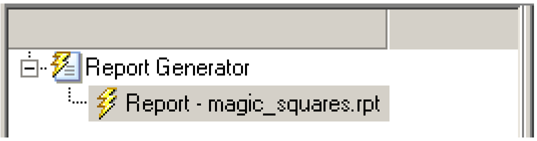
   Tip Copy and paste this text from the HTML documentation into the Report Explorer.

```
This report creates a series of magic squares
and displays them as images.

A magic square is a matrix in which the
columns, rows, and diagonal all add up to the
same number.
```

Note When you change a Properties pane field, its background color changes. This indicates that there are unapplied changes to that field. As soon as you perform any action with another component, MATLAB Report Generator applies the changes, and the background color becomes white again.

5 Save your report. Select **File > Save As** and name your report's report template `magic_squares.rpt`.

The new file name appears in the Outline pane on the left.



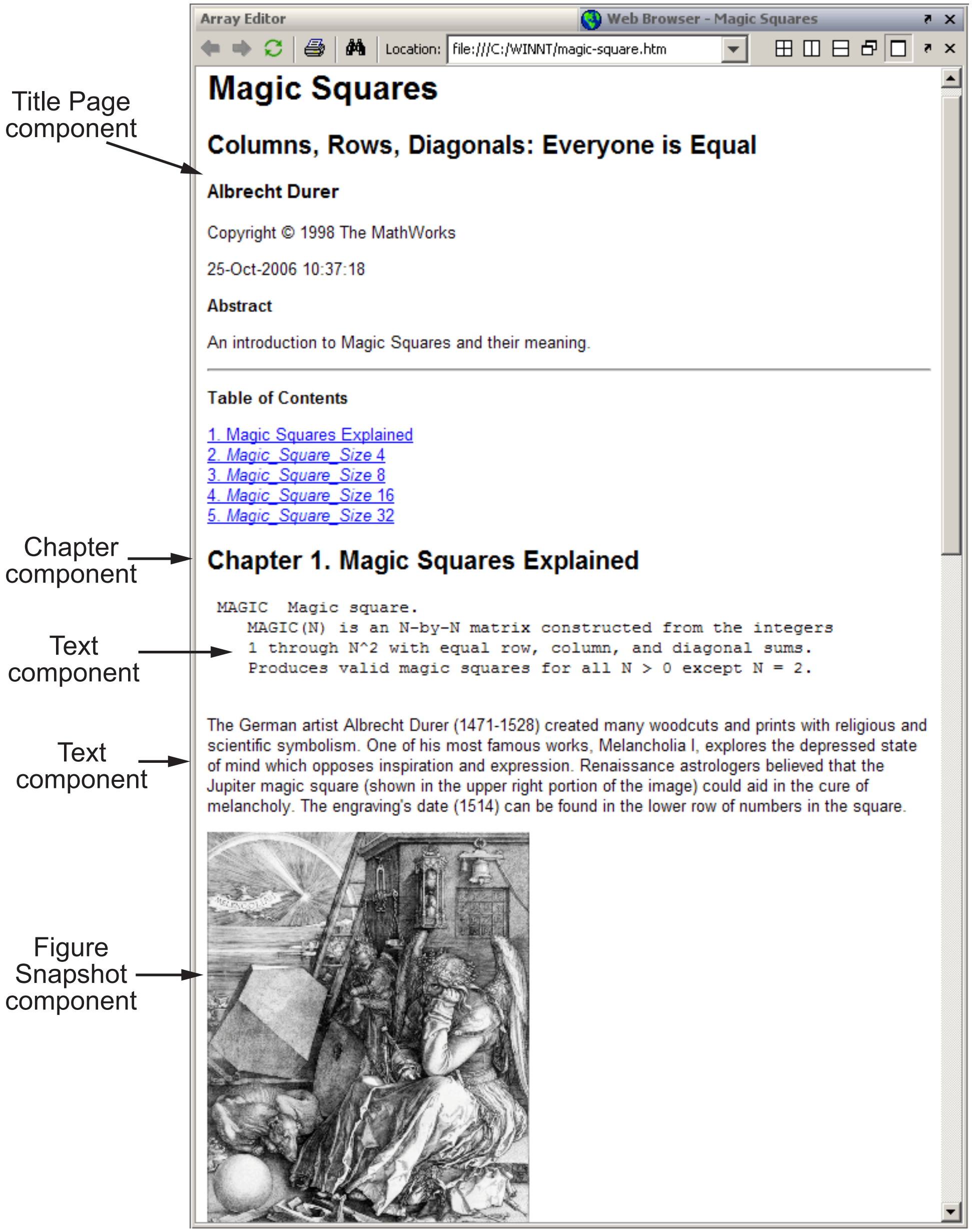
Adding Components to the Report Template

In this section...

About Components

Report components specify what information you want to include in the report. The following figure shows a sample page from the report you create in this tutorial and what components are used to produce it.

Note Report components added to the report template must not be deactivated in order for the report to generate correctly.



Array Editor
Web Browser - Magic Squares
Location: file:///C:/WINNT/magic-square.htm
Title Page component
Chapter component
Text component
Text component
Figure Snapshot component

Magic Squares

Columns, Rows, Diagonals: Everyone is Equal

Albrecht Durer

Copyright © 1998 The MathWorks

25-Oct-2006 10:37:18

Abstract

An introduction to Magic Squares and their meaning.

Table of Contents

Chapter 1. Magic Squares Explained

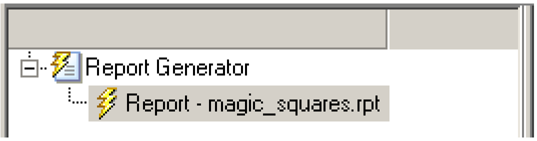
```
 MAGIC  Magic square.
    MAGIC(N) is an N-by-N matrix constructed from the integers
    1 through N^2 with equal row, column, and diagonal sums.
    Produces valid magic squares for all N > 0 except N = 2.
```

The German artist Albrecht Durer (1471-1528) created many woodcuts and prints with religious and scientific symbolism. One of his most famous works, Melancholia I, explores the depressed state of mind which opposes inspiration and expression. Renaissance astrologers believed that the Jupiter magic square (shown in the upper right portion of the image) could aid in the cure of melancholy. The engraving's date (1514) can be found in the lower row of numbers in the square.

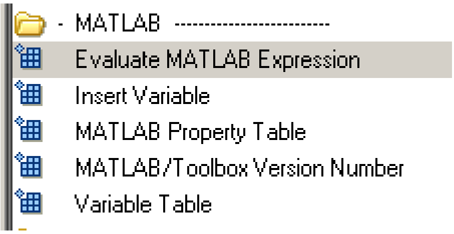
Setting Initial Values

To specify the sizes of the magic squares in the report, use the Evaluate MATLAB Expression component.

1 In the Outline pane on the left, select the top level of the report template.



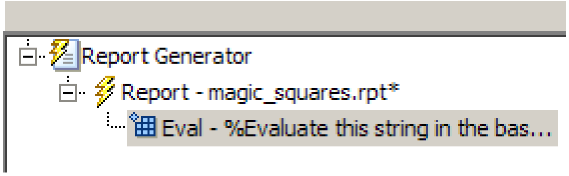
2 In the Options pane in the middle, under the `MATLAB` category, select `Evaluate MATLAB Expression`.


- MATLAB ------------------------
Evaluate MATLAB Expression
Insert Variable
MATLAB Property Table
MATLAB/Toolbox Version Number
Variable Table

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

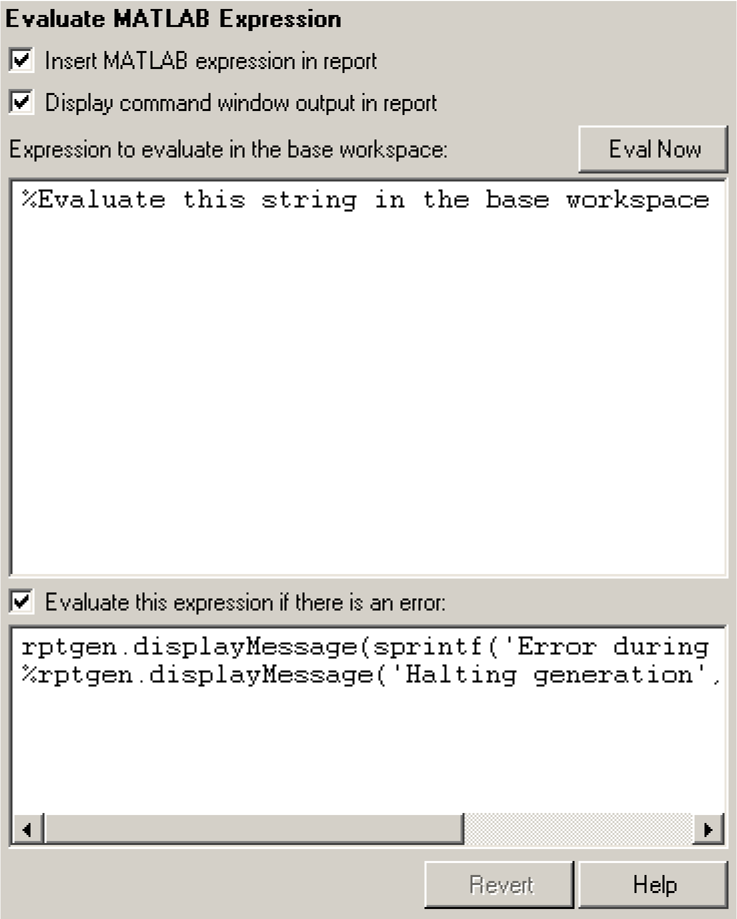
You cannot edit the component information in the Properties pane until you have added the component to the report.

In the Outline pane on the left, the `Eval` component appears under the `magic_squares` report.


Report Generator
Report - magic_squares.rpt*
Eval - %Evaluate this string in the bas...

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

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



Evaluate MATLAB Expression

☑ Insert MATLAB expression in report

☑ Display command window output in report

Expression to evaluate in the base workspace:

```
%Evaluate this string in the base workspace
```

☑ Evaluate this expression if there is an error:

```
rptgen.displayMessage(sprintf('Error during
%rptgen.displayMessage('Halting generation',
```

4 Clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes. You do not want to include the MATLAB code details or its output in this report.

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

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

```
%This MATLAB code sets up two variables
%that define how the report runs.
%magicSizeVector is a list of MxM
%Magic Square sizes to insert into
%the report. Note that magic
%squares cannot be 2x2.

magicSizeVector=[4 8 16 32];

%largestDisplayedArray sets the
%limit of array size that will be
%inserted into the report with the
%Insert Variable component.

largestDisplayedArray=15;
```

6 In the **Evaluate this expression if there is an error** text box, replace the existing text with the following text.

```
disp(['Error during eval: ', lasterr])
```

This causes an error to display if the MATLAB code fails.

Tip To execute these commands immediately, click the **Eval Now** button at the top right corner of the Report Explorer. This is an easy way to check if your commands are correct, and ensure that they will not cause report-generation problems.

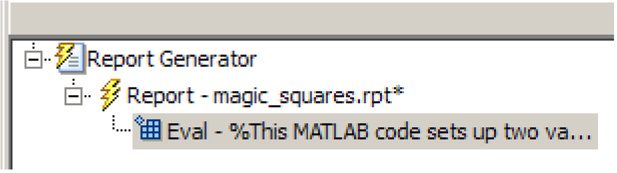
---

7 Save the report by clicking **File > Save**.

Creating a Title Page

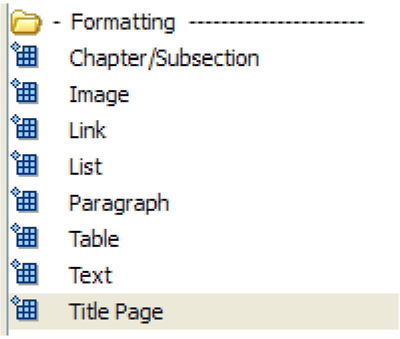
To create a title page for the report, use the Title Page component.

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



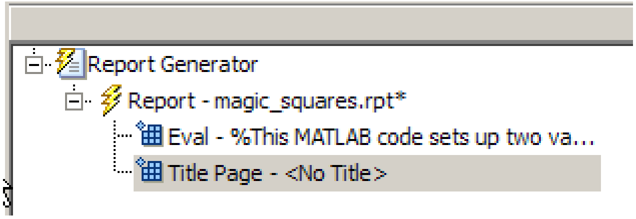
Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...

2 In the Options pane in the middle, under the `Formatting` category, double-click `Title Page` to add the component to the report.



- Formatting ---------------------
Chapter/Subsection
Image
Link
List
Paragraph
Table
Text
Title Page

Because the `Eval` component's icon indicates that this component cannot have children, the `Title Page` component is a sibling of the `Eval` component. Likewise, the `Title Page` component cannot have children either.



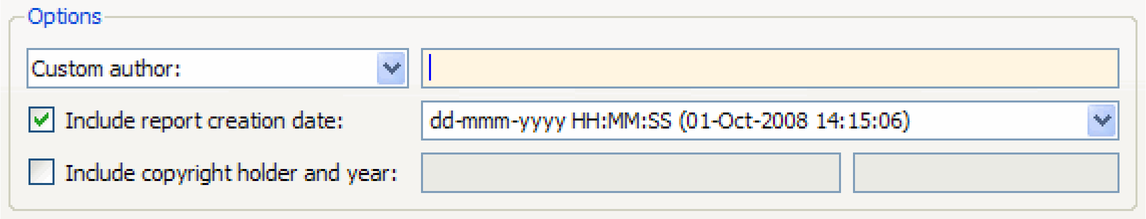
Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...
Title Page - <No Title>

3 In the Properties pane on the right,

a. In the **Title** text box, enter `Magic Squares`.

b. In the **Subtitle** text box, enter `Columns, Rows, Diagonals:  Everyone is Equal`.

c. Under **Options**, choose `Custom author` from the selection list.



Options

Custom author: |

☑ Include report creation date: dd-mmm-yyyy HH:MM:SS (01-Oct-2008 14:15:06)

□ Include copyright holder and year:

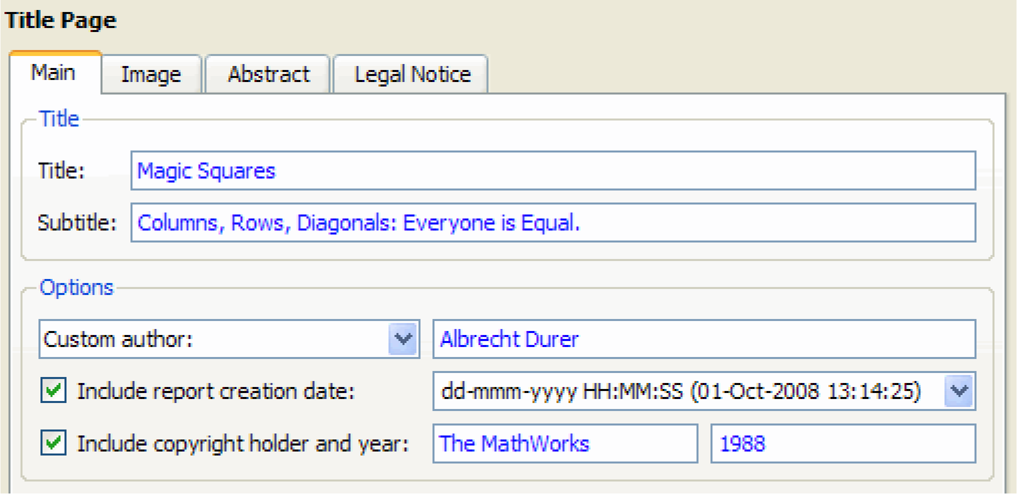
d. In the field to the right of the **Custom author** field, enter `Albrecht Durer`.

Albrecht Dürer created the etching that contains a magic square. Your final report includes an image of that etching.

e. Select the **Include copyright holder and year** check box.

f. In the next text box, enter `The MathWorks`.

g. In the second text box, enter `1998`.



Title Page

Title

Title: Magic Squares

Subtitle: Columns, Rows, Diagonals: Everyone is Equal.

Options

Custom author: Albrecht Durer

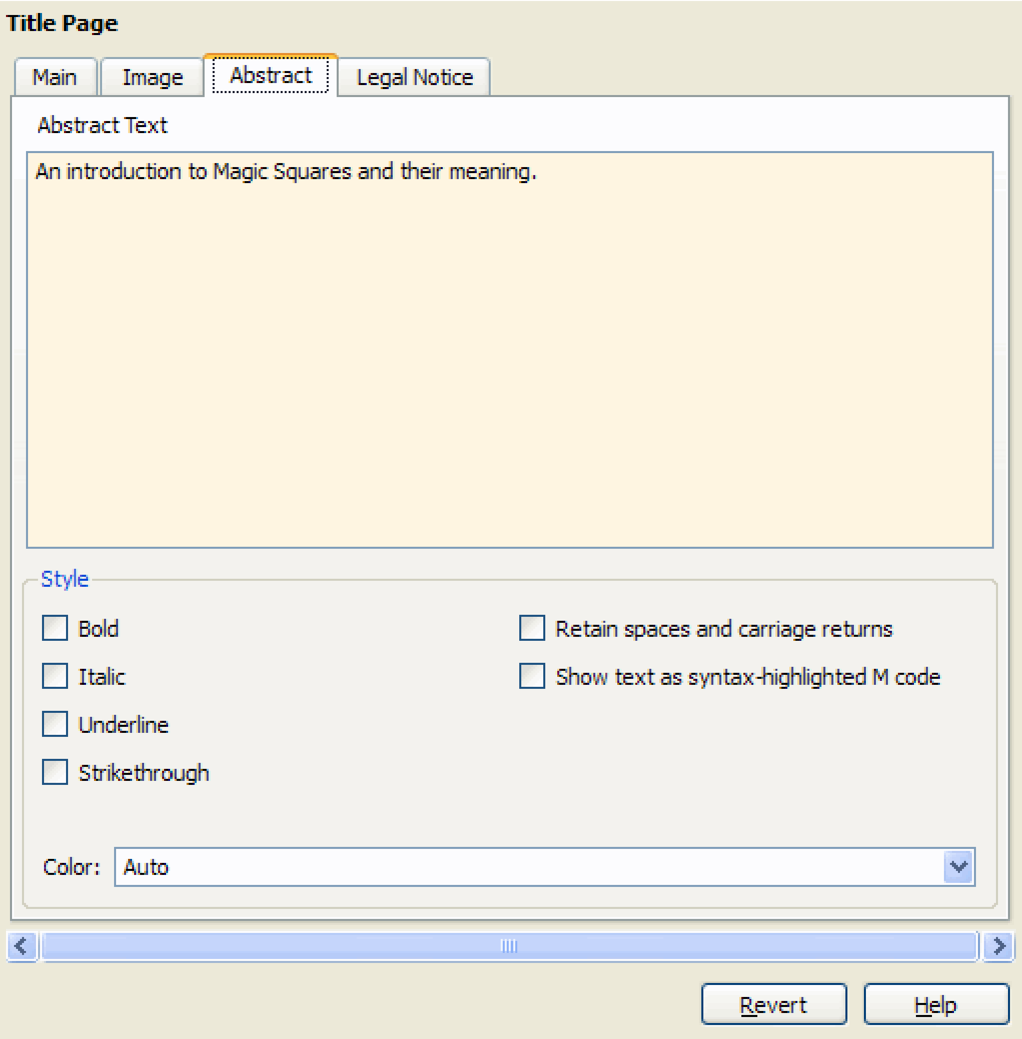
☑ Include report creation date: dd-mmm-yyyy HH:MM:SS (01-Oct-2008 13:14:25)

☑ Include copyright holder and year: The MathWorks 1988

h Click the **Abstract** tab and then enter the following text:

```
An introduction to Magic Squares and their meaning.
```

The pane should look as follows:



Abstract Text

An introduction to Magic Squares and their meaning.

Style

□ Bold

□ Italic

□ Underline

□ Strikethrough

□ Retain spaces and carriage returns

□ Show text as syntax-highlighted M code

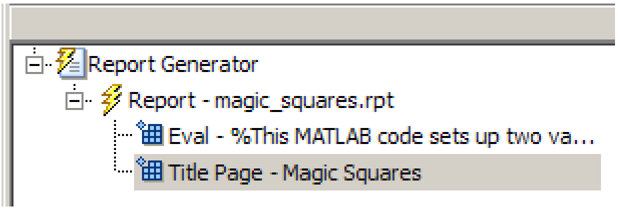
Color: Auto

4 Save the report.

Adding the First Chapter

Add a chapter to the report by using the Chapter/Subsection component.

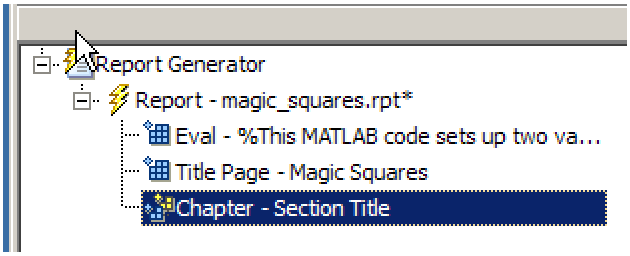
1 In the Outline pane on the left, select the `Title Page` component.



Report Generator
Report - magic_squares.rpt
Eval - %This MATLAB code sets up two va...
Title Page - Magic Squares

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

The Outline pane looks as follows.



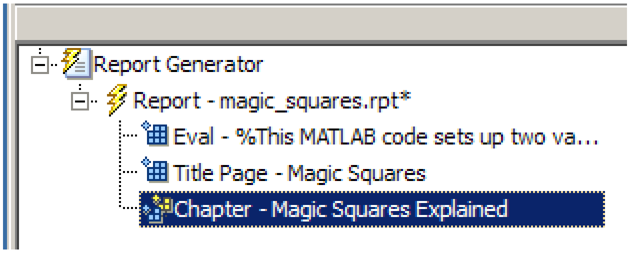
Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...
Title Page - Magic Squares
Chapter - Section Title

The `Eval`, `Title Page`, and `Chapter` components are all child components of the report's top level, but siblings of one another.

The `Chapter` component can have child components. The next section explains how to add child components to this `Chapter` component.

3 In the Properties pane on the right, enter `Magic Squares Explained` for the custom chapter title.

The Outline pane on the left changes to reflect the chapter title.


Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...
Title Page - Magic Squares
Chapter - Magic Squares Explained

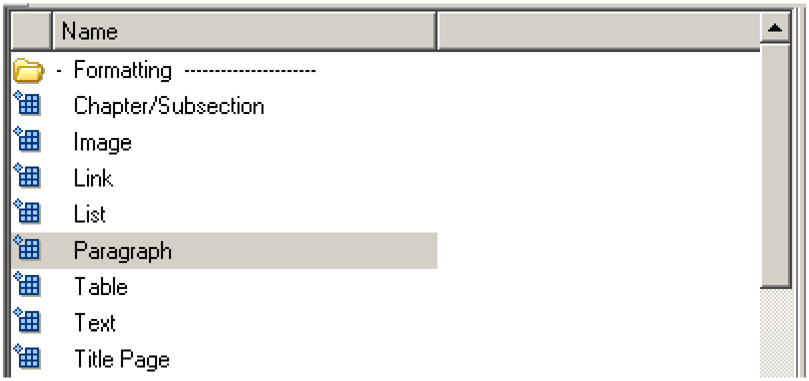
4 Save the report.

Adding Components to the First Chapter

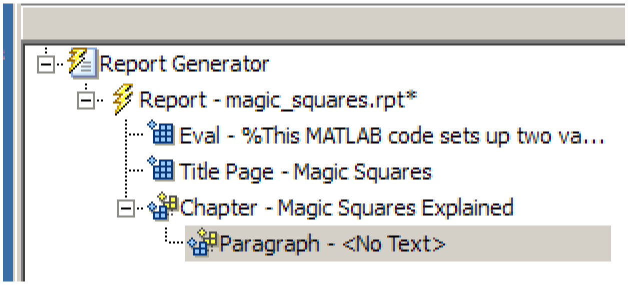
Add introductory text to the first chapter using the Paragraph and Text components.

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

2 In the Options pane in the middle, under the `Formatting` category, double-click `Paragraph`.



In the Outline pane on the left, the new component appears as a child of the `Chapter` component.



Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...
Title Page - Magic Squares
Chapter - Magic Squares Explained
Paragraph - <No Text>

3 By default, the Paragraph component inherits its text from its child components. Add two `Text` components.

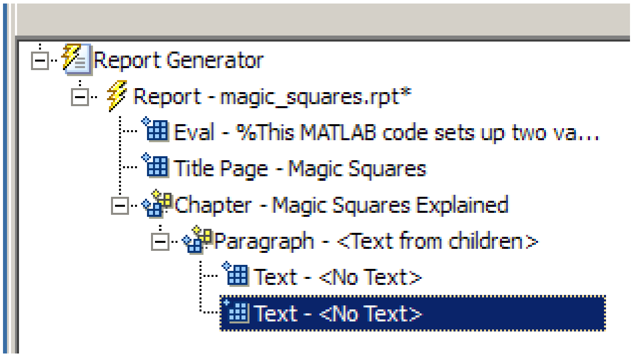
Note The `Text` component must have the `Paragraph` component as its parent.

---

4 In the Options pane in the middle, under the `Formatting` category, double-click `Text`.

5 Double-click `Text` again to add a second component.

The Outline pane looks as follows.



Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...
Title Page - Magic Squares
Chapter - Magic Squares Explained
Paragraph - <Text from children>
Text - <No Text>
Text - <No Text>

6 In the Outline pane on the left, select the first `Text` component.

7 In the **Text to include in report** text box, enter `%<help('magic')>`.

The % sign and angle brackets <> indicate to the MATLAB Report Generator software that this is MATLAB code to evaluate. The command `help('magic')` displays information about the MATLAB `magic` function.

8. In the Outline pane on the left, select the second `Text` component.

9. In the **Text to include in report** text box, enter the following text.

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

   ```
   The German artist Albrecht Durer (1471-1528)
   created many woodcuts and prints with religious
   and scientific symbolism.  One of his most famous
   works, Melancholia I, explores the depressed state
   of mind that opposes inspiration and expression.
   Renaissance astrologers believed that the Jupiter magic
   square (shown in the upper right portion of the image)
   could aid in the cure of melancholy. The engraving's
   date (1514) can be found in the lower row of numbers
   in the square.
   ```

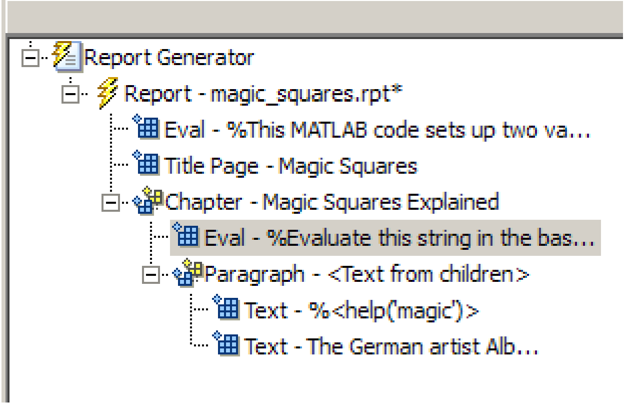
10. Save the report.

    The next step includes an image of the etching shown at the beginning of this section.

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

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

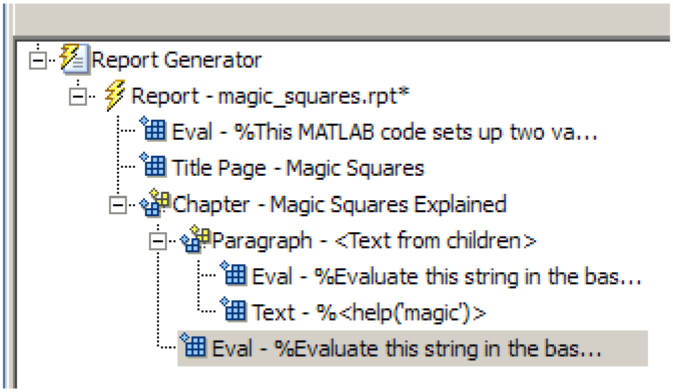
    The new component becomes a child of the `Chapter` component.



```
Report Generator
    Report - magic_squares.rpt*
        Eval - %This MATLAB code sets up two va...
        Title Page - Magic Squares
        Chapter - Magic Squares Explained
            Eval - %Evaluate this string in the bas...
            Paragraph - <Text from children>
                Text - %<help('magic')>
                Text - The German artist Alb...
```

Move the `Eval` component under the `Paragraph` component so that the image follows the introductory text.

13 Click the **down** arrow on the toolbar to move the `Eval` component under the `Paragraph` component.



```
Report Generator
    Report - magic_squares.rpt*
        Eval - %This MATLAB code sets up two va...
        Title Page - Magic Squares
        Chapter - Magic Squares Explained
            Paragraph - <Text from children>
                Eval - %Evaluate this string in the bas...
                Text - %<help('magic')>
            Eval - %Evaluate this string in the bas...
```

14 With the `Eval` component still selected, do the following in the Properties pane on the right:

a Clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes. You do not want to include the code or its output in the report.

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

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

```
%This loads a self-portrait of Albrecht
%Durer, a German artist. There is a
%magic square in the upper right corner
%of the image.

durerData=load('durer.mat','-mat');
figure('Units','Pixels',...
'Position',[200 200 size(durerData.X,2)*.5 size(durerData.X,1)*.5 ]);

image(durerData.X);
colormap(durerData.map);
axis('image');
set(gca,...
    'Xtick',[],...
    'Ytick',[],...
    'Units','normal',...
    'Position',[0 0 1 1]);

clear durerData
```

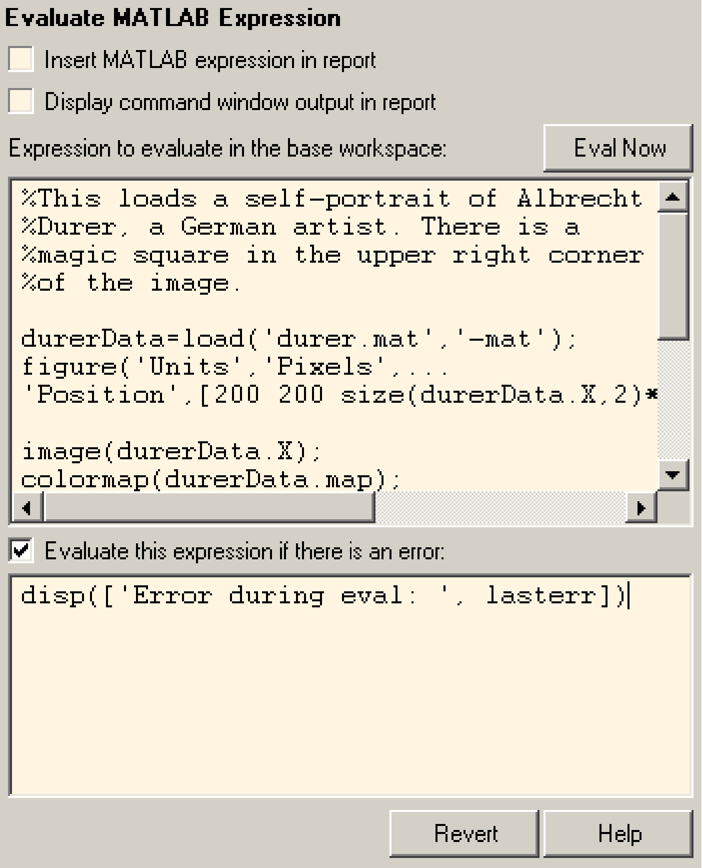
This MATLAB code displays the Dürer etching in a MATLAB figure window.

c In the **Evaluate expression if there is an error** text box, replace the existing text with the following text:

```
disp(['Error during eval: ', lasterr])
```

This code executes if an error occurs while loading the Dürer etching.

The Properties pane on the right looks as follows.



Evaluate MATLAB Expression

□ Insert MATLAB expression in report

□ Display command window output in report

Expression to evaluate in the base workspace:

```
%This loads a self-portrait of Albrecht
%Durer, a German artist. There is a
%magic square in the upper right corner
%of the image.

durerData=load('durer.mat','-mat');
figure('Units','Pixels',...
'Position',[200 200 size(durerData.X,2)*

image(durerData.X);
colormap(durerData.map);
```

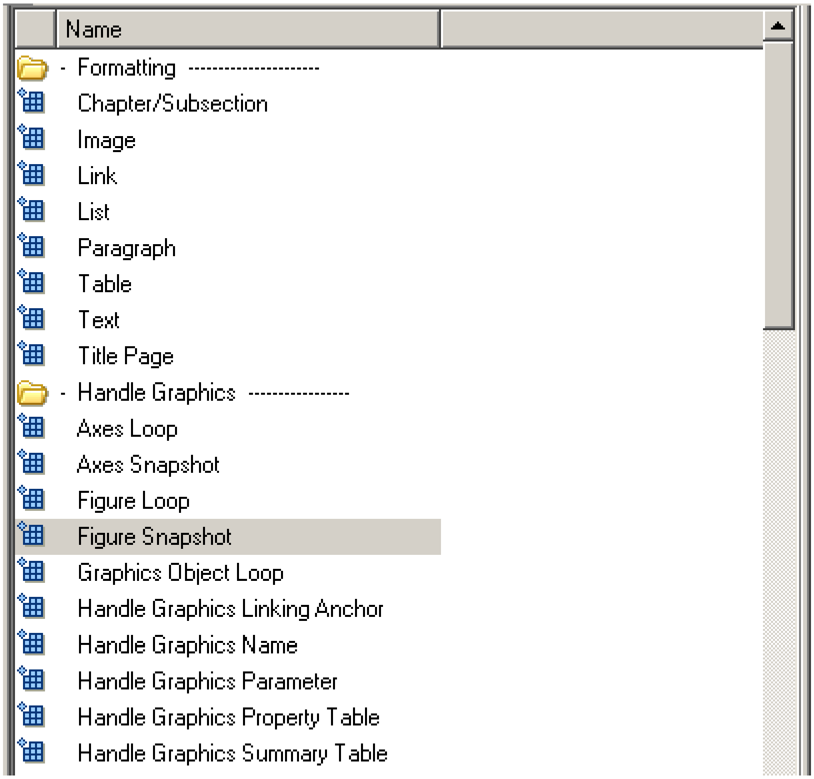
☑ Evaluate this expression if there is an error:

```
disp(['Error during eval: ', lasterr])
```

Now that the image of the Dürer etching appears in the MATLAB workspace, include it in the report using the Figure Snapshot component.

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

16 In the Options pane in the middle, under the `Handle Graphics` category, double-click `Figure Snapshot`.



Name

- Formatting --------------------
  - Chapter/Subsection
  - Image
  - Link
  - List
  - Paragraph
  - Table
  - Text
  - Title Page
- Handle Graphics ----------------
  - Axes Loop
  - Axes Snapshot
  - Figure Loop
  - Figure Snapshot
  - Graphics Object Loop
  - Handle Graphics Linking Anchor
  - Handle Graphics Name
  - Handle Graphics Parameter
  - Handle Graphics Property Table
  - Handle Graphics Summary Table

17 In the Properties pane on the right:

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

b In the **Invert hardcopy** list, select `Don't invert`.

Selecting this option specifies not to change the image's on-screen colors for printing.

Delete the image after adding it to the report.

18 In the Outline pane on the left, select the `Figure Snapshot` component.

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

20 In the Properties pane on the right:

a Clear the **Insert MATLAB expression in report** and **Display command window output in report** check boxes. You do not want to include the code or its output in the report.

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

```
%This command deletes the Durer image
delete(gcf);
```

The `delete(gcf)` command deletes the current image in the MATLAB workspace, in this case, the Dürer etching.

c In the **Evaluate expression if there is an error** text box, replace the existing text with the following text:

```
disp(['Error during eval: ', lasterr])
```

This code executes if an error occurs while deleting the Dürer etching.

21 Save the report.

The contents of the first chapter are now complete.

Creating the Magic Squares and Their Images

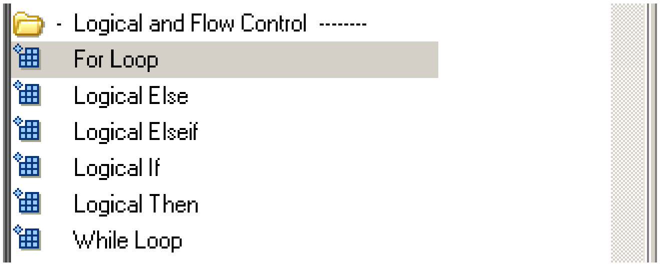
Use components to create several magic squares and insert the contents of the squares, or representative images, into the report.

Creating the For Loop

Each square has its own chapter. A For Loop component performs tasks for each square.

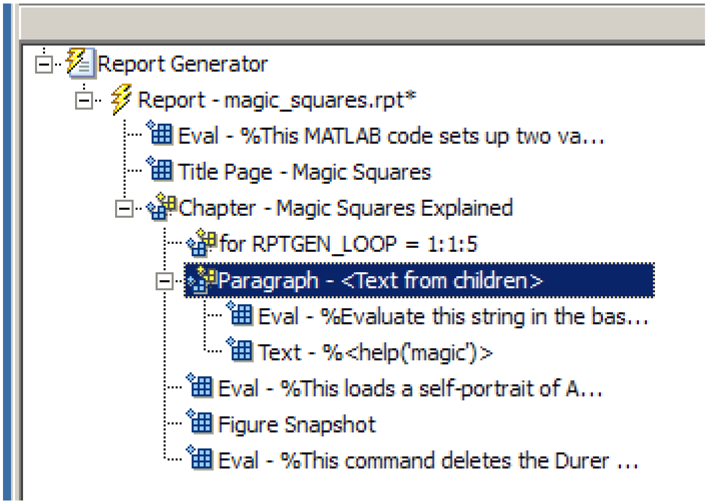
1 In the Outline pane on the left, select the `Chapter` component.

2 In the Options pane in the middle, under the `Logical and Flow Control` category, double-click `For Loop`.



- Logical and Flow Control --------
For Loop
Logical Else
Logical Elseif
Logical If
Logical Then
While Loop

The Outline pane on the left looks as follows.

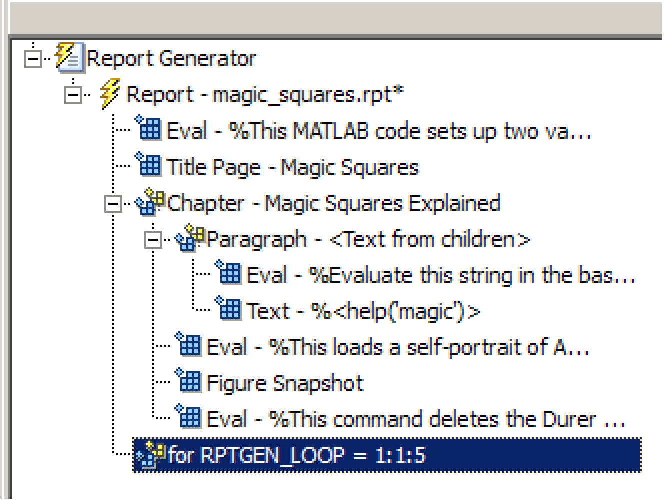


Report Generator
Report - magic_squares.rpt*
Eval - %This MATLAB code sets up two va...
Title Page - Magic Squares
Chapter - Magic Squares Explained
for RPTGEN_LOOP = 1:1:5
Paragraph - <Text from children>
Eval - %Evaluate this string in the bas...
Text - %<help('magic')>
Eval - %This loads a self-portrait of A...
Figure Snapshot
Eval - %This command deletes the Durer ...

This component appears inside the `Chapter` component. However, the magic squares should be processed *after* the first chapter, so the `for` component should be a sibling of the `Chapter` component, not a child.

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

4 Click the **left** arrow to make the `for` component a sibling, not a child, of the `Chapter` component.



```
Report Generator
  Report - magic_squares.rpt*
    Eval - %This MATLAB code sets up two va...
    Title Page - Magic Squares
    Chapter - Magic Squares Explained
      Paragraph - <Text from children>
        Eval - %Evaluate this string in the bas...
        Text - %<help('magic')>
      Eval - %This loads a self-portrait of A...
      Figure Snapshot
      Eval - %This command deletes the Durer ...
    for RPTGEN_LOOP = 1:1:5
```

5 In the Properties pane on the right:

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

```
length(magicSizeVector)
```

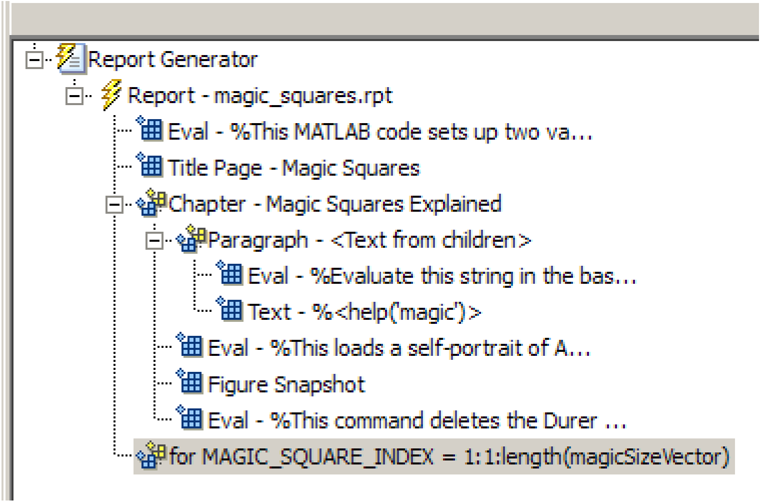
This is the length of the vector that contains the various sizes for the magic square matrices.

b In the **Variable name** text box, replace the existing text with the following text:

```
MAGIC_SQUARE_INDEX
```

This variable acts as a loop index.

The Outline pane on the left looks as follows.



```
Report Generator
    Report - magic_squares.rpt
        Eval - %This MATLAB code sets up two va...
        Title Page - Magic Squares
        Chapter - Magic Squares Explained
            Paragraph - <Text from children>
                Eval - %Evaluate this string in the bas...
                Text - %<help('magic')>
            Eval - %This loads a self-portrait of A...
            Figure Snapshot
            Eval - %This command deletes the Durer ...
        for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
```

6 Save the report.

Adding a Chapter for Each Square

Add a chapter for each square processed using the Chapter/Subsection component.

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

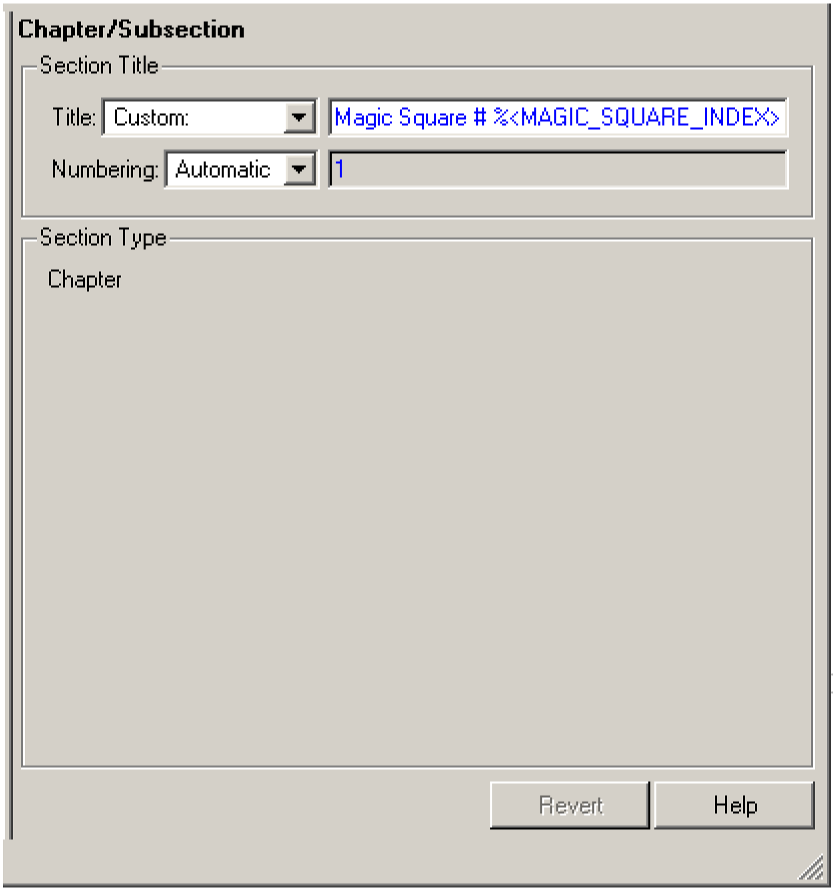
2 In the Options pane in the middle, under the `Formatting` category, double-click `Chapter/Subsection`.

   It becomes a child of the `for` component.

3 In the Properties pane on the right, select `Custom` from the **Title** list and enter the following for the chapter title:

   ```
   Magic Square # %<MAGIC_SQUARE_INDEX>
   ```

   The Properties pane looks as follows.



Chapter/Subsection

Section Title

Title: Custom: Magic Square # %<MAGIC_SQUARE_INDEX>

Numbering: Automatic 1

Section Type

Chapter

4 Save the report.

Determining the Matrix Size

Extract the size of each magic square matrix from `magicSizeVector` using an Evaluate MATLAB Expression component.

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

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

3 In the Properties pane on the right:

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, replace the existing text with the following text:

```
magic_Square_Size=magicSizeVector(MAGIC_SQUARE_INDEX);
```

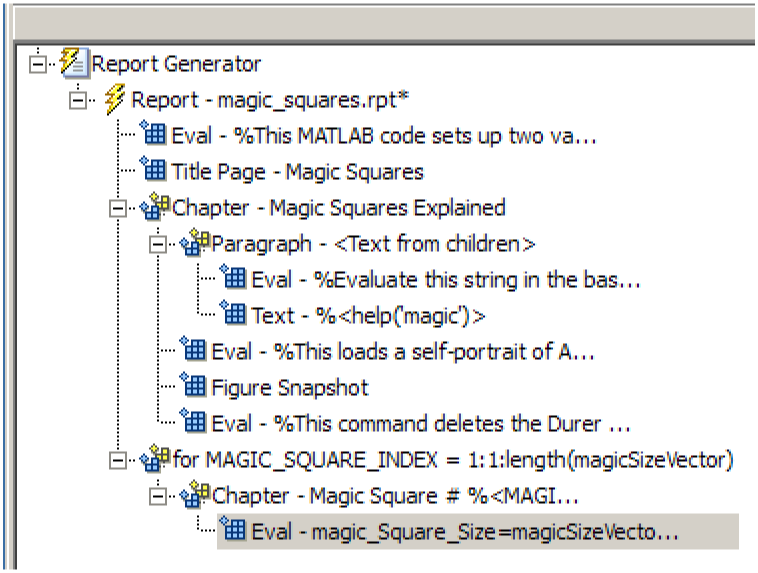
This command extracts the next size for the magic square from the vector of sizes initialized in the first `Eval` component of the report. The variable `magic_Square_Size` represents the size of the current magic square being processed.

c In the **Evaluate expression if there is an error** text box, replace the existing text with the following:

```
disp(['Error during eval: ', lasterr])
```

This code executes if an error occurs while attempting to extract a value from `magicSizeVector`.

The Outline pane on the left looks as follows.



```
Report Generator
  Report - magic_squares.rpt*
    Eval - %This MATLAB code sets up two va...
    Title Page - Magic Squares
    Chapter - Magic Squares Explained
      Paragraph - <Text from children>
        Eval - %Evaluate this string in the bas...
        Text - %<help('magic')>
      Eval - %This loads a self-portrait of A...
      Figure Snapshot
      Eval - %This command deletes the Durer ...
    for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
      Chapter - Magic Square # %<MAGI...
        Eval - magic_Square_Size=magicSizeVecto...
```

4 Save the report.

Inserting the Magic Square Size into the Report

Insert the size of the magic square into the report using the Paragraph and Insert Variable components.

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

2 In the Options pane in the middle, under the `Formatting` category, double-click `Paragraph`.

Do not change the properties. The variable that contains the size of the magic square goes in this paragraph.

3 In the Outline pane on the left, select the `Paragraph` component (below the `for` component).

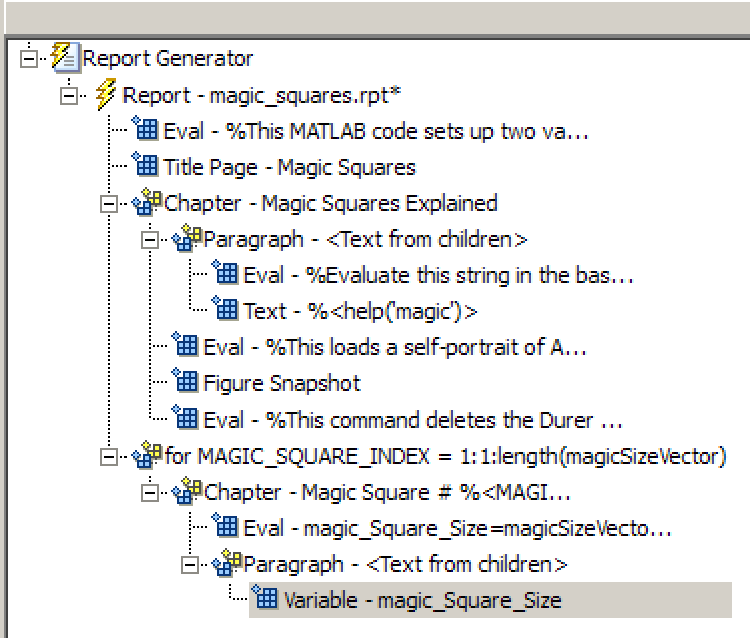
4 In the Options pane in the middle, under the `MATLAB` category, double-click `Insert Variable`.

5 In the Properties pane on the right:

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

b In the **Display as** list, select `Inline text`.

The Outline pane on the left looks as follows.



```
Report Generator
  Report - magic_squares.rpt*
    Eval - %This MATLAB code sets up two va...
    Title Page - Magic Squares
    Chapter - Magic Squares Explained
      Paragraph - <Text from children>
        Eval - %Evaluate this string in the bas...
        Text - %<help('magic')>
      Eval - %This loads a self-portrait of A...
      Figure Snapshot
      Eval - %This command deletes the Durer ...
    for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
      Chapter - Magic Square # %<MAGI...
        Eval - magic_Square_Size=magicSizeVecto...
        Paragraph - <Text from children>
          Variable - magic_Square_Size
```

6 Save the report.

Displaying the Magic Square

To create the magic square and display the associated matrix or image, use the Evaluate MATLAB Expression component.

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

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

Make this component a sibling of the `Paragraph` component, not a child, as described in the next two steps.

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

4 Click the left arrow on the toolbar to make the `Eval` component a sibling of the previous `Paragraph` component.

5 In the Properties pane on the right:

   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, replace the existing text with the following MATLAB code.

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

```
%This MATLAB script produces a magic
%square of size magic_Square_Size
%and creates an image of that square.

mySquare=magic(magic_Square_Size);
clf
imagesc(mySquare);
title(sprintf('Magic Square N=%i',magic_Square_Size))
set(gca,'Ydir','normal');
axis equal;
axis tight;
```

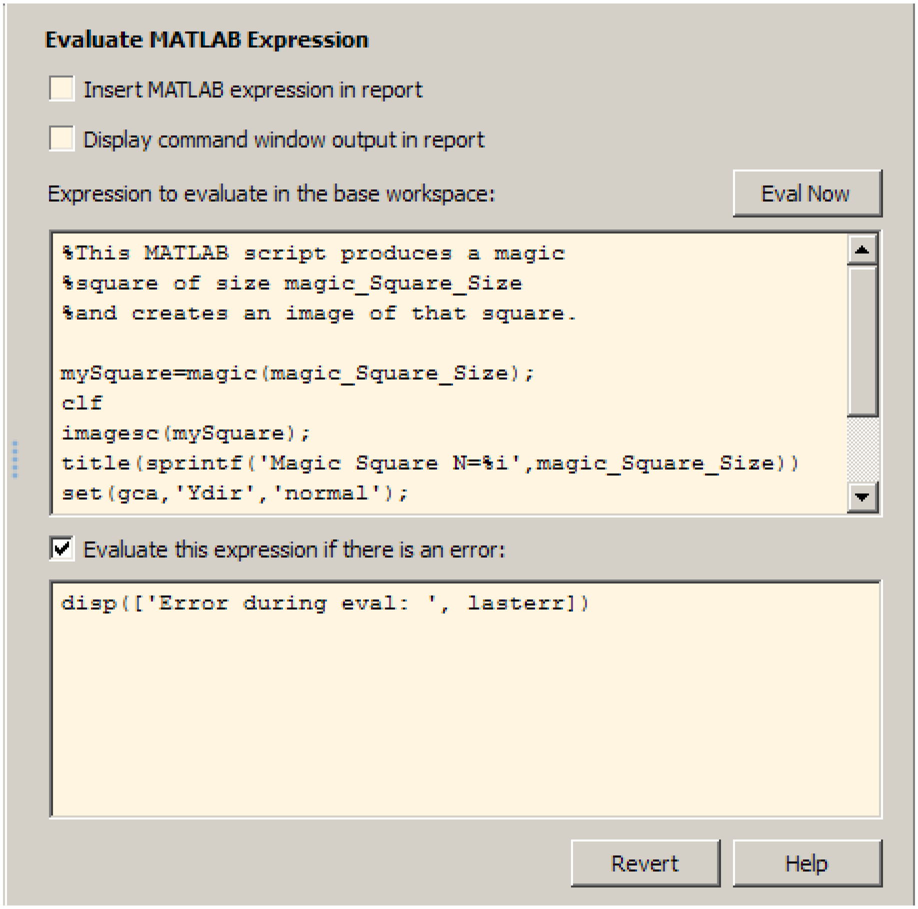
   This code creates a magic square matrix `mySquare` of size `magic_Square_Size`, and opens an image of that matrix in the MATLAB figure window.

   c In the **Evaluate expression if there is an error** text box, replace the existing text with the following:

```
disp(['Error during eval: ', lasterr])
```

   This code executes if an error occurs while creating and displaying the magic square.

The Properties pane on the right looks as follows.



Evaluate MATLAB Expression

□ Insert MATLAB expression in report

□ Display command window output in report

Expression to evaluate in the base workspace:

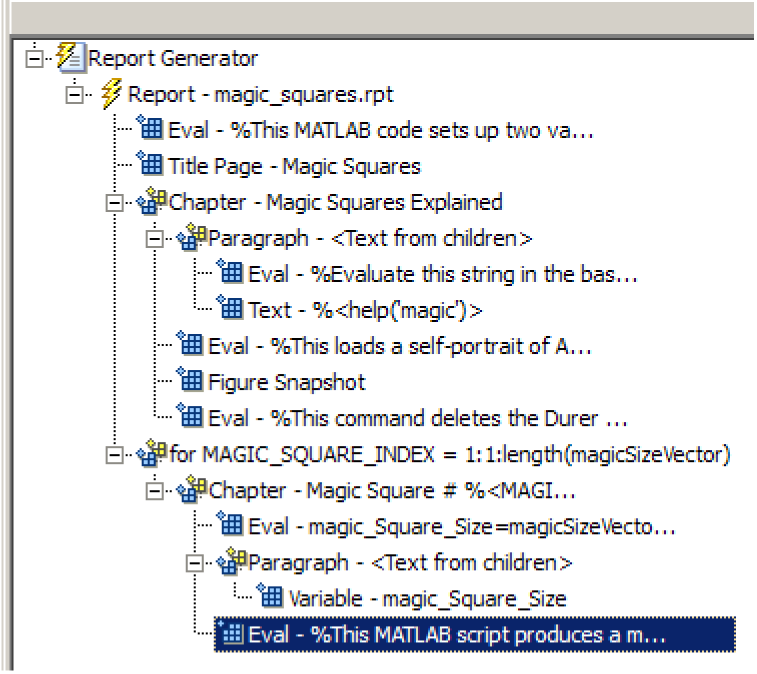
```
%This MATLAB script produces a magic
%square of size magic_Square_Size
%and creates an image of that square.

mySquare=magic(magic_Square_Size);
clf
imagesc(mySquare);
title(sprintf('Magic Square N=%i',magic_Square_Size))
set(gca,'Ydir','normal');
```

☑ Evaluate this expression if there is an error:

```
disp(['Error during eval: ', lasterr])
```

6 Save the report.

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



```
Report Generator
  Report - magic_squares.rpt
    Eval - %This MATLAB code sets up two va...
    Title Page - Magic Squares
    Chapter - Magic Squares Explained
      Paragraph - <Text from children>
        Eval - %Evaluate this string in the bas...
        Text - %<help('magic')>
      Eval - %This loads a self-portrait of A...
      Figure Snapshot
      Eval - %This command deletes the Durer ...
    for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
      Chapter - Magic Square # %<MAGI...
        Eval - magic_Square_Size=magicSizeVecto...
        Paragraph - <Text from children>
          Variable - magic_Square_Size
        Eval - %This MATLAB script produces a m...
```

8 On the Options pane in the middle, under the `Logical and Flow Control` category, double-click `Logical If`.

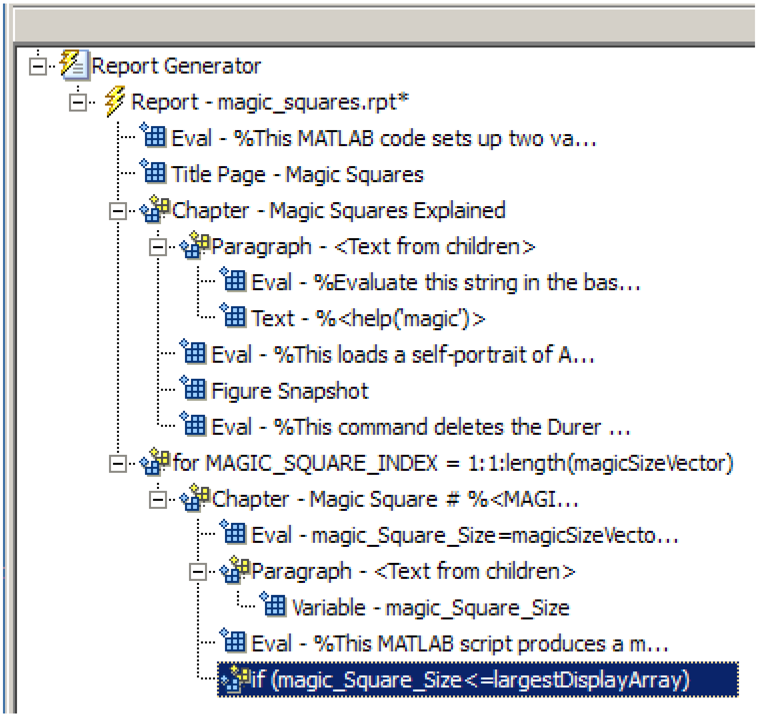
9 On the Properties pane on the right, in the **Test Expression** text box, replace the existing text with the following text:

```
magic_Square_Size<=largestDisplayedArray
```

This command tests if the current matrix size (`magic_Square_Size`) is less than or equal to the value assigned in the first `Eval` component of the report (`largestDisplayedArray=15`).

To process the result of this `Logical If` component, create two child components—`Logical Then` and `Logical Else`. If `magic_Square_Size` is less than or equal to 15, the matrix variable appears in the report. If `magic_Square_Size` is greater than 15, the matrix image appears in the report.

10 On the Outline pane on the left, select the `if` component.



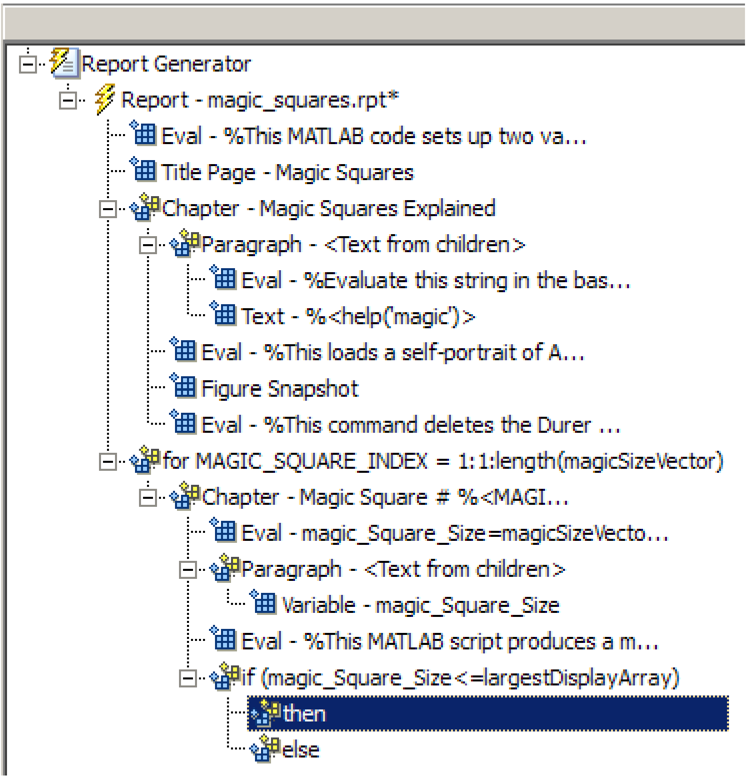
- Report Generator
  - Report - magic_squares.rpt*
    - Eval - %This MATLAB code sets up two va...
    - Title Page - Magic Squares
    - Chapter - Magic Squares Explained
      - Paragraph - <Text from children>
        - Eval - %Evaluate this string in the bas...
        - Text - %<help('magic')>
      - Eval - %This loads a self-portrait of A...
      - Figure Snapshot
      - Eval - %This command deletes the Durer ...
    - for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
      - Chapter - Magic Square # %<MAGI...
        - Eval - magic_Square_Size=magicSizeVecto...
        - Paragraph - <Text from children>
          - Variable - magic_Square_Size
        - Eval - %This MATLAB script produces a m...
        - if (magic_Square_Size<=largestDisplayArray)

11 On the Options pane in the middle, under `Logical and Flow Control`, double-click `Logical Else`.

12 On the Outline pane on the left, select the `if` component again.

13 On the Options pane in the middle, under `Logical and Flow Control`, double-click `Logical Then`.

The `then` component appears above the `else` component.



- Report Generator
  - Report - magic_squares.rpt*
    - Eval - %This MATLAB code sets up two va...
    - Title Page - Magic Squares
    - Chapter - Magic Squares Explained
      - Paragraph - <Text from children>
        - Eval - %Evaluate this string in the bas...
        - Text - %<help('magic')>
      - Eval - %This loads a self-portrait of A...
      - Figure Snapshot
      - Eval - %This command deletes the Durer ...
    - for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
      - Chapter - Magic Square # %<MAGI...
        - Eval - magic_Square_Size=magicSizeVecto...
        - Paragraph - <Text from children>
          - Variable - magic_Square_Size
        - Eval - %This MATLAB script produces a m...
        - if (magic_Square_Size<=largestDisplayArray)
          - then
          - else

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

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

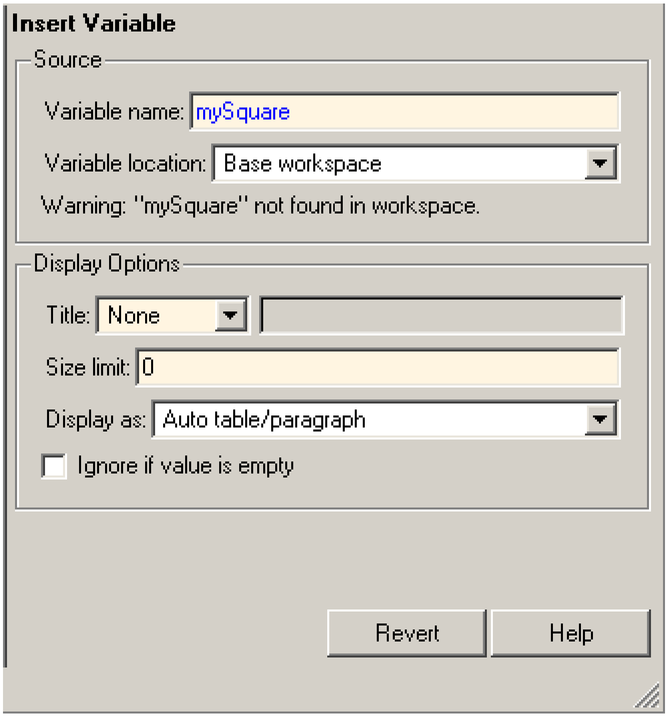
16 In the Properties pane on the right:

a In the **Variable name** text box, enter `mySquare`, which is the variable that contains the magic square of the specified size.

b In the **Title** list, select `None`.

c In the **Size Limit** text box, enter `0`.

The Properties pane on the right looks as follows.



Insert Variable

Source

Variable name: mySquare

Variable location: Base workspace

Warning: "mySquare" not found in workspace.

Display Options

Title: None

Size limit: 0

Display as: Auto table/paragraph

□ Ignore if value is empty

This `Variable` component displays the magic square matrix, stored in the variable `mySquare`.

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

18 In the Options pane in the middle, under the `Handle Graphics` category, double-click `Figure Loop`.

Do not change its properties.

19 In the Outline pane on the left, select the `Figure Loop` component.

20 In the Options pane in the middle, under the `Handle Graphics` category, double-click `Figure Snapshot`.

21 In the Properties pane on the right:

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

b In the **Image size** list, select `Custom`.

c Under the **Image size** list, enter `[5 4]` for the custom image size.

d In the **Invert hardcopy** list, select `Invert`.

This option changes dark axes colors to light axes colors, and vice versa.

The Properties pane on the right looks as follows.

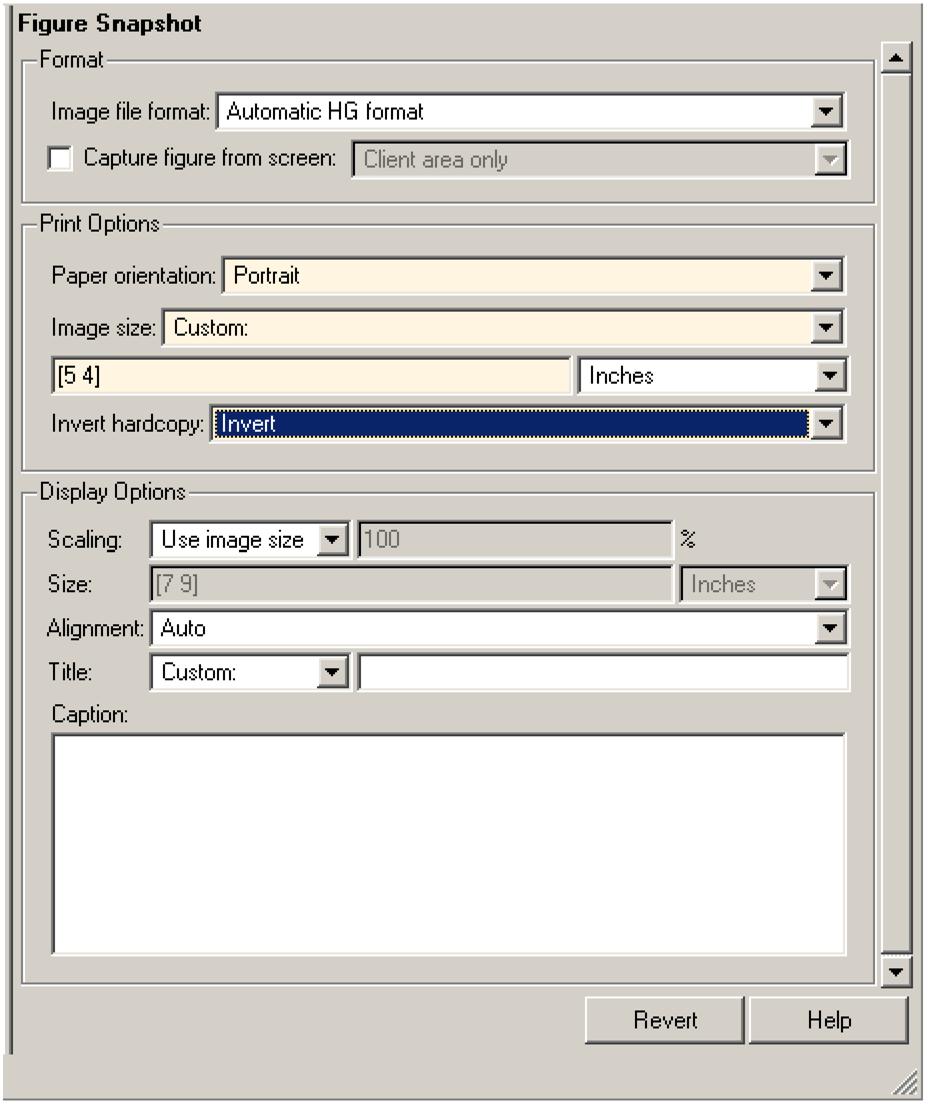


Figure Snapshot

Format

Image file format: Automatic HG format

□ Capture figure from screen: Client area only

Print Options

Paper orientation: Portrait

Image size: Custom:

[5 4] Inches

Invert hardcopy: Invert

Display Options

Scaling: Use image size 100 %

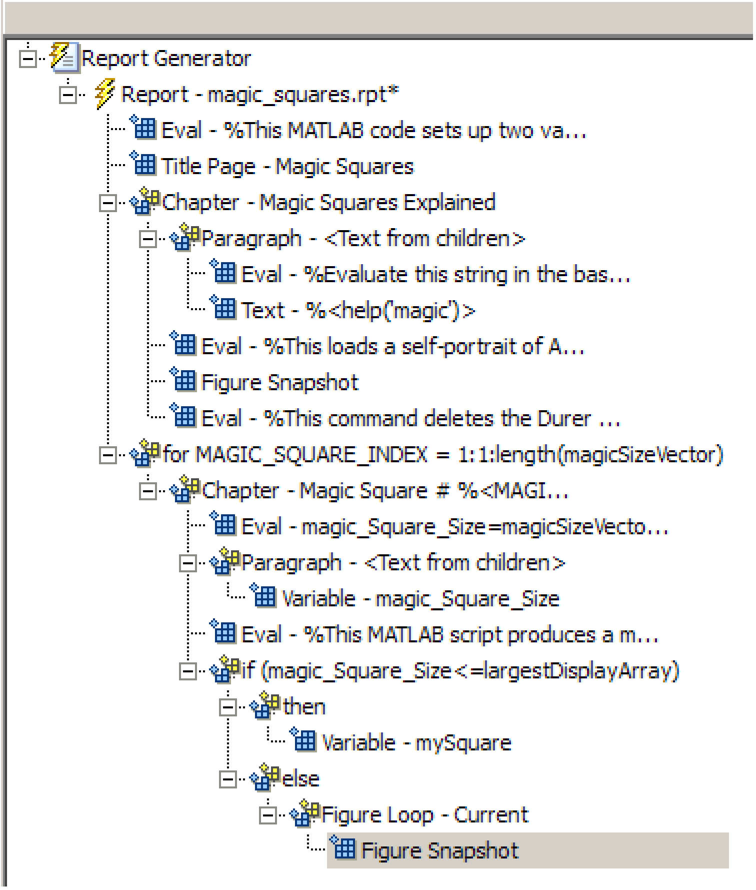
Size: [7 9] Inches

Alignment: Auto

Title: Custom:

Caption:

The Outline pane on the left looks like the following.



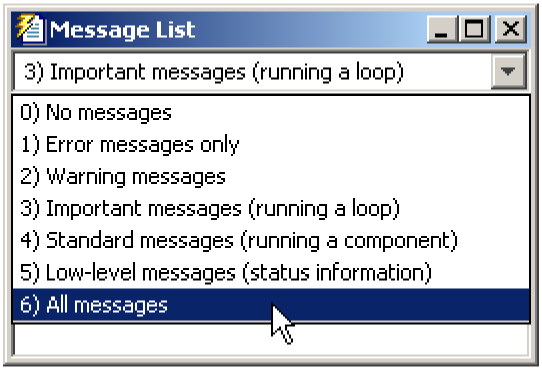
- Report Generator
  - Report - magic_squares.rpt*
    - Eval - %This MATLAB code sets up two va...
    - Title Page - Magic Squares
    - Chapter - Magic Squares Explained
      - Paragraph - <Text from children>
        - Eval - %Evaluate this string in the bas...
        - Text - %<help('magic')>
      - Eval - %This loads a self-portrait of A...
      - Figure Snapshot
      - Eval - %This command deletes the Durer ...
    - for MAGIC_SQUARE_INDEX = 1:1:length(magicSizeVector)
      - Chapter - Magic Square # %<MAGI...
        - Eval - magic_Square_Size=magicSizeVecto...
        - Paragraph - <Text from children>
          - Variable - magic_Square_Size
        - Eval - %This MATLAB script produces a m...
        - if (magic_Square_Size<=largestDisplayArray)
          - then
            - Variable - mySquare
          - else
            - Figure Loop - Current
              - Figure Snapshot

Generating the Report

Now the report contains all the components it needs.

Click the **Report** icon on the toolbar to generate the report. The following displays on your screen.

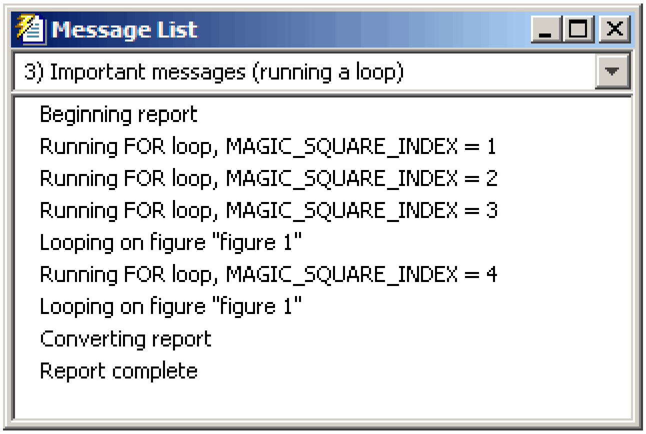
1 A Message List window appears, displaying informational and error messages as the report processes. While the report generates, specify the level of detail you would like the Message List window to display. Options range from 0 (least detail) to 6 (most detail). Click the list located under the title bar of the Message List window to choose an option, as shown here.



3) Important messages (running a loop)

Message List
0) No messages
1) Error messages only
2) Warning messages
3) Important messages (running a loop)
4) Standard messages (running a component)
5) Low-level messages (status information)
6) All messages

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



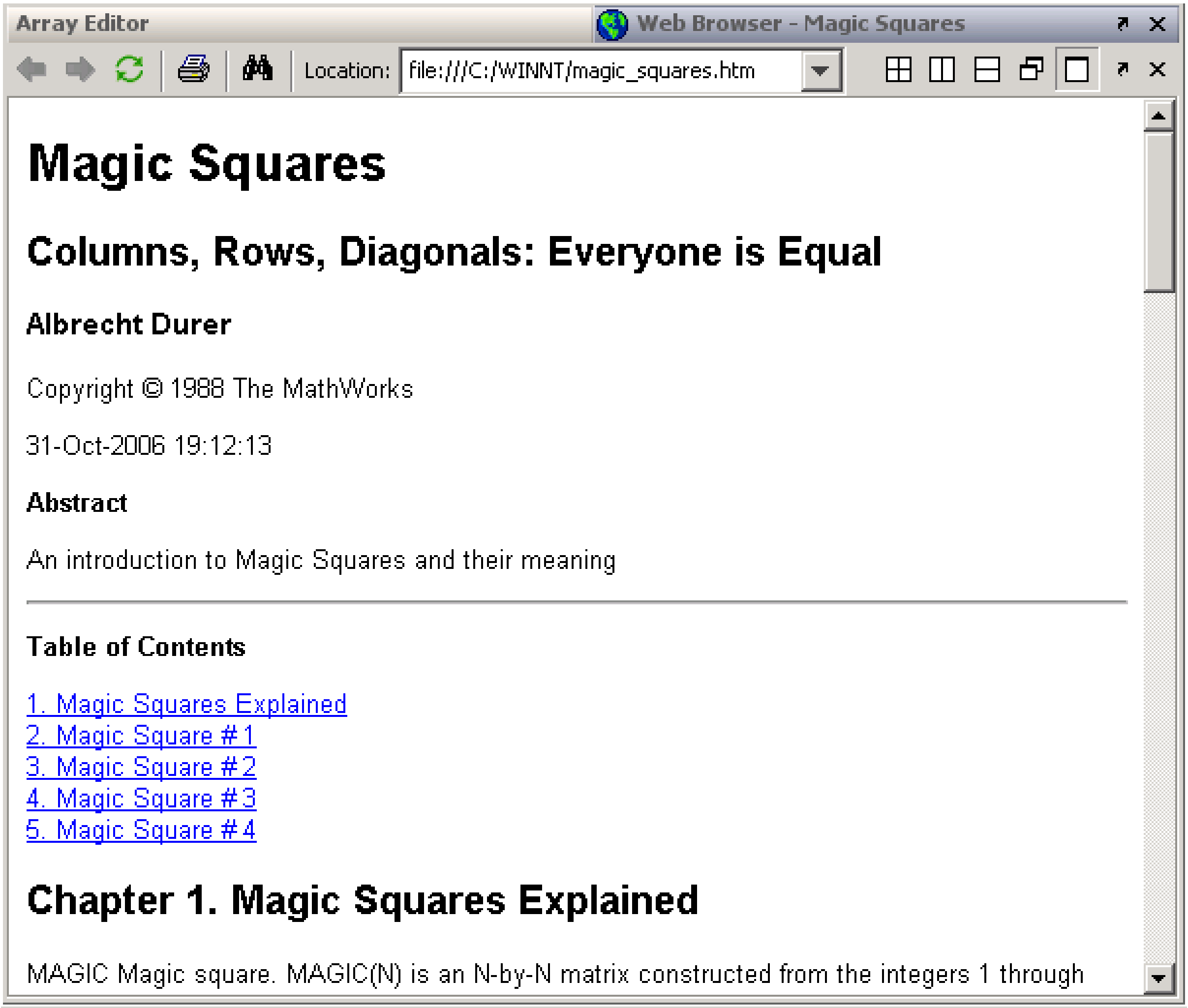
```
Beginning report
Running FOR loop, MAGIC_SQUARE_INDEX = 1
Running FOR loop, MAGIC_SQUARE_INDEX = 2
Running FOR loop, MAGIC_SQUARE_INDEX = 3
Looping on figure "figure 1"
Running FOR loop, MAGIC_SQUARE_INDEX = 4
Looping on figure "figure 1"
Converting report
Report complete
```

2 An image of the etching appears briefly.

3 Images of two magic square images of sizes 16 and 32 appear briefly.

4 In the Outline pane on the left of your Report Explorer window, each component of the report template is highlighted as it is executed.

At the beginning of this tutorial you specified HTML as the output format of this report. When processing finishes, the MATLAB Web browser opens and displays the report's HTML file.



Array Editor
Web Browser - Magic Squares
Location: file:///C:/WINNT/magic_squares.htm

Magic Squares

Columns, Rows, Diagonals: Everyone is Equal

Albrecht Durer

Copyright © 1988 The MathWorks

31-Oct-2006 19:12:13

Abstract

An introduction to Magic Squares and their meaning

Table of Contents

Chapter 1. Magic Squares Explained

MAGIC Magic square. MAGIC(N) is an N-by-N matrix constructed from the integers 1 through

Generating Reports

Setting Report Output Options

In this section...

Setting Preferences

Specify report output settings using the **Preferences** pane. To open this pane, click **File > Preferences**. A list of available options appears in the following table.

Option	Purpose
`Format ID`	Specify the report output format. Choices include `Acrobat (PDF)`, `Rich Text format`, and `Word document`. The default is `Web (HTML)`.
`Extension`	Automatically sets the file extension of the report file according to the value specified for `Format ID`. The default is `html`.
`Simulink Images`	Specify the format for Simulink® images to include in the report. The default is `PNG 24-bit image`.

Option	Purpose
`Stateflow Images`	Specify the format for Stateflow® charts to include in the report. The default is `PNG 24-bit image`.
`HG Images`	Specify the format for Handle Graphics® images to include in the report. The default is `PNG 24-bit image`.
`View command`	Specify the MATLAB command you want to use to view the report.
`Visible in Report Explorer`	Deselect this check box to make the current output format unavailable in the Report Explorer Interface. 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.
`Use version 1.x environment`	Choose this option to use previous versions of MATLAB Report Generator and Simulink Report Generator interfaces. For more information, see "Using Legacy Interfaces" on page 3-19.
`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.

Setting the Report File Folder

Choose a folder to which you have write privileges, to store the report file. A list of options appears in the following table.

Folder	Option
The same folder as the report template	`Same as report template`
The current working folder	`Present working directory`

Folder	Option
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 page 9-68 on the `Text` component reference page in the MATLAB Report Generator documentation.

Setting the Report File Name

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`.

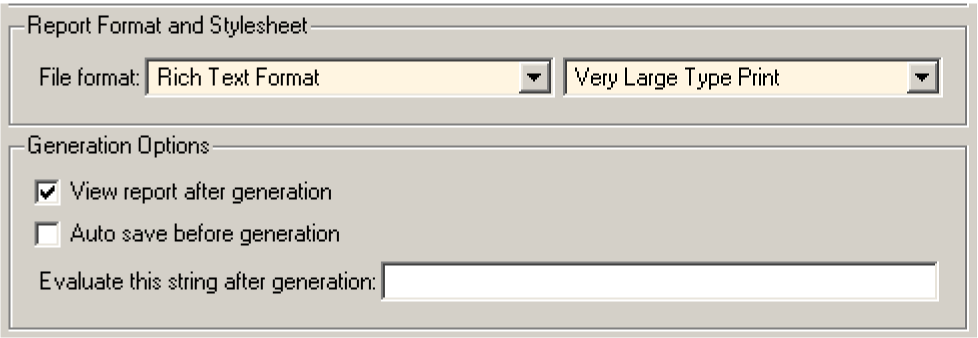
Choose a file name for the report file, using the options listed in the following table.

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

Use `%<VariableName>` notation to specify a file name in the **Custom** text box. For more information, see "%<VariableName> Notation" on page 9-68 on the `Text` component reference page in the MATLAB Report Generator documentation.

Setting the Report Format

Choose the report output format in the **File format** text box. For example, if you want to use Microsoft Word, choose `Word document` or `Rich Text Format`.



Report Format and Stylesheet

File format: Rich Text Format | Very Large Type Print

Generation Options

☑ View report after generation

□ Auto save before generation

Evaluate this string after generation:

Each output format has a default stylesheet associated with it. Specify the stylesheet in the text box next to the **File format** text box.

The following table shows which output format to use with different software, and which stylesheets you can use with each output format.

Viewer	Format	Description	Stylesheets
Adobe Acrobat Reader	Adobe Acrobat (PDF)	Produces a PDF that you can view using Adobe Acrobat Reader software	PDF (see "PDF Stylesheets" on page 3-6)
Web browser	`Web (HTML)` (default)	Use for publishing on the World Wide Web	Web (see "Web Stylesheets" on page 3-7)
Word processor	`Rich Text Format (RTF)` or `Word Document`	Compatible with most word-processing packages, including Microsoft Word software	Print (see "RTF (DSSSL Print) and Word Stylesheets" on page 3-8)
DocBook	`DocBook (XML)`	Produces a report in DocBook format	N/A

Tip To create and use customized styles, see Chapter 6, "Creating Custom Stylesheets".

PDF Stylesheets

PDF 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`	Chapters and sections are not numbered
`Numbered Chapters & Sections`	Chapters and sections are both numbered
`Paginated Sections`	Sections are printed with page breaks
`Custom Header`	Lets you specify custom headers and footers
`Custom Titlepage`	Lets you specify custom title page content and presentation
`Logo stylesheet for PDF`	Lets you specify a logo, such as your company logo, in the header
`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
`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`	Chapters and sections are not numbered
`Numbered Chapters & Sections`	Chapters and sections are both numbered

Note Some Web and Print stylesheets include an automatically generated list of titles. The list of titles includes a list of table titles and a list of figures with titles.

Viewing and Printing Reports

To view your report after generation:

- To view the report automatically, select the **View report after generation** check box in the **Generation Options** section in the Properties pane on the right. When report generation is complete, the viewer associated with the report output format displays the report.
- 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.

To print your report, select the **Print** option from the viewer.

Converting English Strings to Other Languages

Versions 2.0 and later of the MATLAB Report Generator and Simulink Report Generator software use the system language settings through the Sun™ 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.

Autosaving Report Templates

To automatically save the report template before you generate a report, select **Auto save before generation** from the **Generation Options** section of the Properties pane on the right.

Reporting Compiled Simulink Model Data

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. To ensure that a report reflects compiled values, select the **Compile model to report on compiled information** option. 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 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.

Regenerating Images

By default, the **Regenerate Simulink and Stateflow Images** option in the **Generation Options** section of the Properties pane on the right is cleared. As a result, previously generated images are not regenerated each time you generate a report, so reports generate faster. If you make changes that affect a generated image, select this option.

Including a Report Description

Use the **Report Description** field in the Properties pane on the right to record notes and comments about your report template. This text appears in the Properties pane on the right when you select a report template in the Outline pane on the left.

Converting XML Documents

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`.

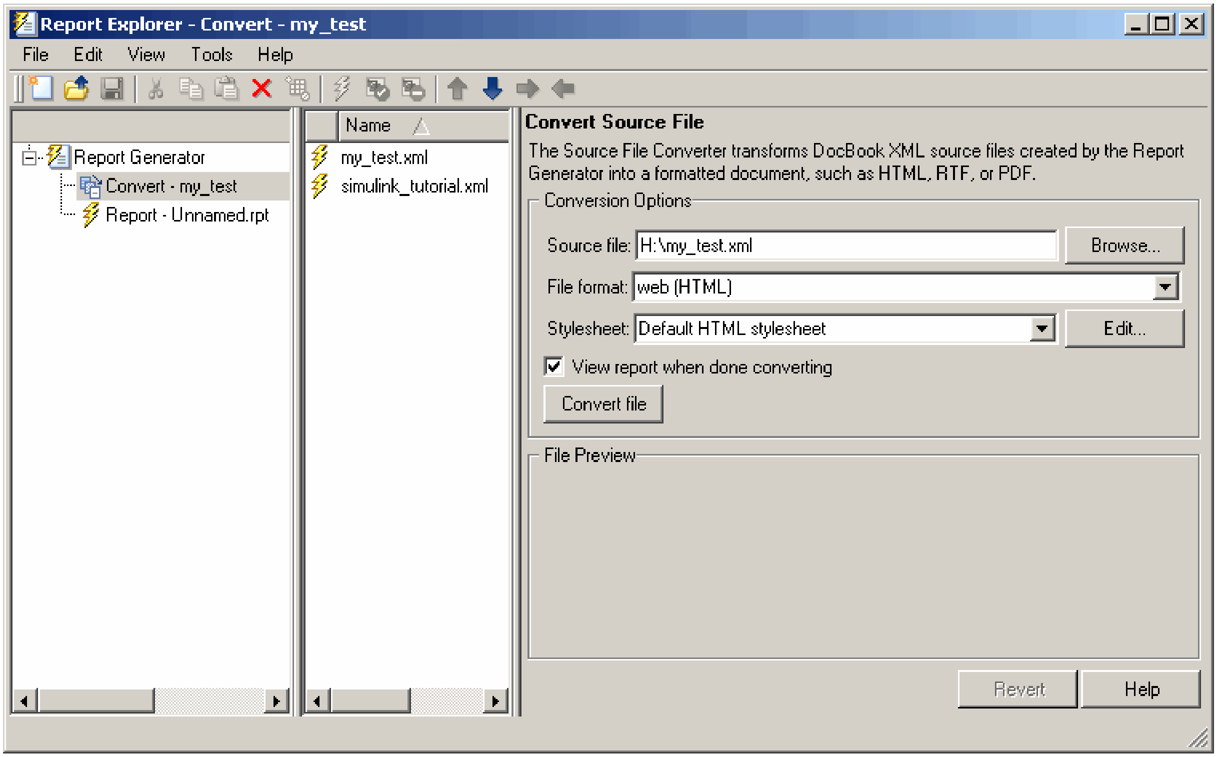
Note The report-generation process can only convert XML source files created by the latest version of the software.

Converting 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.



Convert Source File

The Source File Converter transforms DocBook XML source files created by the Report Generator into a formatted document, such as HTML, RTF, or PDF.

Conversion Options

Source file: H:\my_test.xml

File format: web (HTML)

Stylesheet: Default HTML stylesheet

☑ View report when done converting

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 “Setting the Report Format” on page 3-4.

For more information about customizing stylesheets, see Chapter 6, “Creating Custom Stylesheets”.

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**.

Converting XML Documents Using the Command Line

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

Editing 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.

Creating Log Files

A log file describes a report template's report-generation settings and components. A log file can be used for many purposes, including:

- As a debugging tool
- As a reference to a report template
- To share information about a report template through e-mail

A log file includes the following information:

- Report template 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 template.

Generating MATLAB Code from Report Templates

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

To generate a MATLAB file, load a report template 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 report template filename, preceded by "build."

Generating Reports from MATLAB Files

This example generates a MATLAB file from the `figloop_tutorial.rpt` report template, 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.

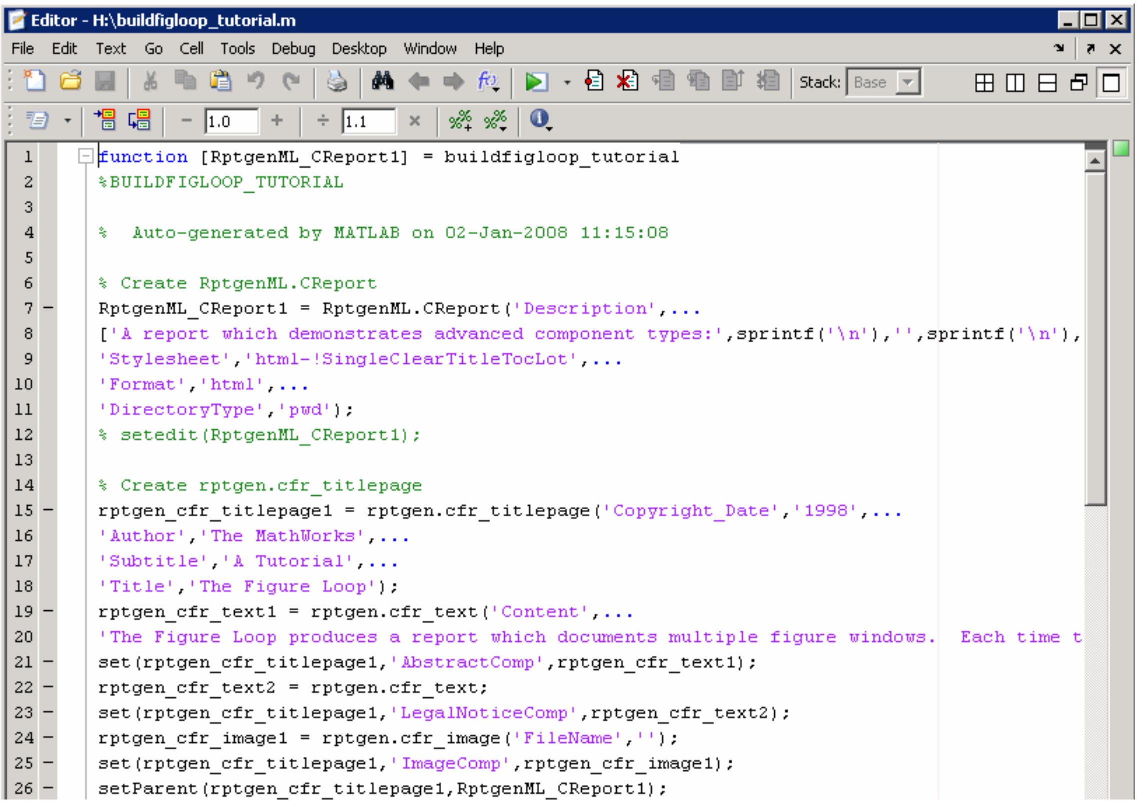
1. Start the Report Explorer by entering `report` in the MATLAB Command Window.

2. In the Options pane in the middle, double-click `figloop_tutorial.rpt` to open its report template.

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 template. It saves this code in the `buildfigloop_tutorial.m` file in the folder you specify. Part of this file appears in the following figure.

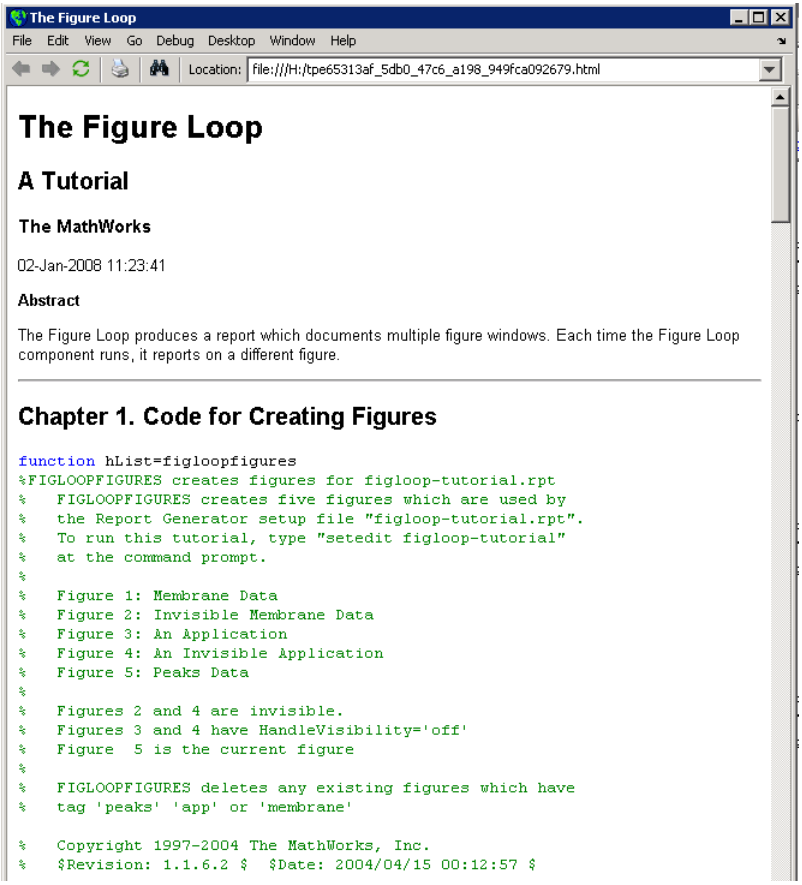

Editor - H:\buildfigloop_tutorial.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1

```
function [RptgenML_CReport1] = buildfigloop_tutorial
%BUILDFIGLOOP_TUTORIAL

%  Auto-generated by MATLAB on 02-Jan-2008 11:15:08

% Create RptgenML.CReport
RptgenML_CReport1 = RptgenML.CReport('Description',...
['A report which demonstrates advanced component types:',sprintf('\n'),'',sprintf('\n'),
'Stylesheet','html-!SingleClearTitleTocLot',...
'Format','html',...
'DirectoryType','pwd');
% setedit(RptgenML_CReport1);

% Create rptgen.cfr_titlepage
rptgen_cfr_titlepage1 = rptgen.cfr_titlepage('Copyright_Date','1998',...
'Author','The MathWorks',...
'Subtitle','A Tutorial',...
'Title','The Figure Loop');
rptgen_cfr_text1 = rptgen.cfr_text('Content',...
'The Figure Loop produces a report which documents multiple figure windows.  Each time t
set(rptgen_cfr_titlepage1,'AbstractComp',rptgen_cfr_text1);
rptgen_cfr_text2 = rptgen.cfr_text;
set(rptgen_cfr_titlepage1,'LegalNoticeComp',rptgen_cfr_text2);
rptgen_cfr_image1 = rptgen.cfr_image('FileName','');
set(rptgen_cfr_titlepage1,'ImageComp',rptgen_cfr_image1);
setParent(rptgen_cfr_titlepage1,RptgenML_CReport1);
```

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
File Edit View Go Debug Desktop Window Help
Location: file:///H:/tpe65313af_5db0_47c6_a198_949fca092679.html

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 $
```

Working with Legacy Report Templates

In this section…

What Are Legacy Report Templates?

Legacy report templates are files that you created in previous versions of the MATLAB Report Generator or Simulink Report Generator software. You can load legacy report templates in the current version of the Report Explorer. Alternatively, you can use earlier versions of the product interfaces to work with legacy report templates.

Enabling Legacy Interfaces

Enable the previous versions of the interfaces at the MATLAB command line:

```
RptgenML.v1mode(true)
```

The result is:

```
ans =

     1
```

A logical `1` indicates that Version 1 mode is on. To turn off Version 1 mode, enter:

```
RptgenML.v1mode(false)
```

The result is:

```
ans =

     0
```

A logical `0` indicates that Version 1 mode is off.

Using Legacy Interfaces

When you enable Version 1 mode, you can use previous versions of MATLAB Report Generator or Simulink Report Generator interfaces.

When using earlier versions of these interfaces, the following restrictions apply:

- You cannot use a report template created in the latest version of the MATLAB Report Generator or Simulink Report Generator software with earlier interfaces. If you save a legacy report template in the latest version of the software, you can no longer work with the same report template in the old interface.
- You cannot use the old interfaces with the latest version of the MATLAB Report Generator or Simulink Report Generator software. The Report Template Editor can coexist with the Report Explorer, but The MathWorks™ does not recommend using these interfaces together.

Troubleshooting

In this section…

Managing Memory Usage

By default, the MATLAB software sets a limit of 100 MB on the amount of memory the Sun 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.

Running 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.

Note This option is available on UNIX® platforms only. (UNIX is a registered trademark of The Open Group in the United States and other countries.)

Increasing the MATLAB JVM Memory Allocation Limit

Increase the amount of JVM memory available by increasing the MATLAB JVM memory allocation limit.

1 Create a file named `java.opts`. In this file, include the `-Xmx` option, specifying the amount of memory you want to allocate to the JVM software. For example, to increase the JVM memory allocation limit to 128 MB, use the following syntax in the `java.opts` file:

```
-Xmx128m
```

Caution To avoid virtual memory thrashing, never set the `-Xmx` option to more than 66% of available physical RAM.

2. Put the `java.opts` file into the appropriate folder. Where you put the file depends on what operating system you are running.

   - On UNIX systems, put the `java.opts` file in the folder where you intend to start the MATLAB software session, and navigate to that folder before doing so.
   - On Microsoft Windows® systems:

     a. Put the `java.opts` file in a folder where you intend to start the MATLAB software session.

     b. Create a shortcut to the MATLAB software.

     c. Right-click the shortcut and select **Properties**.

     d. In the Properties dialog box, specify the MATLAB startup folder for the name of the folder in which you created the `java.opts` file.

   - On Apple® Mac OS® systems, put the `java.opts` file into the `/Applications/matlab` folder.

Displaying HTML Reports 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 template file.

4

Working with Components

About Components

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 template. You use components to customize the appearance and output of reports.

The following sections provide examples of how to use different types of components.

Working with Looping Components

In this section...

About Looping Components

A *looping component* runs its child components a specified number of times. There are several looping components, including logical loops and Handle Graphics loops.

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.

The following 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, it applies its child components to Handle Graphics figures. The `figloop-tutorial` report template, included with the MATLAB Report Generator software, creates a report that documents multiple Handle Graphics figures.

For background information, see "Understanding Handle Graphics Objects" in the MATLAB documentation.

Editing the Figure Loop Tutorial Report Template

To edit the figure loop tutorial report template, at the MATLAB command prompt, enter the following command:

```
setedit figloop-tutorial
```

Displaying Figures

To display the Handle Graphics figures, enter the following at the MATLAB command line:

```
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.

“Figure Properties” on page 4-5 describes the properties of figures used in this tutorial.

Editing Figure Loop Components

Select the Figure Loop component called `Figure Loop Section 1` from the Outline pane on the left. The Properties pane for the Figure Loop component appears.

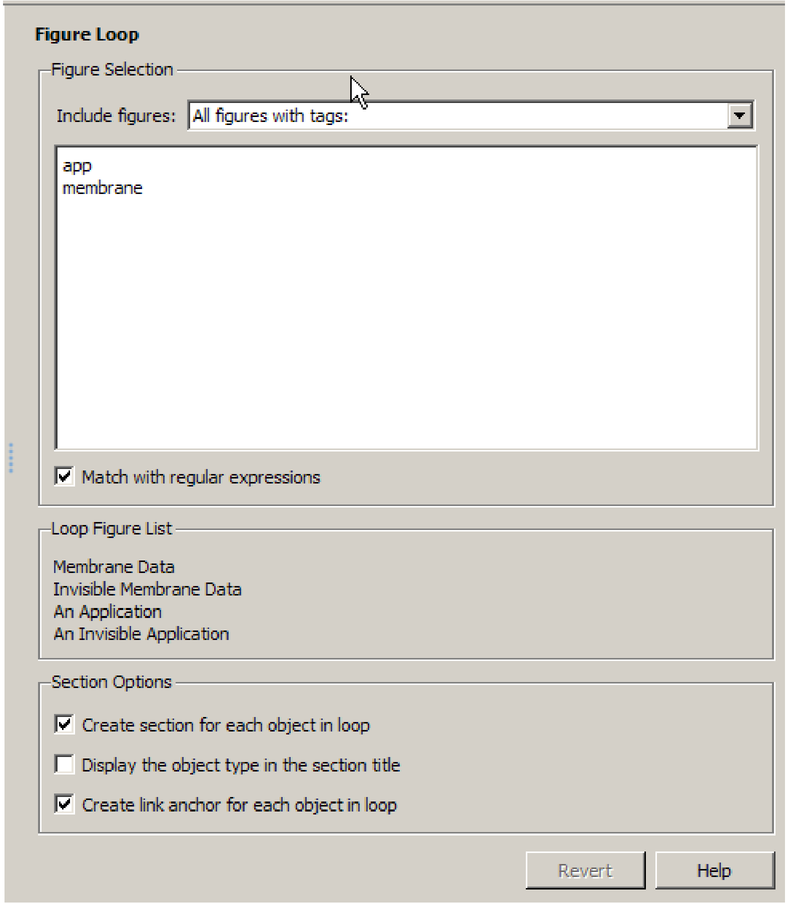


Figure Loop

Figure Selection

Include figures: All figures with tags:

app
membrane

☑ Match with regular expressions

Loop Figure List

Membrane Data
Invisible Membrane Data
An Application
An Invisible Application

Section Options

☑ Create section for each object in loop

☐ Display the object type in the section title

☑ Create link anchor for each object in loop

Figure Properties

You can control what figures get displayed in the report, based on the properties of each figure.

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

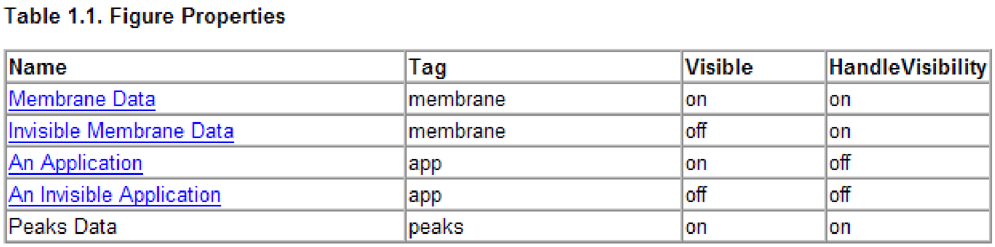


Table 1.1. Figure Properties

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 tutorial, you will not change these properties. For background information, see "Setting and Querying Property Values" in the MATLAB documentation.

Looping on the Current Figure

To include only the current figure in the report, select `Current figure only` from the **Include figures** selection list. In this case, "current figure" refers to the figure that is current when the report generates. This may not be the same figure 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.

Looping on Visible Figures

To include snapshots of all visible figures in your report, select `Visible figures` in the **Include figures** selection list. 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. Click the **Report** button in the Report Explorer toolbar to generate the report.

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

Looping on Figures with Tags

To include figures with specified tags in the report:

1 Select the `All figures with tags` option in the **Include figures** selection list.

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.

Modifying 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. With this option selected, the loop automatically creates an individual section for each object found in the loop. It uses 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. Enable this option by selecting **Create section for each object in loop**. The generated report precedes section titles with object titles.

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 **Display the Object Type in the Section Title** option.

4 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
```

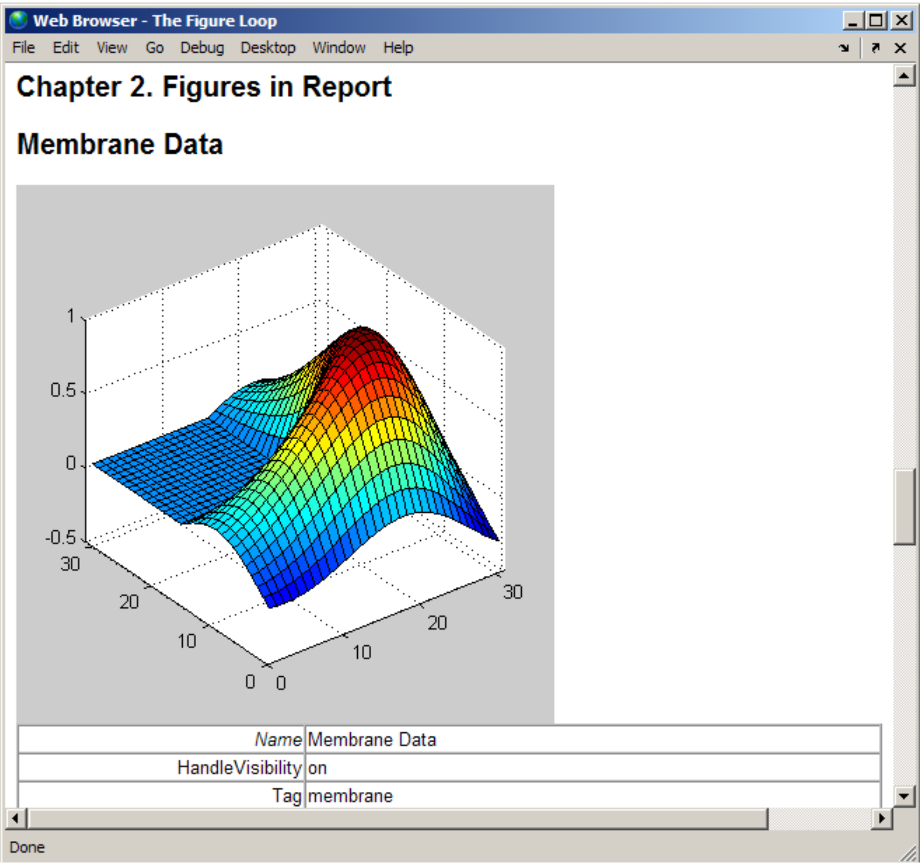
Create a Link Anchor for Each Object in Loop. Selecting this option creates a hyperlink to the object in the generated report.

Working with Property Table Components

In this section...

About Property Table Components

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


Web Browser - The Figure Loop
File Edit View Go Debug Desktop Window Help
1
0.5
0
-0.5
30
20
10
0
0
10
20
30
Done

Chapter 2. Figures in Report

Membrane Data

Name	Membrane Data
HandleVisibility	on
Tag	membrane

Many types of Property Table components are available, including:

- MATLAB Property Table
- Simulink Property Table
- Stateflow Property Table

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

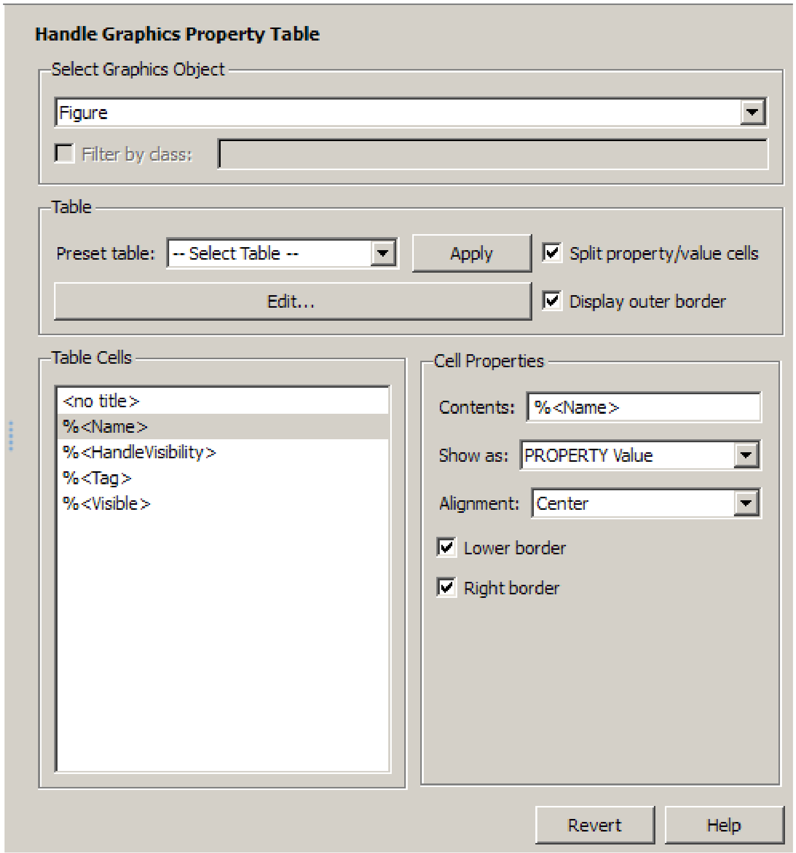
Opening the Example Report Template

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
```

Examining 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.



Handle Graphics Property Table

Select Graphics Object: Figure

☐ Filter by class:

Preset table: -- Select Table --

☑ Split property/value cells

☑ Display outer border

Table Cells:

<no title>
%<Name>
%<HandleVisibility>
%<Tag>
%<Visible>

Contents: %<Name>

Show as: PROPERTY Value

Alignment: Center

☑ Lower border

☑ Right border

Selecting 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.

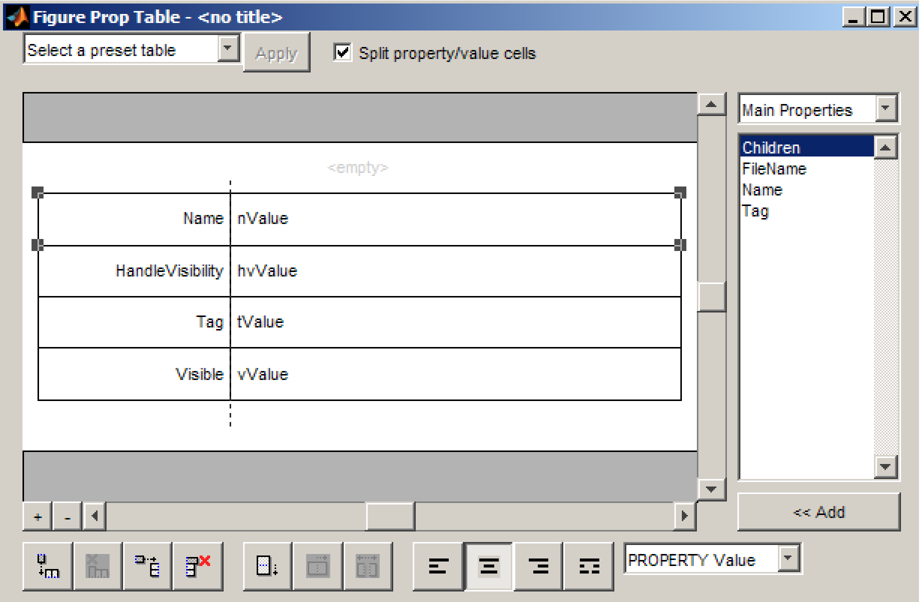
You can select a different object type on which to report in the **Object type** list in the component's Properties pane.

Displaying Property Name/Property Value Pairs

Splitting 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.



☑ Split property/value cells

Name	nValue
HandleVisibility	hvValue
Tag	tValue
Visible	vValue

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.

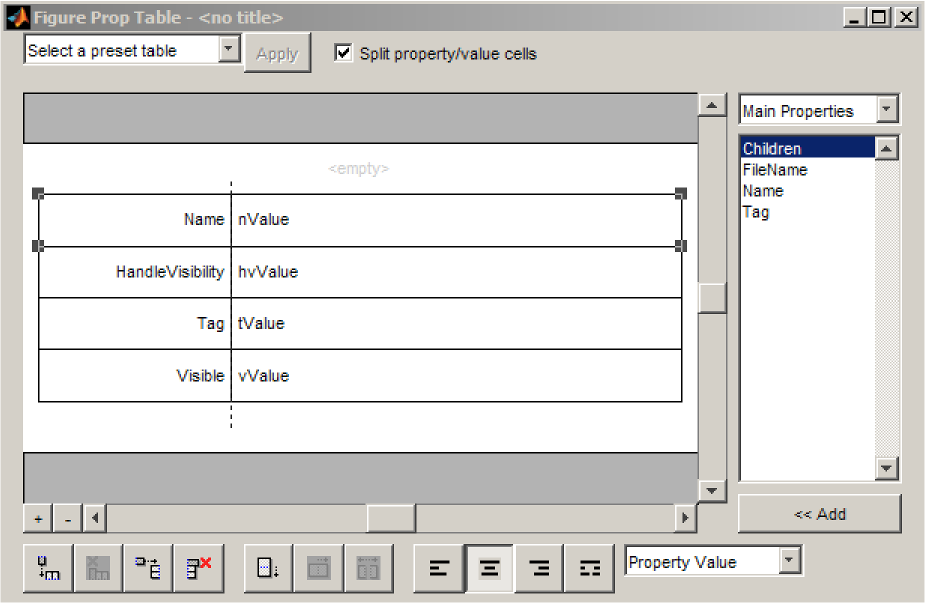


Figure Prop Table - <no title>

☑ Split property/value cells

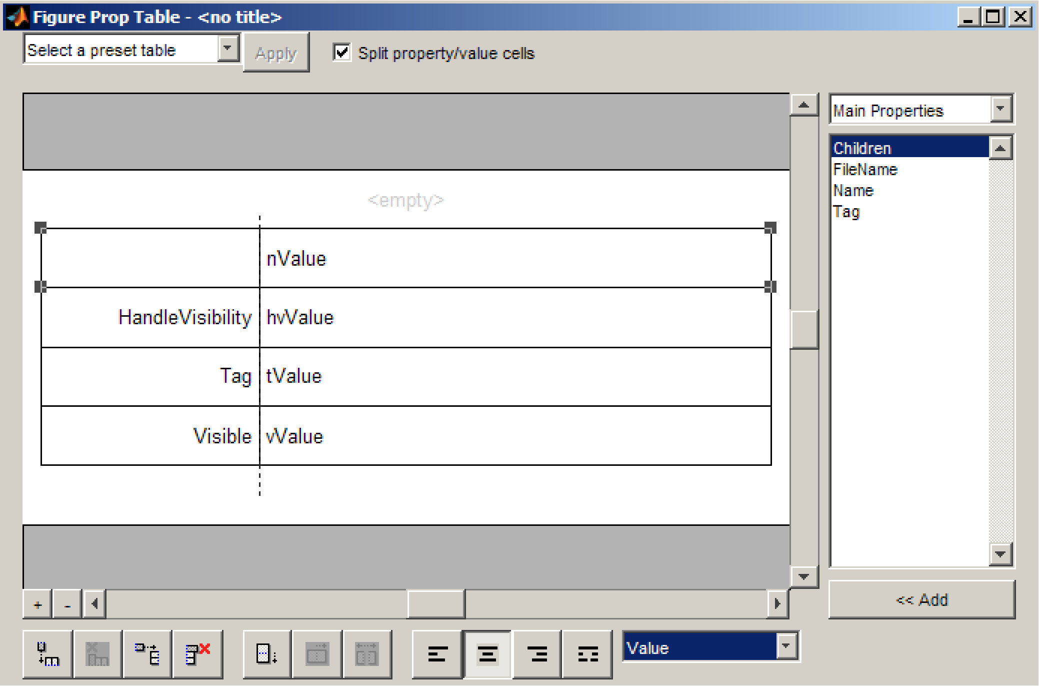
<empty>

Name	nValue
HandleVisibility	hvValue
Tag	tValue
Visible	vValue

Property Value

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.

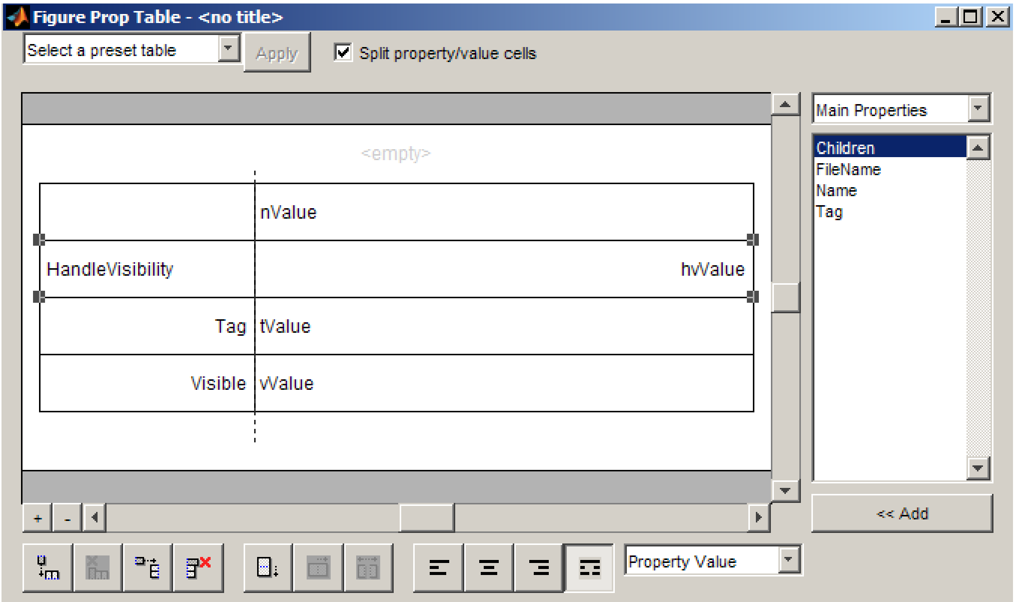


	nValue
HandleVisibility	hvValue
Tag	tValue
Visible	vValue

Formatting Options

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

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



Editing Table Titles

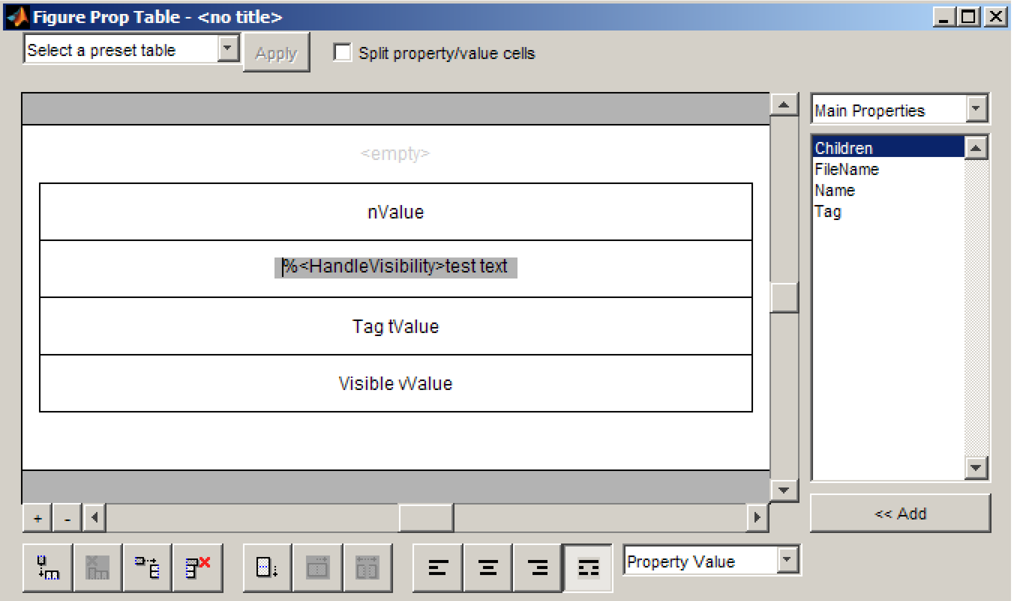
Table titles can contain properties and text. By default, the title of a table is the same as the value of the `%<Name>` property. You can modify this property to modify the table title.

Note Table titles are always in nonsplit mode.

Entering 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.



<empty>

nValue

%<HandleVisibility>test text

Tag tValue

Visible vValue

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.

Adding, Replacing, and Deleting Properties in Tables

Adding Table Properties

To add a Handle Graphics property to a table:

1. In the Figure Prop 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 "Entering Text into Table Cells" on page 4-15. For information about adding new table rows, see "Adding and Deleting Columns and Rows" on page 4-18.

Replacing Table Properties

To replace a property in a cell of a table in split mode, follow the instructions in "Adding Table Properties" on page 4-16.

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

Deleting Table Properties

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

Displaying or Hiding Cell Borders

To toggle cell borders on and off:

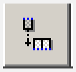
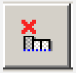
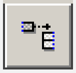
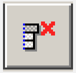
1 Place your cursor in a cell and right-click to invoke its context menu.

2 Choose **Cell borders > Top**, **Bottom**, **Right**, or **Left** to toggle the specified border on or off.

Adding and Deleting 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.

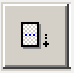
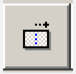
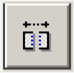
Button	Action
	Add column (added to the left of the selected column)
	Delete selected column
	Add row (added above the selected row)
	Delete selected row

Resizing Columns

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

Merging and Splitting 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

Zooming and Scrolling

Zooming

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

Scrolling

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

Selecting Tables

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.

Working with Summary Table Components

In this section...

About Summary Table Components

Summary Table components insert tables that include specified properties for objects into generated reports. 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.

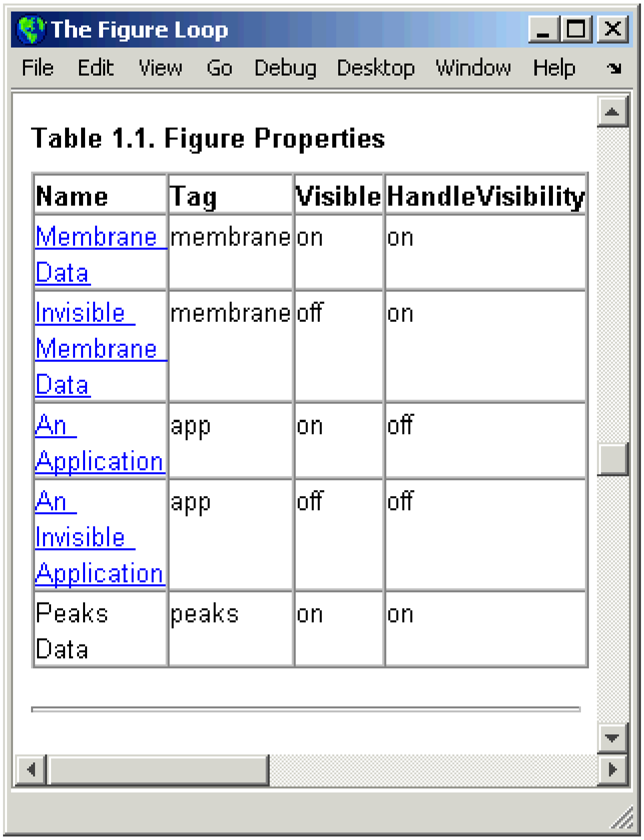


Table 1.1. Figure Properties

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
- Stateflow Summary Table

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

Opening 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:

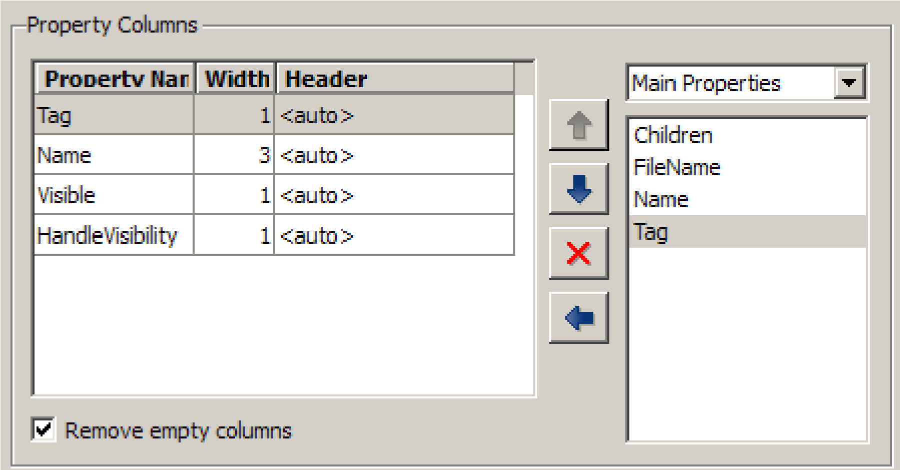
```
setedit figloop-tutorial
```

Selecting 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.

Adding and Removing 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.



Property Columns

Property Nar	Width	Header
Tag	1	<auto>
Name	3	<auto>
Visible	1	<auto>
HandleVisibility	1	<auto>

☑ Remove empty columns

Main Properties

- Children
- FileName
- Name
- Tag

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 interface, 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 page 9-68 on the `Text` component reference page in the MATLAB Report Generator documentation.

Setting 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.

Setting 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:

- "Looping on the Current Figure" on page 4-6
- "Looping on Visible Figures" on page 4-6
- "Looping on Figures with Tags" on page 4-6

Creating Custom Components

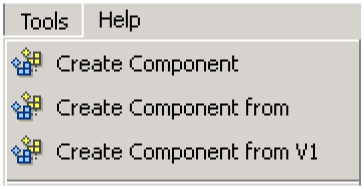
About Custom Components

In most cases, the components provided with the MATLAB Report Generator software should be more than adequate for your needs. You can, however, create custom components if you want to generate a report via functionality that is not available in the standard MATLAB Report Generator components. For example, you can create a component that inserts a corporate logo into your report, or a component that plots data.

Component Creation Process

To create a component:

1. Open the Report Explorer.
2. Select one of the component creation choices from the **Tools** menu:

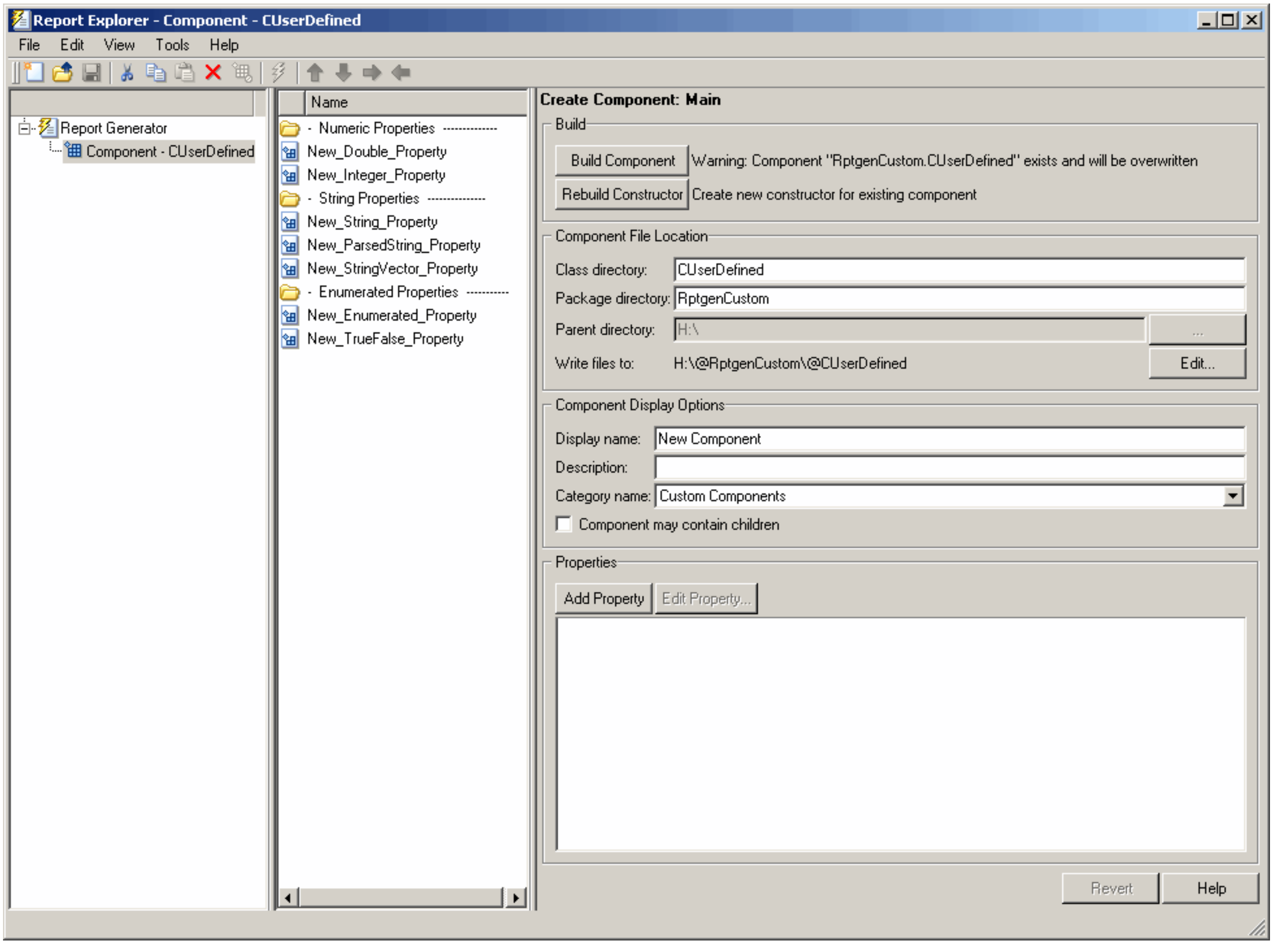


- To create a custom component, select **Create Component**.
- To create a custom component from an existing component, select **Create Component from**.
- To create a component from an existing version 1 component, select **Create Component from V1**.

Tip You can also create a custom component by clicking the **Create a new user-defined reporting component** link in the Report Explorer Properties pane on the right.

The Report Explorer displays as follows:

- The Outline pane on the left displays the structure of components you create.
- The Options pane in the middle lists properties you add to components.
- The Properties pane on the right specifies the behavior of component properties.



Create Component: Main

Build

Warning: Component "RptgenCustom.CUserDefined" exists and will be overwritten

Create new constructor for existing component

Component File Location

Class directory: CUserDefined

Package directory: RptgenCustom

Parent directory: H:\

Write files to: H:\@RptgenCustom\@CUserDefined

Component Display Options

Display name: New Component

Description:

Category name: Custom Components

☐ Component may contain children

Properties

3 Specify properties of the component in the Properties pane of the Report Explorer. For more information, see “Defining Components” on page 5-6.

4 Specify tasks you want the component to perform by editing the MATLAB files that comprise the framework of the component. For more information, see “Defining Component Tasks” on page 5-14.

5 Build the component. For more information, see “Building Components” on page 5-12.

After you build the custom component, you can use it to specify options for your generated report in the report template.

Note You must restart the MATLAB software session before using a newly created or rebuilt component.

Defining Components

In this section...

Required Component Data

You must specify the following information when you create a component:

1. The path where you want to put the folder that contains all files for the component. For information on how to specify this folder, see “Specifying the Location of Component Files” on page 5-6.

2. Properties of the component. For more information, see “Specifying Component Properties” on page 5-9.

3. Display options for the component, including its display name, category, and description. For more information, see “Setting Component Display Options” on page 5-7.

Specifying the Location of Component Files

You can create components that perform similar functions and group them in *Package Directories*. Each package folder must have a *Parent Directory* that is in the MATLAB path. When you build a new component, the MATLAB Report Generator software creates files that make up the component. These files are stored in the folder structure `<parent>/@package_name/@class_name`.

Specify these directories in the following fields in the **Component File Location** area of the Properties pane:

1 **Class Directory Field**. Specify a class name for your component. The build process creates a folder with the name you specify and places the component's files in it. The class folder name must be unique for each component in the package. By convention, component class names begin with an uppercase or lowercase letter `c`; for example, `cUserDefinedComponent`.

2 **Package Directory Field.** Specify the folder in which to store files for groups of components you create. Files for each component are stored in a subfolder with the name you entered into **Class Directory Field**.

3 **Parent Directory Field**. Specify this folder when you create a package for the first time. This folder is the parent folder of the Package Directory.

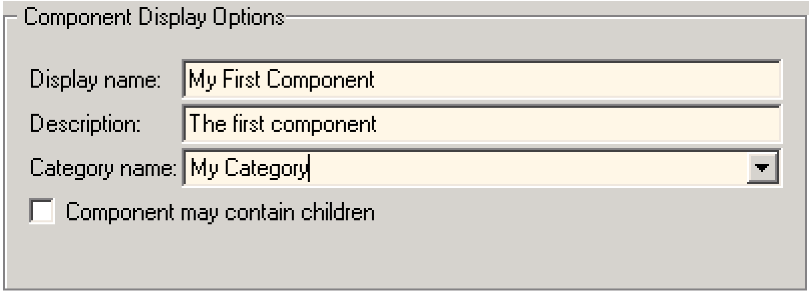
Setting Component Display Options

You can specify how you want your component to appear in the Report Explorer by entering data in the **Component Display Options** area of the Properties pane. Enter the following information:

1 **Display Name.** Specify a display name for the component to appear in the list of components for its associated category. Component categories and display names appear in the Options pane in the middle of the Report Explorer.

   For information on specifying component categories, see step 3, **Category Name**.

   The following example shows how to create a component called `My First Component` in a category called `My Category`.



Component Display Options

Display name: My First Component

Description: The first component

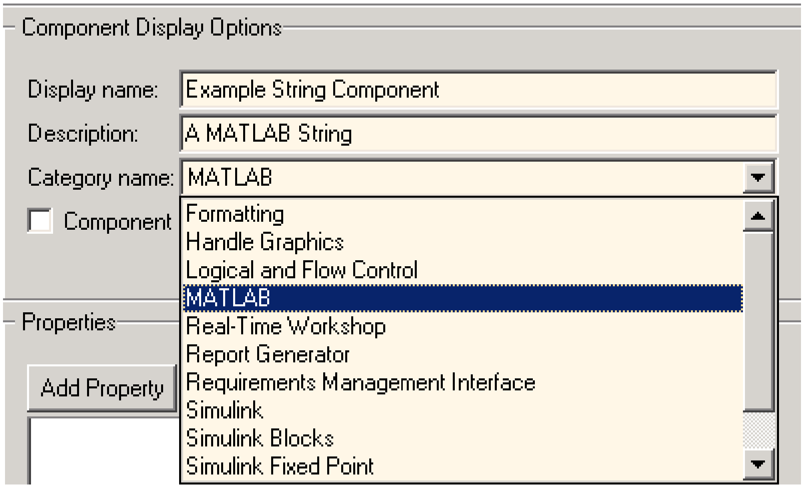
Category name: My Category

□ Component may contain children

2 **Description.** Enter a description for the component. This description appears when you click the component name or category name in the Options pane in the middle of the Report Explorer. Make the description informative, but brief.

3 **Category Name.** Specify the category of components to which the new component belongs. The component appears under this category in the Options pane in the middle of the Report Explorer.

Predefined choices appear in the **Category name** list. Select a component category from this list.



Component Display Options

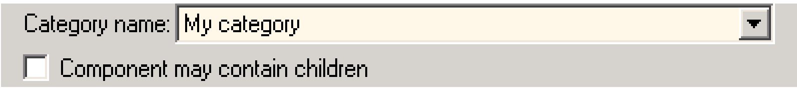
Display name: Example String Component

Description: A MATLAB String

Category name: MATLAB

□ Component

To create a custom component category, type the name for the category into the **Category name** field. This category name appears in the list of available categories in the Report Explorer.



Category name: My category

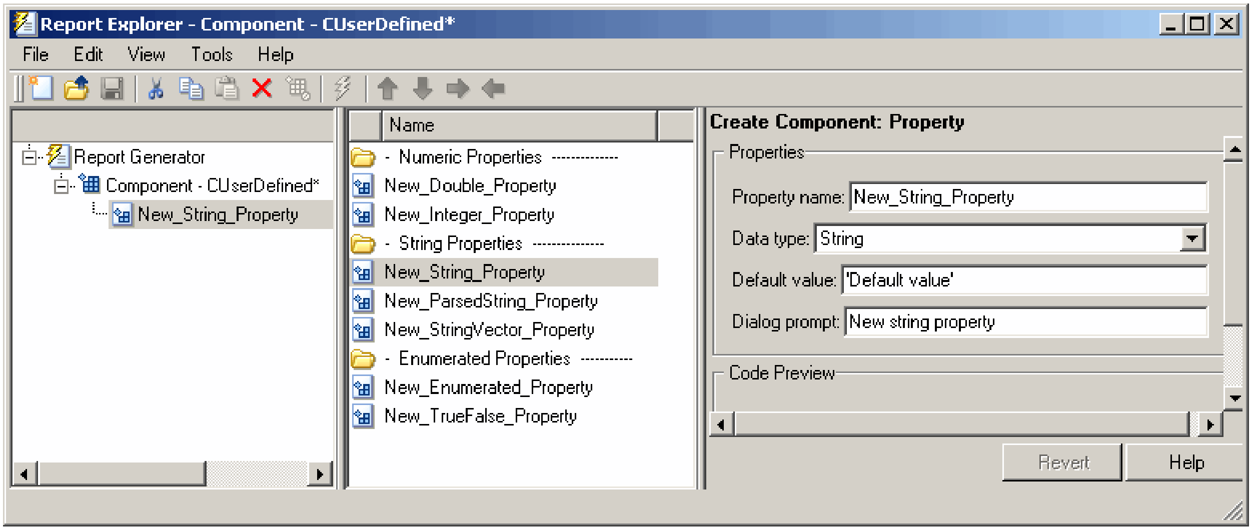
□ Component may contain children

4 **child components**.

Select the **Component may contain children** check box if you want the component to have child components. Child components appear under the component in the Report Explorer hierarchy. During report generation, the component runs all child components and includes their output in the report.

Specifying Component Properties

Component properties determine how a component behaves and what information it inserts into a report. To see the current value of a component's property, double-click it in the Outline pane on the left in the Report Explorer. For example, the following figure displays the property values for `New_String_Property`.



Report Explorer - Component - CUserDefined*

Create Component: Property

Properties

Property name: New_String_Property

Data type: String

Default value: 'Default value'

Dialog prompt: New string property

Code Preview

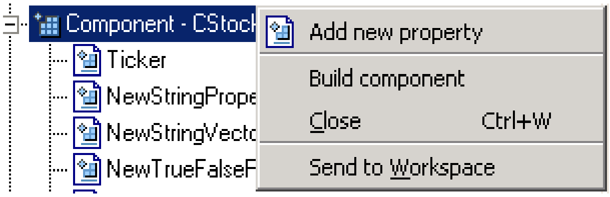
File
Edit
View
Tools
Help
Report Generator
Component - CUserDefined*
New_String_Property
Name
- Numeric Properties -------------
New_Double_Property
New_Integer_Property
- String Properties --------------
New_String_Property
New_ParsedString_Property
New_StringVector_Property
- Enumerated Properties ----------
New_Enumerated_Property
New_TrueFalse_Property
Revert
Help

Adding Properties to Components

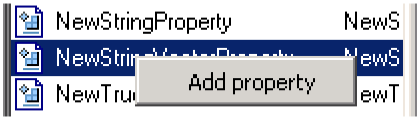
You add properties to a component from the properties list. Each property has a default value that you can modify as needed.

There are several ways to add properties to components:

1. Right-click the name of the component to which you want to add properties in the Outline pane on the left. Select **Add new property** from its context menu.

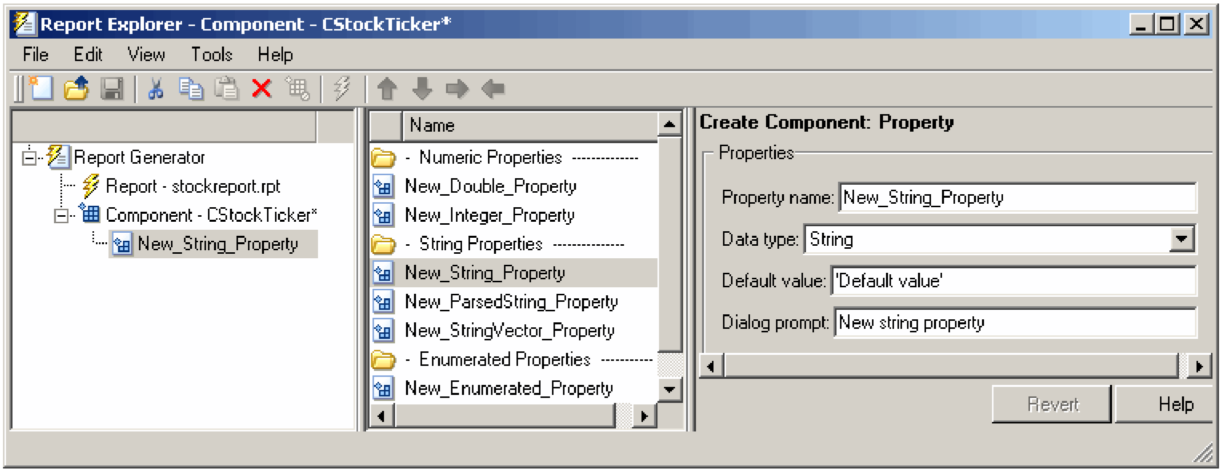
   

2. Right-click the name of a predefined property in the Options pane in the middle. From the context menu, select **Add property**.

3. Left-click the name of a property in the Options pane, and then drag it on top of a component in the Outline pane on the left.

4. Double-click the property name in the Options pane in the middle. The property is added to the component and property values appear in the Properties pane on the right.



Report Explorer - Component - CStockTicker*
File
Edit
View
Tools
Help
Report Generator
Report - stockreport.rpt
Component - CStockTicker*
New_String_Property
Name
- Numeric Properties ------------
New_Double_Property
New_Integer_Property
- String Properties -------------
New_String_Property
New_ParsedString_Property
New_StringVector_Property
- Enumerated Properties ---------
New_Enumerated_Property
Revert
Help

Create Component: Property

Properties

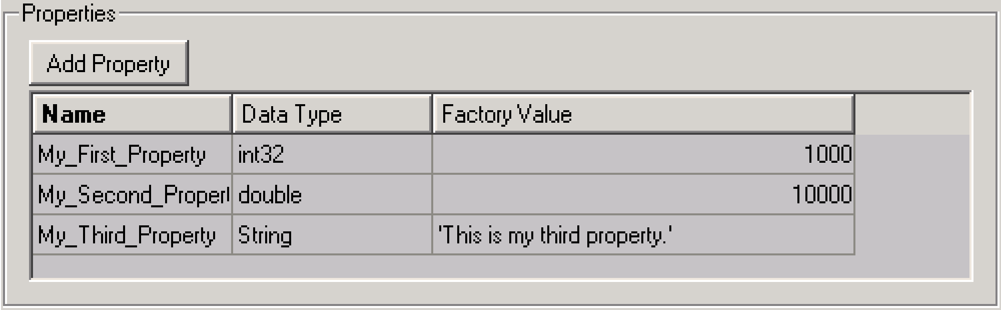
Property name: New_String_Property

Data type: String

Default value: 'Default value'

Dialog prompt: New string property

5 Click the **Add Property** button on the Properties pane on the right.



Name	Data Type	Factory Value
My_First_Property	int32	1000
My_Second_Propert	double	10000
My_Third_Property	String	'This is my third property.'

Specifying Component Properties

1 **Property Name.** Create a name for the new property. A property name must be a valid MATLAB variable name, and must be unique within a component.

2 **Data Type.** Specify the property's data type. Options are:

- `Double`
- `Enumeration`
- `Integer`

- `String`
- `String Vector`
- `%<Parsed String>`

  Use this data type to include the value of a variable in the MATLAB workspace in a component. For more information about this data type, see "%<VariableName> Notation" on page 9-68 on the `Text` reference page.
- `True/False`

For more details about MATLAB data types, see "Classes (Data Types)" in the MATLAB Programming Fundamentals documentation.

3 **Default Value.** Set a default value for the property. The default value must be compatible with the data type. If incompatibilities exist between the default value and the data type, the component might not build.

4 **Dialog Prompt.** This text appears next to the widget on the component's dialog box. It indicates what the property does and how it affects report generation.

Note When the component builds, a colon is appended to your entry in the **Dialog prompt** field. Your entry appears in the Properties pane with the colon appended.

Modifying Existing Components

Report components are modifiable. You can derive a new component from an existing component by double-clicking the name of the component and modifying its values and properties.

Building Components

After you have entered all data required for defining the component, you build it by clicking the **Build Component** button. The build process creates all files needed for the component and stores them in the specified folder. For more information about specifying where components are stored, see "Specifying the Location of Component Files" on page 5-6.

Note Existing files in this location are overwritten.

Rebuilding Existing Components

To add, remove, or change properties of an existing component, use the **Rebuild Constructor** button. This button becomes active only after you have previously created a component using the **Build Component** button. To activate the **Rebuild Constructor** button, specify the **Package name** and **Class name** for an existing component. These fields are located in the **Component File Location** area of the Properties pane.

If you select a component using **Tools > Create component from**, the component's fields are filled in automatically and the button becomes active.

After you have finished modifying the component, click the **Rebuild Constructor** button to rebuild the component. Writable files in the component's folder location are not overwritten.

Removing a Component

To remove a component:

1. Delete its class folder, `<root>/@package_name/@class_name`. If the component you want to remove is the only component in the package, delete the entire package.

2. Edit `<root>/@package_name/rptcomps2.xml` to remove the XML element that registers the component.

Defining Component Tasks

In this section...

About Component Customization

Building a component creates MATLAB files in your MATLAB workspace. You can specify tasks you want your component to perform by editing these MATLAB files.

Note You *must* specify the format and content of your report output by editing `execute.m`. This file is called during report generation to invoke your component's tasks. You can optionally specify additional component properties and behavior by editing other MATLAB files.

For more information, see the following sections:

- "Required Customization: Specifying Format and Content of Report Output" on page 5-15
- "Changing a Component's Outline String in the Report Explorer Hierarchy" on page 5-17
- "Modifying the Appearance of Properties Dialog Boxes" on page 5-18
- "Specifying Additional Component Properties" on page 5-18

Required Customization: Specifying Format and Content of Report Output

After you build the component, specify the format and content of your report output by editing the `execute.m` file.

The `execute` command has the following syntax:

```
out = execute(thisComp, parentDoc)
```

Where:

- `thisComp` is a handle to the component that you are running.
- `parentDoc` is a handle to the document that you are generating.
- `out` is a Document Object Model (DOM) node or string to add to the report.

  For information on manipulating DOM nodes, see `xmlwrite` in the MATLAB documentation.

One or more default lines of code within the `execute.m` file show each property for the component. Here is an example of a component property line within an `execute.m` file:

```
pstring = thisComp.NewStringProperty;  % New string property;
```

The following sections describe how to edit `execute.m` to create additional report elements.

Creating Tables

To create a table, replace the `Source` property value with the name of a cell array or structure:

```
out = execute(rptgen.cfr_table(...
'Source', tableSrc,...
'numHeaderRows',1,...
'TableTitle','Example Title'),...
 parentDoc);
```

For more information, enter `help(rptgen.cfr_table)` at the MATLAB command line.

Creating Lists

To create a list, replace the Source property value with the name of a cell vector:

```
out = execute(rptgen.cfr_list(...
'Source', listSrc,...
'ListStyle','orderedlist',...
'ListTitle','Example List'),...
 parentDoc);
```

For more information, enter help(rptgen.cfr_list) at the MATLAB command line.

Creating Text

To create text, replace the ParaText property value with a text string:

```
out = execute(rptgen.cfr_paragraph(...
'ParaText', paraSrc,...
 parentDoc);
```

For more information, enter help(rptgen.cfr_paragraph) at the command line.

Creating Figures

To create figures, specify a figure in the FigureHandle property value.

```
figSrc = gcf;
out = execute(rptgen_hg.chg_fig_snap(...
'FigureHandle', figSrc,...
'Title', '',...
'isResizeFigure', 'manual',...
'PrintSize', [6 4],...
'PrintUnits', 'inches'),...
 parentDoc);
```

For more information, enter help(rptgen_hg.chg_fig_snap) at the MATLAB command line.

Running Child Components

The following code runs child components. The first line calls `execute.m` for child components. The second line appends the results of running the child components to the report:

```
childOut = thisComp.runChildren(parentDoc);
out = parentDoc.createDocumentFragment(out, childOut);
```

Changing a Component's Outline String in the Report Explorer Hierarchy

To change the string used to describe the component in the Report Explorer hierarchy, edit the `getOutlineString` MATLAB file. By default, `getoutlinestring` returns the display name of the component. The `getOutlineString` command has the following syntax:

```
olstring = getOutlineString(thisComp)
```

Where:

- `thisComp` is the component whose description you are specifying.
- `olstring` is a single-line string that displays information about the component. It can contain a maximum of 32 characters.

Customize the string to include additional information about the component, such as information about its properties. In the following example, the `truncatestring` function converts input data into a single-line string. If the data is empty, the second argument is the return value, The third argument is the maximum allowed size of the resulting string.

```
cInfo = '';
pstring = rptgen.truncateString(thisComp.string,'<empty>',16);
```

Use a dash (-) as a separator between the name and additional component information, as follows:

```
if ~isempty(cInfo)
   olstring = [olstring, '-', cInfo];
end
```

Modifying the Appearance of Properties Dialog Boxes

You can edit the `getdialog boxeschema` file to control most aspects of dialog box layout, including:

- Creation and placement of widgets
- Organization of widgets into panes
- Creation of the top-level display within which panes reside

The syntax of the `getdialog boxeschema` command is:

```
dlgstruct = getdialog boxeschema(thisComp, name)
```

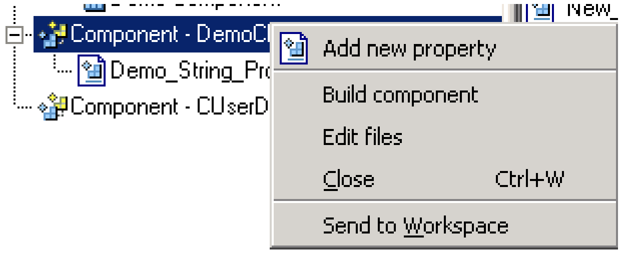
Where:

- `thisComp` is the instance of the component being edited.
- `name` is a string that is passed to `getdialog boxeschema` to build a specific type of pane. Usually, `name` is empty in the Report Explorer.

Note The MathWorks does not recommend modifying fields that are not explicitly included in this file. These fields are subject to change in future releases.

Specifying Additional Component Properties

You can edit additional MATLAB files to customize your component further. To access these files, right-click the component in the Outline pane on the left in the Report Explorer and select **Edit files** from its context menu.



For more information, see the following sections:

- "Specifying Whether Components Can Have Children Components" on page 5-19
- "Modifying a Component's Description" on page 5-19
- "Changing a Component's Display Name" on page 5-19
- "Changing a Component's Category Name" on page 5-20
- "Registering Components" on page 5-20
- "Displaying Component Help in the MATLAB Help Browser" on page 5-20

Specifying Whether Components Can Have Children Components

To specify whether a component can have children, edit `getParentable.m`. This command returns the value `true` or `false`. For example, if you no longer want your component to have child components, modify the value within the code as follows:

```
p = false;
```

Modifying a Component's Description

The description n `getDescription.m` is the same value as the **Description** field in the Report Explorer. The following example shows how to edit the `compDesc` string in this file to change a component's description to `A demonstration component`:

```
compDesc = 'A demonstration component';
```

Changing a Component's Display Name

The display name in `getName.m` is the same value as the **Display name** field in the Report Explorer. The following example shows how to edit the `compName` string in this file to change a component's display name to `Demo Component`:

```
compName = 'Demo Component';
```

Changing a Component's Category Name

The category name in getType.m is the same value as the **Category name** field in the Report Explorer. The following example shows how to edit the compCategory string in this file to change a component's category name to Custom Components:

```
compCategory = 'Custom Components';
```

Registering Components

You can register components in the Report Explorer using rptcomps2.xml. This file also helps build the list of available components.

The content of this file must be consistent with the values in the getName.m and getType.m files. If you have changed values in either of these files, you must also change their values in rptcomps2.xml. You must restart the MATLAB software session for the Report Explorer to display new information.

Displaying Component Help in the MATLAB Help Browser

The viewHelp.m file displays a help file for the component within the MATLAB Help browser. To display the help file, highlight the name of the component in the Report Explorer and click **Help**.

Examples

In this section...

Note These examples require the Datafeed Toolbox™ software.

Fetching Securities Data and Displaying It in a Table

This example shows how to create a component that fetches securities data and displays it in a report as a table.

1. Create a component named `Equity Values` in the class folder named `CStockTicker`.

2. Give the component one string property named `Ticker`, and specify its attributes.

   a. In the Options pane of the Report Explorer, double-click **New_String_Property**.

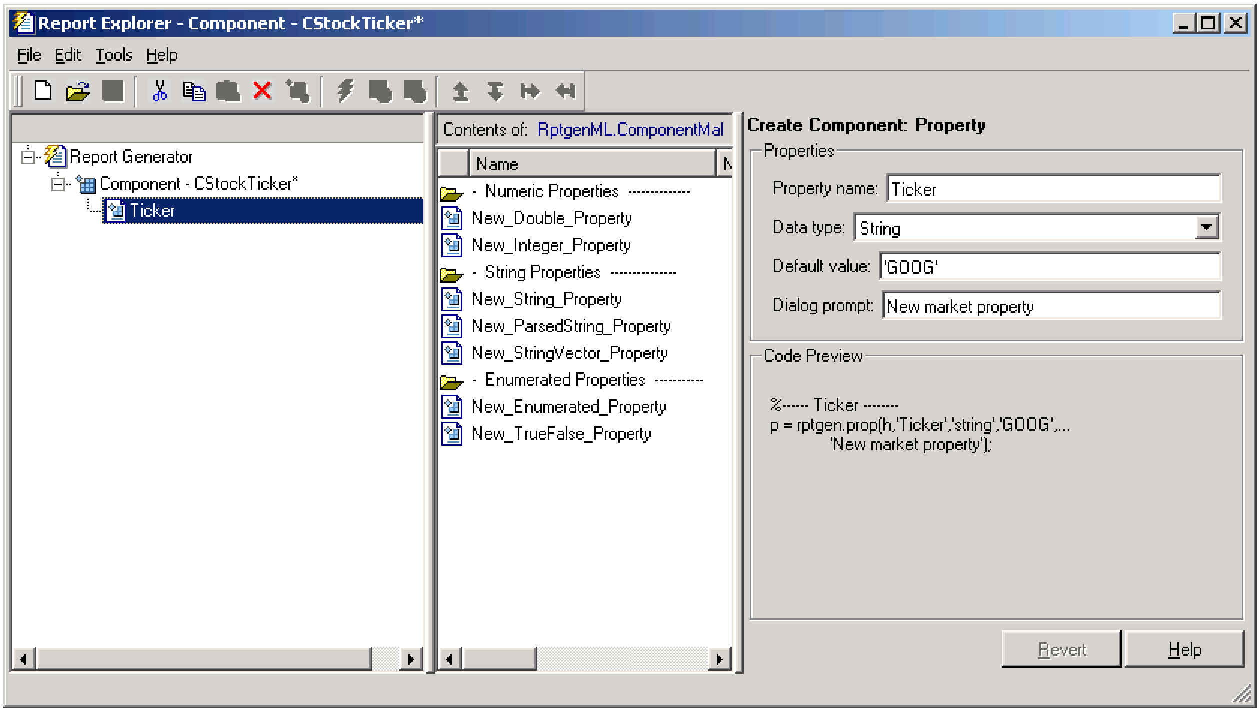
   b. For **Property name**, specify a valid MATLAB variable name.

   c. Specify the property's data type. In this case, `Ticker` is a string value, which is the default data type.

   d. Specify the property's default value.

      Because this example fetches ticker values for the security `Google`, set the **Default value** to `'GOOG'`. (The single quotation marks are required to specify a string value for this field.)

      Your specified settings appear in the **Code Preview** pane.


Report Explorer - Component - CStockTicker*
File Edit Tools Help
Report Generator
Component - CStockTicker*
Ticker
Contents of: RptgenML.ComponentMal
Name
- Numeric Properties ------------
New_Double_Property
New_Integer_Property
- String Properties -------------
New_String_Property
New_ParsedString_Property
New_StringVector_Property
- Enumerated Properties ----------
New_Enumerated_Property
New_TrueFalse_Property
Revert
Help

Create Component: Property

Properties

Property name: Ticker

Data type: String

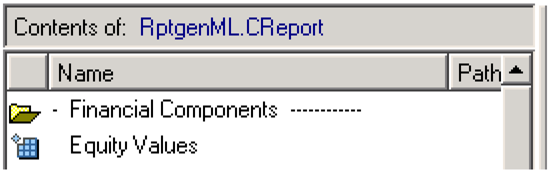
Default value: 'GOOG'

Dialog prompt: New market property

Code Preview

```
%------ Ticker -------
p = rptgen.prop(h,'Ticker','string','GOOG',...
          'New market property');
```

3 To build the new component, click the **Build** button in the Report Explorer. The Equity Values component now appears in the Options pane in the middle of the Report Explorer.


Contents of: RptgenML.CReport
Name
Path
- Financial Components ------------
Equity Values

4 Edit the component's `execute.m` file to retrieve stock market data and display it in a table in the generated report.

a In the `@CStockTicker` folder, open `execute.m`.

b Enter the following text into `execute.m`.

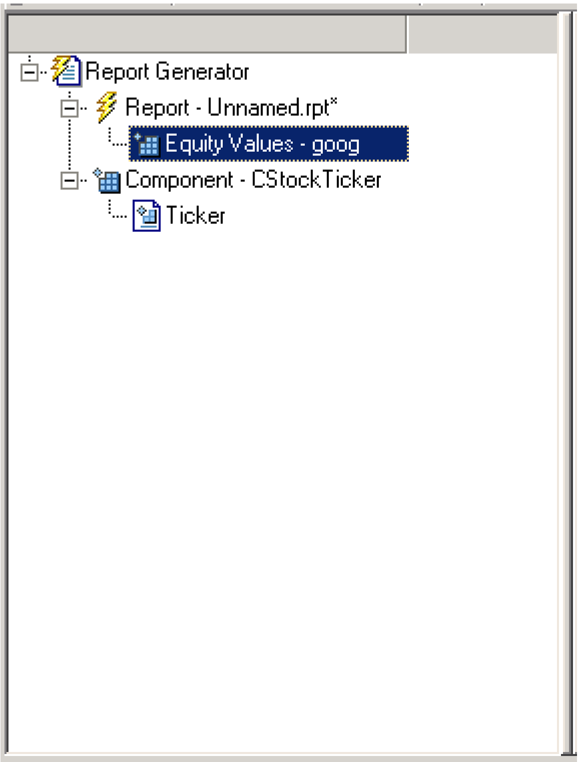
```
function out=execute(thisComp,parentDoc,varargin)
```

```
stockQuote = fetch(GOOG, thisComp.Ticker);
stockQuote.Date = datestr(stockQuote.Date,1);
stockQuote.Time = datestr(stockQuote.Time,13);
out = execute(rptgen.cfr_table(...
     'Source', stockQuote,...
     'numHeaderRows', 0,...
     'TableTitle', 'Stock Market Pricing Data'),...
    parentDoc);
```

5 Append the security symbol, `goog`, to the component name. Enter the following text into `getOutlineString.m`.

```
function olstring=getOutlineString(thisComp)

olstring = [getName(thisComp),' - ',thisComp.Ticker];
```

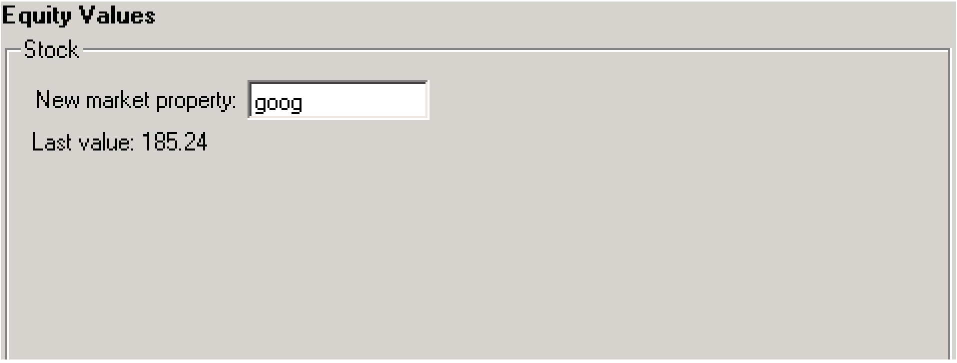
The component name now appears as `Equity Values    goog`.


Report Generator
Report - Unnamed.rpt*
Equity Values - goog
Component - CStockTicker
Ticker

6 Modify the `getdialog boxeschema.m` file to change the appearance of the Properties pane. Enter the following text into this file to display the last quoted price for the security in the Properties pane.

```
function dlgStruct = getdialog boxeschema(thisComp, name)

try
    currQuote = fetch(yahoo, thisComp.Ticker);
    quoteStr = sprintf('Last value: %g', currQuote.Last);
catch
    quoteStr = sprintf('Warning: ...
    "%s" is not a valid symbol.', thisComp.Ticker);
end

dlgStruct = thisComp.dlgMain(name,...
thisComp.dlgContainer({
  thisComp.dlgWidget('Ticker',...
    'DialogRefresh',true,...
    'RowSpan',[1 1],'ColSpan',[1 1]);
  thisComp.dlgText(quoteStr,...
    'RowSpan',[2 2],'ColSpan',[1 1]);
},'Stock',...
  'LayoutGrid',[3 2],...
  'RowStretch',[0 0 1],...
  'ColStretch',[0 1]));
```

The Properties pane for the component, `Equity Values`, now looks as follows.



Equity Values

Stock

New market property: goog

Last value: 185.24

7 Click **File > Report** to generate the report. The following output appears in the report.

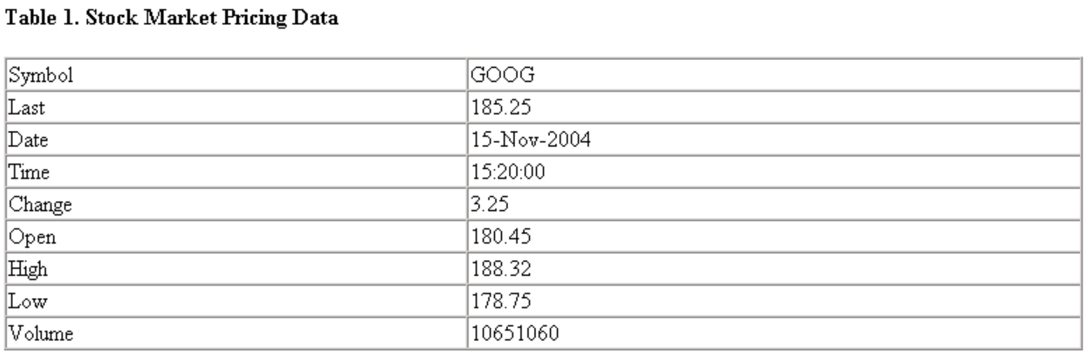


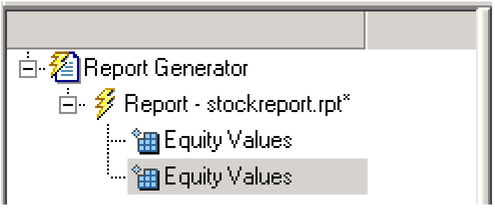
Table 1. Stock Market Pricing Data

Symbol	GOOG
Last	185.25
Date	15-Nov-2004
Time	15:20:00
Change	3.25
Open	180.45
High	188.32
Low	178.75
Volume	10651060

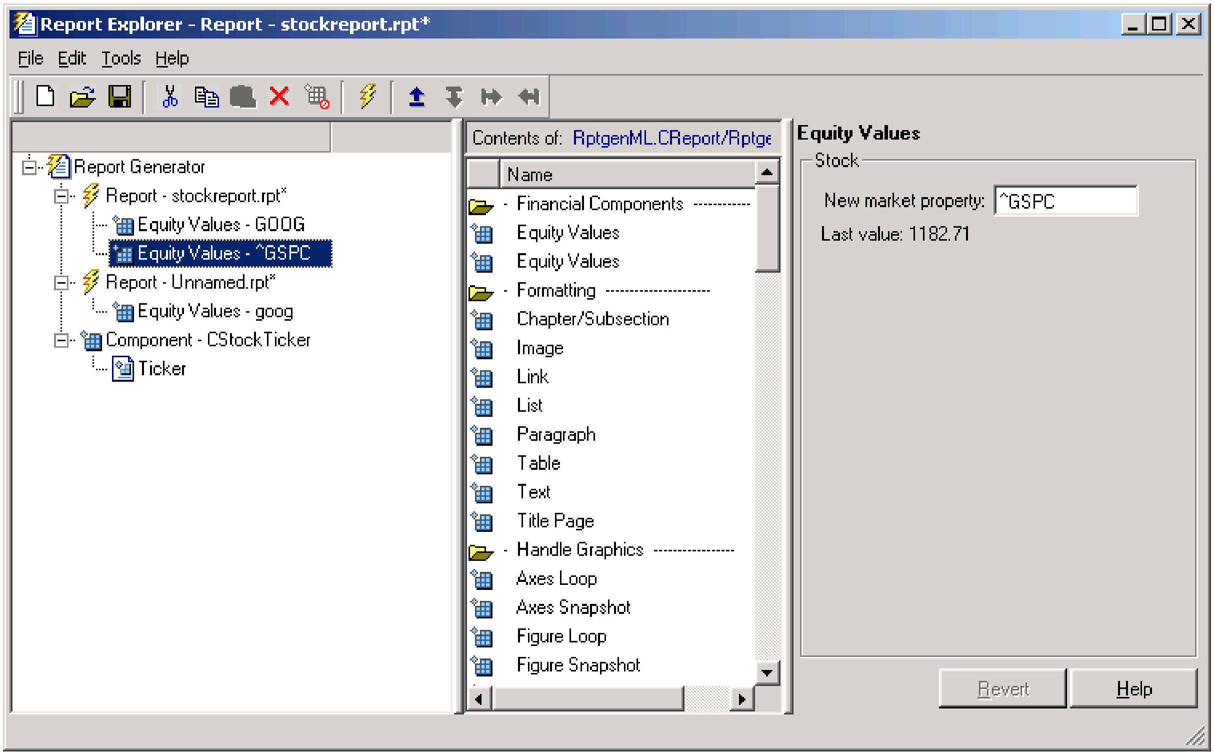
Displaying Securities Data in Two Tables

This example, which shows how to use multiple properties within a component, expands upon "Fetching Securities Data and Displaying It in a Table" on page 5-21.

1 Create a report template and save it as `stockreport.rpt`. Add two `Equity Values` components to the template.



2 Edit the entry in the **New marker property** field to change the `ticker` property of the second component to `'^GSPC'` (S&P 500 index).



New market property: ^GSPC

Last value: 1182.71

3 Run the report.

The report displays two tables of data, one for Google™ and another for the S&P 500 index.

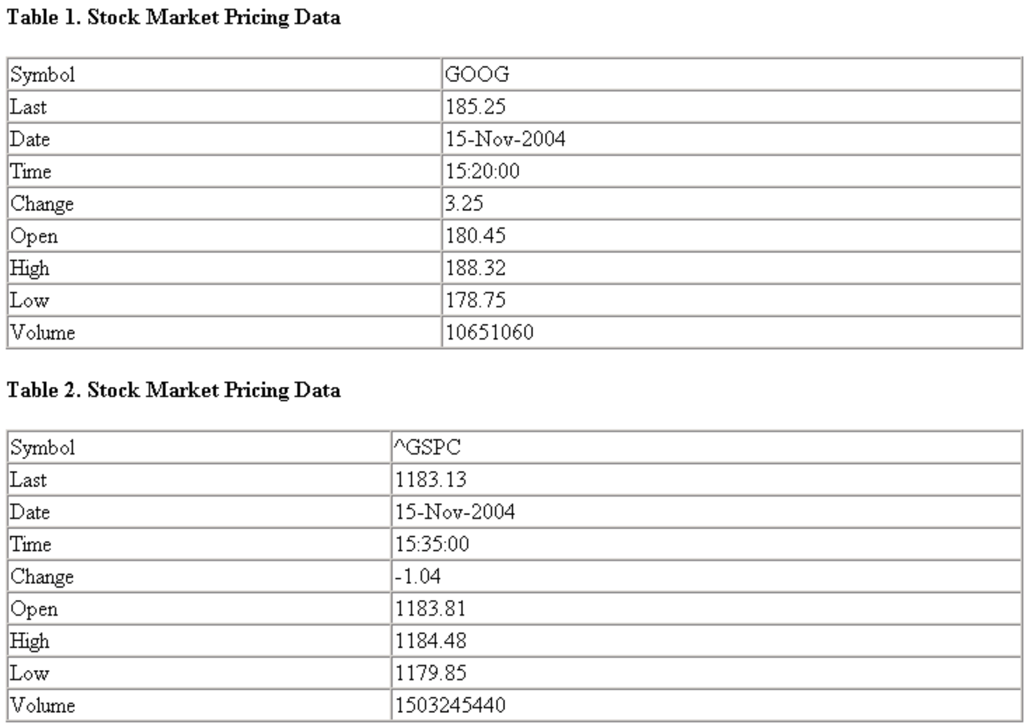


Table 1. Stock Market Pricing Data

Symbol	GOOG
Last	185.25
Date	15-Nov-2004
Time	15:20:00
Change	3.25
Open	180.45
High	188.32
Low	178.75
Volume	10651060

Table 2. Stock Market Pricing Data

Symbol	^GSPC
Last	1183.13
Date	15-Nov-2004
Time	15:35:00
Change	-1.04
Open	1183.81
High	1184.48
Low	1179.85
Volume	1503245440

Creating Custom Stylesheets

About Stylesheets

In this section...
"Built-In Versus Custom Stylesheets" on page 6-2
"Customizing Stylesheets Using Data Items" on page 6-3

Built-In Versus Custom Stylesheets

Stylesheets specify formatting and display settings for reports. The report-generation process uses stylesheets to convert reports from DocBook XML format to a format that you specify. If you want to generate the given report in a different format than initially specified, you can convert the XML document using a different or modified stylesheet.

The following table lists report output formats and their default stylesheets.

Report Format	Default Stylesheet
HTML	Uses stylesheets for either single- or multiple-page documents
PDF	Formatting Object (FO) stylesheet
RTF, Word	Document Style Semantics and Specification Language (DSSSL) stylesheet

The following table shows a list of properties for the built-in stylesheets.

Properties of Stylesheets

Name	Description
Description	A description of the stylesheet.
Display name	The stylesheet name that appears in the Options pane.
Transform type	The process used to generate reports that use a specified stylesheet. Supported types are: • HTML • FO (Formatting Object) for PDF reports • DSSSL (Document Style Semantics and Specification Language) for RTF and Word reports **Note** This field is not editable.

In most cases, the stylesheets provided with the MATLAB Report Generator software should be more than adequate for your needs. However, you may want to modify the built-in stylesheets to meet special requirements. For example, suppose one of the built-in stylesheets meets your requirements, but you want to change the page orientation. You can create a custom stylesheet by editing the built-in stylesheet to your specifications.

Customizing Stylesheets Using Data Items

Each built-in stylesheet includes editable styles, also called *data items*, organized in categories. These data items specify styles that the file converter uses for a given report. You can edit these data items to customize stylesheets for your reports.

Data items can be of different types, some of which require different editing methods. For more information about editing data items, see “Editing Stylesheet Data Items” on page 6-10.

Tip See the **Help** area at the bottom of the Properties pane on the right for a description of a specific data item that you are editing.

Working with Stylesheets

In this section...

Using the Report Explorer to Edit Stylesheets

To edit a stylesheet:

1. In Report Explorer, select a report template in the Outline pane on the left.

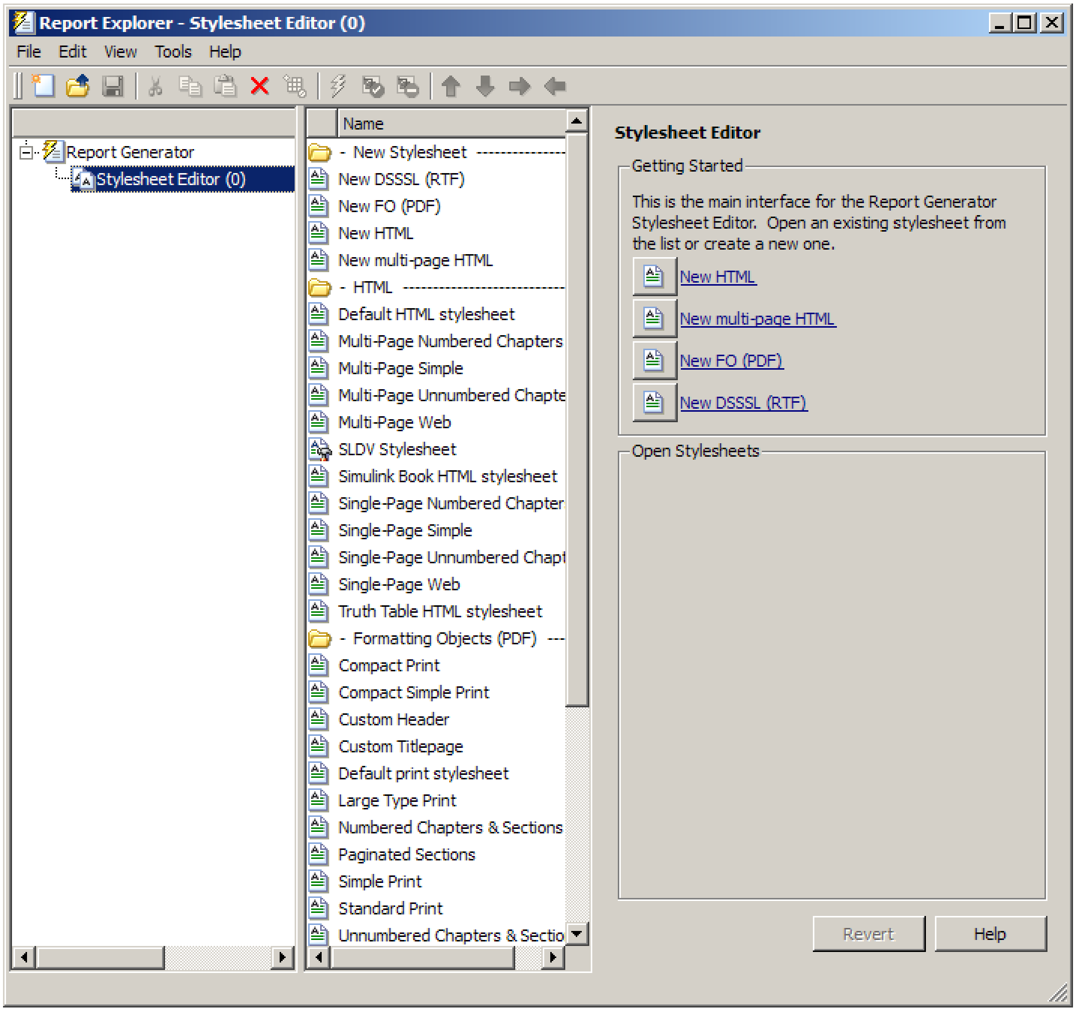
2. From the menu bar, click **Tools > Edit Stylesheet**.

The Report Explorer displays as follows.

- The Outline pane on the left displays the structure of stylesheets you create.
- The Options pane in the middle lists stylesheets available for customizing.

Tip Double-click a category to collapse it. Double-click it again to expand it.

- The Properties pane on the right shows properties of stylesheets, such as name and description.



This is the main interface for the Report Generator Stylesheet Editor. Open an existing stylesheet from the list or create a new one.

New HTML
New multi-page HTML
New FO (PDF)
New DSSSL (RTF)

- New Stylesheet ---------------
New DSSSL (RTF)
New FO (PDF)
New HTML
New multi-page HTML
- HTML ---------------------------
Default HTML stylesheet
Multi-Page Numbered Chapters
Multi-Page Simple
Multi-Page Unnumbered Chapte
Multi-Page Web
SLDV Stylesheet
Simulink Book HTML stylesheet
Single-Page Numbered Chapter
Single-Page Simple
Single-Page Unnumbered Chapt
Single-Page Web
Truth Table HTML stylesheet
- Formatting Objects (PDF) ---
Compact Print
Compact Simple Print
Custom Header
Custom Titlepage
Default print stylesheet
Large Type Print
Numbered Chapters & Sections
Paginated Sections
Simple Print
Standard Print
Unnumbered Chapters & Sectio

You can use the Report Explorer to work with stylesheets as follows.

Task	Pane to Use	Action
Create a stylesheet	Properties	Click the link that corresponds to the kind of stylesheet you want to create
Open an existing stylesheet	Properties	Click the name of the stylesheet, which appear in the **Open Stylesheets** area
Select a stylesheet to use for converting an XML source file	Options	Select a stylesheet by clicking on it
View a list of customized styles in a stylesheet	Outline	Expand any open stylesheet
View a list of styles in a stylesheet	Outline or Options	Double-click the stylesheet
View a list of stylesheets available for editing in a given category	Options	Double-click the folder that corresponds to the kind of output you want (that is, HTML, PDF, RTF, or Word)
View open stylesheets	Outline	Expand the Stylesheet Editor item in Report Explorer
Change the name or description of the current stylesheet	Properties	Edit the text in the **Display Name** or **Description** field.
Convert an XML source file using the current stylesheet	Properties	Click **Send to Source File Converter** in the Properties pane.
Edit customized style data	Properties	Click the style data item, which appears in the **Stylesheet Customizations** area
Open a style data item for editing or viewing	Options	Double-click the data item that you want to edit.
View a list of customized style data	Outline	Expand the stylesheet

Creating New Stylesheets

To create a stylesheet:

1 Open the Report Explorer.

2 From the menu bar, click **Tools > Edit Stylesheet**.

3 In the Properties pane on the right, choose the built-in stylesheet for the format with which you want to work. Options are:

- **New HTML.** Creates a stylesheet for HTML reports.
- **New multi-page HTML.** Creates a stylesheet for HTML reports with more than one page.
- **New FO (PDF).** Creates a stylesheet for PDF reports.
- **New DSSSL (RTF).** Creates a stylesheet for RTF reports.

The new stylesheet appears in the Outline pane on the left.

4 In the Properties pane on the right, modify the properties for the stylesheet as needed. Add data items to the new stylesheet:

a Drag the data item you want to add from the Options pane in the middle to the stylesheet in the Outline pane on the left.

b In the Properties pane on the right, edit the data items for the selected style. For more information, see “Editing Stylesheet Data Items” on page 6-10

5 Save the stylesheet. For information about how to save a stylesheet, see “Saving Stylesheets” on page 6-8.

Saving Stylesheets

You must save a stylesheet before you can use it to convert a source file or associate it with a report. To use the Report Explorer to save a stylesheet:

1 Select the stylesheet that you want to save in the Outline pane on the left.

2 Select **File > Save As** from the menu bar and specify a new name for the stylesheet (to avoid overwriting built-in stylesheets). You must save the

file in a folder in your MATLAB path for the stylesheet to appear in the Report Explorer. The file name must be unique in the MATLAB path.

By convention, MATLAB Report Generator stylesheets have `.rgs` as their file name extension.

Deleting Stylesheets

To use the Report Explorer to delete a stylesheet that you created:

1. Select the stylesheet that you want to delete in the Outline pane on the left.
2. Click the stylesheet to delete from the Options pane in the middle.
3. Click **Delete stylesheet** in the stylesheet's Properties pane on the right.

You must restart the MATLAB software session for deleted stylesheets to disappear from the Options pane.

Note You cannot delete built-in stylesheets.

Editing Stylesheet Data Items

Categories of Data Items in Built-In Stylesheets

You can edit data items in built-in stylesheets to customize them. Data items appear in *categories*, according to their function. The following tables list the categories and data items for each type of stylesheet provided with the MATLAB Report Generator software.

Categories of Styles in PDF (FO) Stylesheets

Category	Description of Data Items in Category
Automatic labeling	Options for enumeration of parts of the report, such as chapters and sections
Callouts	Options and specifications related to callouts, such as defaults, use of graphics, size, path, fonts, characters, and extensions
Cross References	Option to control whether page numbers appear in report
Font Families	Specification of defaults for body text, copyright, quotes, symbols, dingbats, monospace, sans serif, and titles
Graphics	Specification of default width and options related to scaling attributes
Lists	Specification of spacing related to lists and list items
Meta/*Info	Options related to year ranges

Categories of Styles in PDF (FO) Stylesheets (Continued)

Category	Description of Data Items in Category
Miscellaneous	Options and specifications for placement of titles, comments, variable lists, block quotations, ulinks, hyphenations of URLs, verbatim environment display, use of SVG, table footnote numbers, superscript, and subscript
Pagination and General Styles	Specifications of page orientation, margins, double-sided, paper type, hyphenation, line height, columns, master font, draft mode, watermark, blank pages, rules for headers and footers, and content of headers and footers **Note** You can specify parameters in this category, such as margin widths and header and footer height, in units of inches (`in`), millimeters (`mm`), or picas (`pi`), where 1 pica = 1/6 inch.
Property Sets	Specification and options related to figure titles, monospace properties, verbatim text, section titles, and levels of sections
Reference Pages	Option to control whether the class name is displayed
Stylesheet Extensions	Line numbering and table columns extensions
Table of Contents (TOC)/List of Tables (LOT)/Index Generation	Specifications for layout of TOC, depth of sections, indentation, and margins
Tables	Specifications for size of tables and their borders

Categories of Styles in HTML and Multi-Page HTML Stylesheets

Category of Style	Description of Data Items in Category
Automatic labeling	Options for enumeration of parts of the report, such as chapters and sections
Callouts	Options and specifications related to callouts, such as defaults, use of graphics, size, path, fonts, characters, and extensions
Chunking	Options related to using an explicit TOC for chunking, depth of section chunks, navigational graphics, and display of titles in headers and footers
Stylesheet Extensions	Line numbering, graphic size, and table columns extensions
Graphics	Specification of default width and depth, use of HTML embed for SVG, viewports, and options related to scaling attributes
HTML	Specifications related to dynamically served HTML, base and head elements, type of stylesheet, css, propagation of styles, longdesc, validation, cleanup, draft mode, watermark, and generation of abstract
Linking	Specification of Mailto URL and target for ulinks
Meta/*Info	Options related to year ranges
Miscellaneous	Options and specifications for comments, verbatim environment pixels, em space, use of SVG, and table footnote numbers
Reference Pages	Option control whether the class name is displayed
Table of Contents (TOC)/List of Tables (LOT)/Index Generation	Specifications for layout of TOC, depth of sections, indentation, and margins

Categories of Styles in HTML and Multi-Page HTML Stylesheets (Continued)

Category of Style	Description of Data Items in Category
Tables	Specifications for size of tables, table cell spacing and padding, and borders
XSLT Processing	Options related to header and footer navigation and rules

Categories of Styles in RTF (DSSSL) Stylesheets

Category of Style	Description of Data Items in Category
Admonitions	Options and path for admonition graphics
Backends	Options for Tex, MIF, and RTF back-end usage
Bibliographies	Options related to checking citations; suppressing, enumerating, and using titles of entries
Fonts	Specifications for font family and size to use for some elements
Footnotes	Options for ulinks as footnotes and page location
Graphics	Specifications for file extensions, file names, and loading library database
Indents	Specifications for hanging indents, first paragraphs, and start of blocks
Labeling	Enumeration of sections and other elements
Miscellaneous	Options for floating formal objects, punctuation for run-in heads and honorifics, bold for first use of term, minimum leading between lines, and automatic hyphenation
OLinks	Using an extension for finding outline information
Object Rules	Specifications for placement and width of rules

Categories of Styles in RTF (DSSSL) Stylesheets (Continued)

Category of Style	Description of Data Items in Category
Paper/Page Characteristics	Specifications for paper type, page numbers, width of pages, margins, and columns; heading-levels, sides; and writing mode (such as left-to-right)
Quadding	Specifications for justifying paragraphs
RefEntries and Functions	Options related to generation and display of reference entries and synopses for functions
Running heads	Options for generating and displaying running heads of chapters
Table of Contents (TOC)/List of Tables (LOT)	Options to produce or display TOC for sets, books, parts, references, articles. Options to display TOC on title page
Tables	Specification of width in simple list
Titlepages	Options to produce and display title pages for sets, books, parts, references, articles. Options for author's name and ordering elements
VariableLists	Options and specifications for term length and formatting
Verbatim Environments	Specifications for width, enumeration, size, indentation, line frequency, and callouts
Vertical Spacing	Specifications for space between lines and paragraphs

Editing Data Items in Simple Versus Advanced Edit Mode

- To edit a data item in *simple edit mode*, edit a simple string that corresponds to the data in the stylesheet. This string appears in the field to the right of the **Value** label. For some values, use a selection list to change the value instead of typing in text.

- To edit a data item in *advanced edit mode*, edit the XML code directly.

Note This section gives instructions for simple edit mode, except where explicitly specified otherwise.

The user interface is in simple edit mode when the data item appears in a pane labeled **Value**. It is in advanced edit mode when the data item appears in a pane labeled **Value (XML)**. To switch from simple to advanced edit mode, click **Edit as XML**.

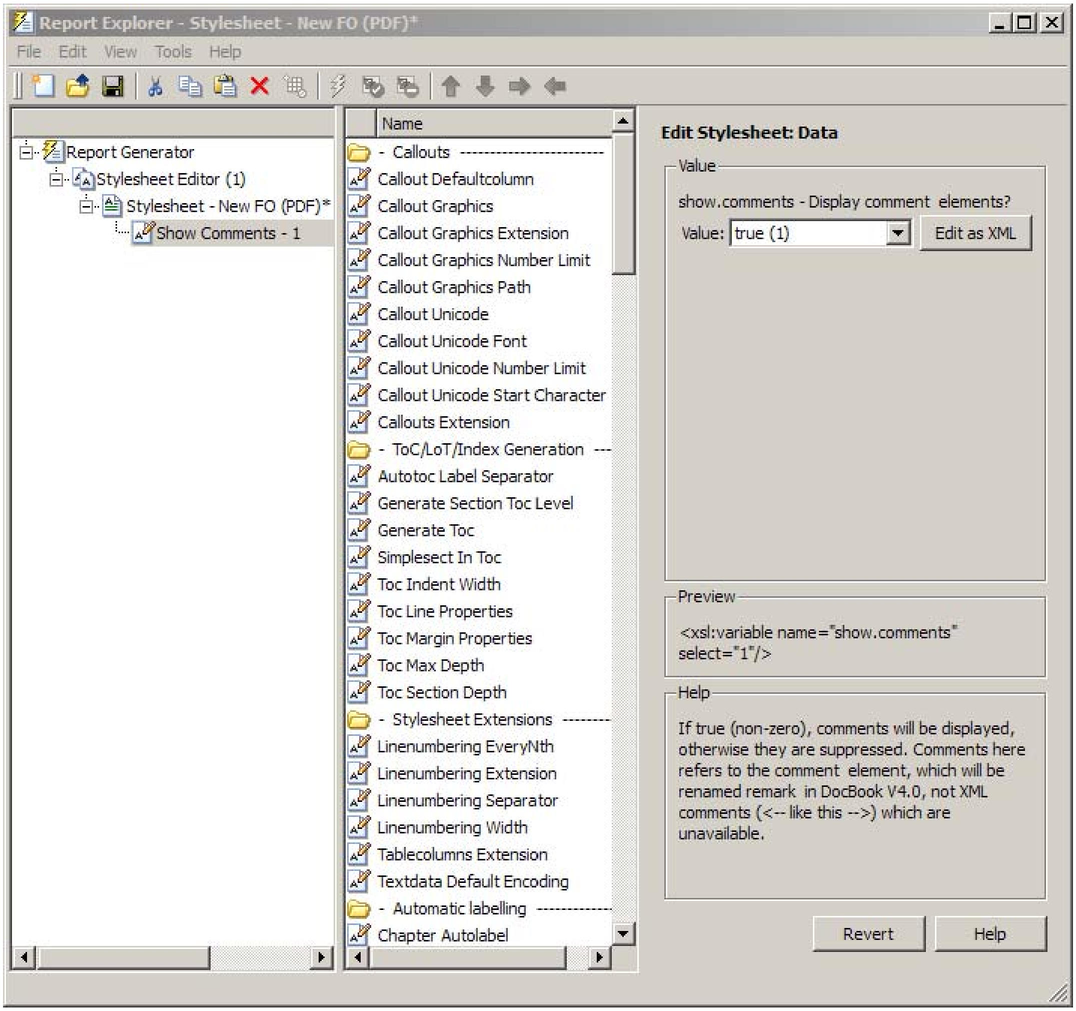
Edit values for most data items in PDF and HTML stylesheets in either simple edit mode or advanced edit mode. Edit values for RTF stylesheets in simple edit mode only. Data items in RTF stylesheets do not support advanced edit mode.

Note To modify content for headers and footers you edit *stylesheet cells*, which do not appear in either simple or advanced mode. For more information, see "Using Stylesheet Cells to Manage PDF Report Header and Footer Content" on page 6-22.

Working with Data Items

Select a stylesheet from the Options pane in the middle of the Report Explorer. The Outline pane on the left shows the name of the current style data item inside its stylesheet. The Options pane in the middle shows a list of available stylesheet data items. The Properties pane on the right displays **Stylesheet Editor: Data**. It also includes the following information:

- The value of the data item is in a pane labeled **Value** in simple edit mode or **Value (XML)** in advanced edit mode.
- To the right of the value is the **Edit as XML** toggle button.
- The **Preview** pane includes a partial view of the stylesheet that specifies the data item. The data in this pane is not editable.
- The **Help** pane contains information about the data item. This information is not editable.



Report Generator
Stylesheet Editor (1)
Stylesheet - New FO (PDF)*
Show Comments - 1

- Callouts ------------------------
- Callout Defaultcolumn
- Callout Graphics
- Callout Graphics Extension
- Callout Graphics Number Limit
- Callout Graphics Path
- Callout Unicode
- Callout Unicode Font
- Callout Unicode Number Limit
- Callout Unicode Start Character
- Callouts Extension
- ToC/LoT/Index Generation ---
- Autotoc Label Separator
- Generate Section Toc Level
- Generate Toc
- Simplesect In Toc
- Toc Indent Width
- Toc Line Properties
- Toc Margin Properties
- Toc Max Depth
- Toc Section Depth
- Stylesheet Extensions --------
- Linenumbering EveryNth
- Linenumbering Extension
- Linenumbering Separator
- Linenumbering Width
- Tablecolumns Extension
- Textdata Default Encoding
- Automatic labelling ------------
- Chapter Autolabel

Edit Stylesheet: Data

Value

show.comments - Display comment elements?

Value: true (1)

Preview

```
<xsl:variable name="show.comments"
select="1"/>
```

Help

If true (non-zero), comments will be displayed, otherwise they are suppressed. Comments here refers to the comment element, which will be renamed remark in DocBook V4.0, not XML comments (<-- like this -->) which are unavailable.

Editing Boolean and Enumerated Values

In the previous figure, theShow Comments data item is of type Boolean. Its current value is true(1). Change this value using the menu list for the value field. In this case, the only other possible value is false(2).

Editing Strings

For the values of some data items, the Report Explorer displays text in the editable **Value** field. You can specify an XML expression, though you are not required to do so.

Editing XML Expressions

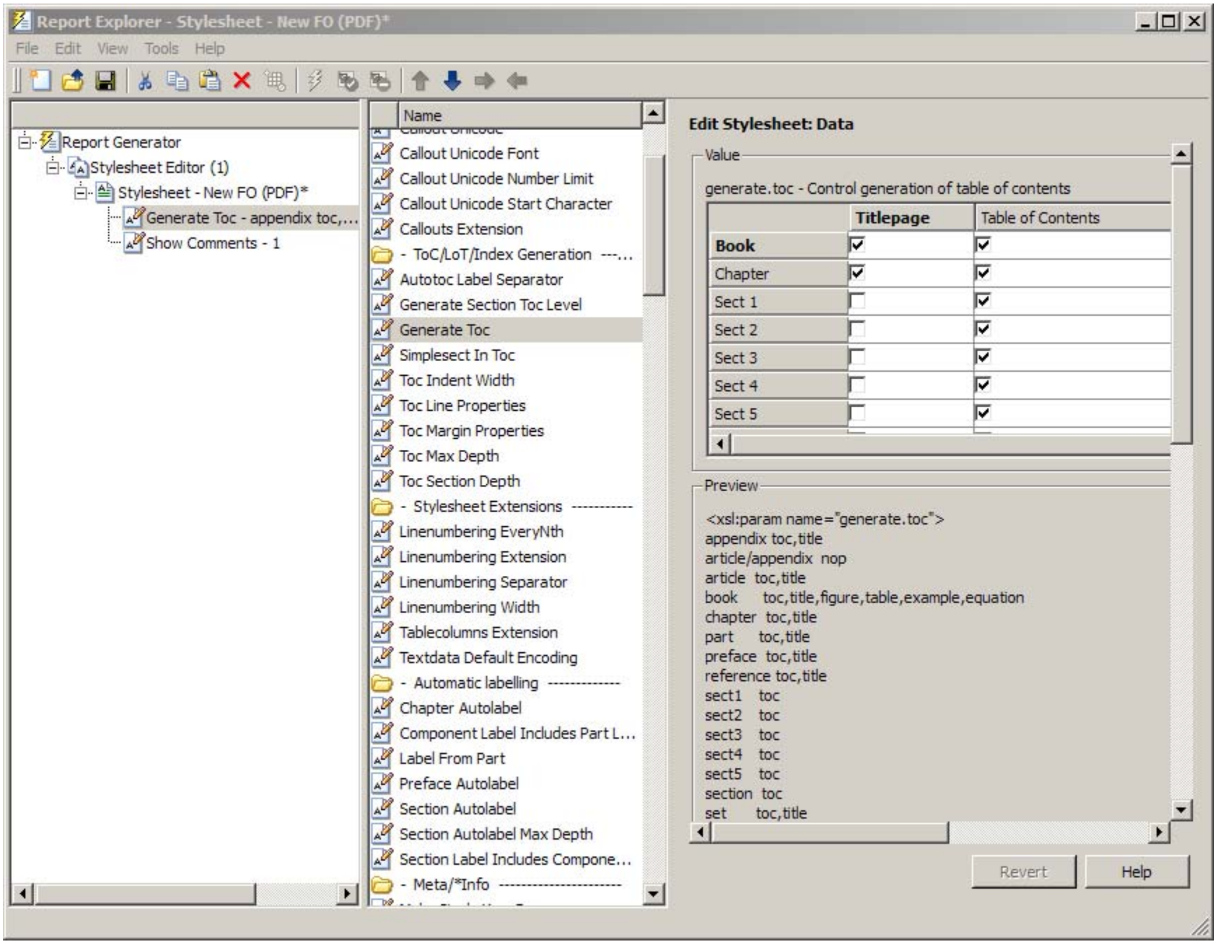
To make complex changes to a stylesheet, consider using Advanced edit mode. This enables you to edit XML expressions directly in the **Value (XML)** pane. If this pane does not appear, click **Edit as XML** to switch to advanced edit mode.

Make sure that you enter valid XML. Invalid XML values generate an error, which appears at the top of the Properties pane.

Modifying TOC Properties

To change values for generation of the report's table of contents (TOC), select the appropriate values from a matrix of check boxes.

The following figure shows the values for the **Generate Toc** data item on the **PDF** stylesheet. Select the check boxes to control the values that appear in the report's title page and table of contents.



Edit Stylesheet: Data

Value

generate.toc - Control generation of table of contents

	Titlepage	Table of Contents
Book	☑	☑
Chapter	☑	☑
Sect 1	□	☑
Sect 2	□	☑
Sect 3	□	☑
Sect 4	□	☑
Sect 5	□	☑

Preview

```
<xsl:param name="generate.toc">
appendix toc,title
article/appendix  nop
article  toc,title
book     toc,title,figure,table,example,equation
chapter  toc,title
part     toc,title
preface  toc,title
reference toc,title
sect1    toc
sect2    toc
sect3    toc
sect4    toc
sect5    toc
section  toc
set      toc,title
```

Modifying Title Placement Properties

The **Title Placement** data items, which are in the **Miscellaneous** category, control the position of titles for figures and tables.

Selecting one of these data items for editing causes the Properties pane on the right to display possible values in a menu list. Specify whether you want the title to appear before or after a given figure or table.

Modifying Attributes

An *attribute* is a data item that specifies information for an XML element. An attribute must be a child of an *attribute set*. For more information, see "Editing Attribute Sets" on page 6-19.

Note The information in the **Help** area of the Properties pane of an attribute describes the set to which the attribute belongs.

Editing Attribute Sets

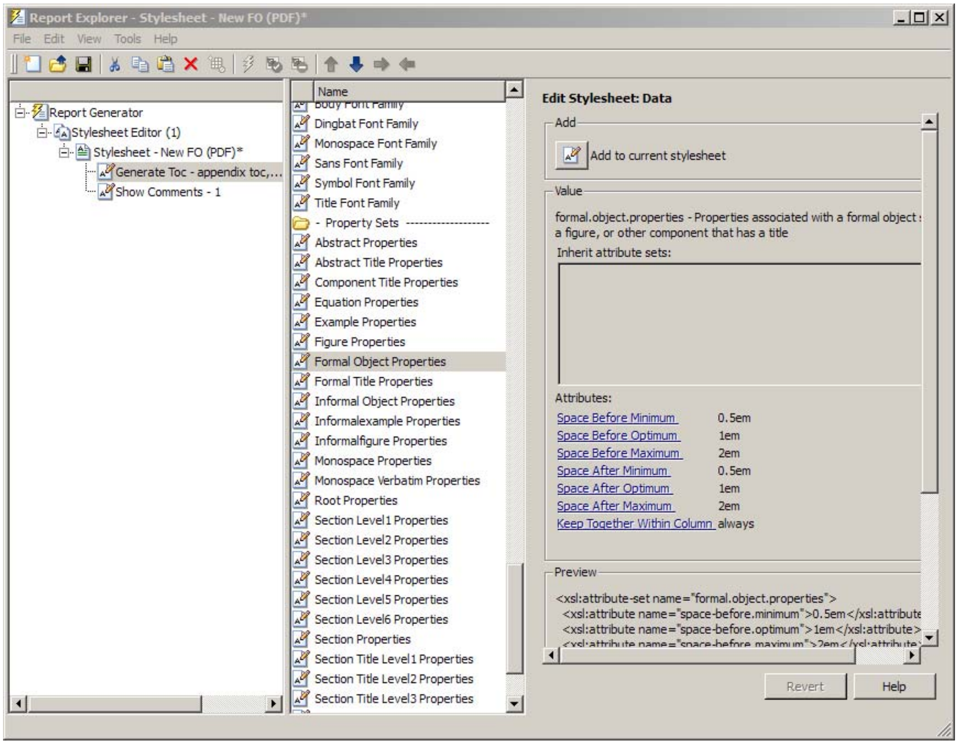
An *attribute set* consists of a group of attributes. Selecting an attribute set in the Outline pane on the left causes the Properties pane to list the attributes that belong to that set.

To edit a specific attribute, expand the attribute set in the Outline pane and select the attribute you want to edit.

To edit the attribute set, type `text` in the **Inherit attribute sets** area of the Properties pane.

An example of an attribute set is `Formal Object Properties`, a data item in the `Property Sets` category of the default print stylesheet for PDF documents.

Here is an example of the Report Explorer showing the `Formal Object Properties` attribute set in the Properties pane.


Report Explorer - Stylesheet - New FO (PDF)*
File Edit View Tools Help
Report Generator
Stylesheet Editor (1)
Stylesheet - New FO (PDF)*
Generate Toc - appendix toc,...
Show Comments - 1
Name
Body Font Family
Dingbat Font Family
Monospace Font Family
Sans Font Family
Symbol Font Family
Title Font Family
- Property Sets ------------------
Abstract Properties
Abstract Title Properties
Component Title Properties
Equation Properties
Example Properties
Figure Properties
Formal Object Properties
Formal Title Properties
Informal Object Properties
Informalexample Properties
Informalfigure Properties
Monospace Properties
Monospace Verbatim Properties
Root Properties
Section Level1 Properties
Section Level2 Properties
Section Level3 Properties
Section Level4 Properties
Section Level5 Properties
Section Level6 Properties
Section Properties
Section Title Level1 Properties
Section Title Level2 Properties
Section Title Level3 Properties
Edit Stylesheet: Data
Add
Add to current stylesheet
Value
Revert
Help

formal.object.properties - Properties associated with a formal object :
a figure, or other component that has a title

Inherit attribute sets:

Attributes:

Space Before Minimum	0.5em
Space Before Optimum	1em
Space Before Maximum	2em
Space After Minimum	0.5em
Space After Optimum	1em
Space After Maximum	2em
Keep Together Within Column	always

Preview

```
<xsl:attribute-set name="formal.object.properties">
  <xsl:attribute name="space-before.minimum">0.5em</xsl:attribute
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">2em</xsl:attribute>
```

Editing Varpair Values

Data items in RTF stylesheets appear as `varpair` data items, which are name/value pairs of information. RTF stylesheets are the only type of stylesheet that includes `varpair` data items.

Edit `varpair` data items as strings or as Boolean values. Boolean values appear as `true (#t)` and `false (#f)`.

Note You cannot edit RTF stylesheet data items as XML.

Note Data of type `varpair` is sometimes represented in stylesheets as DSSSL rather than XML. As a result, the code that appears in the **Preview** pane of the Properties pane on the right looks different from code associated with other kinds of MATLAB Report Generator stylesheets.

Deleting Data Items

To delete a customized data item:

1. Right-click the data item in the Outline pane on the left.
2. Select **Delete**.

Using Stylesheet Cells to Manage PDF Report Header and Footer Content

In this section...

About Stylesheet Cells and Cell Groups

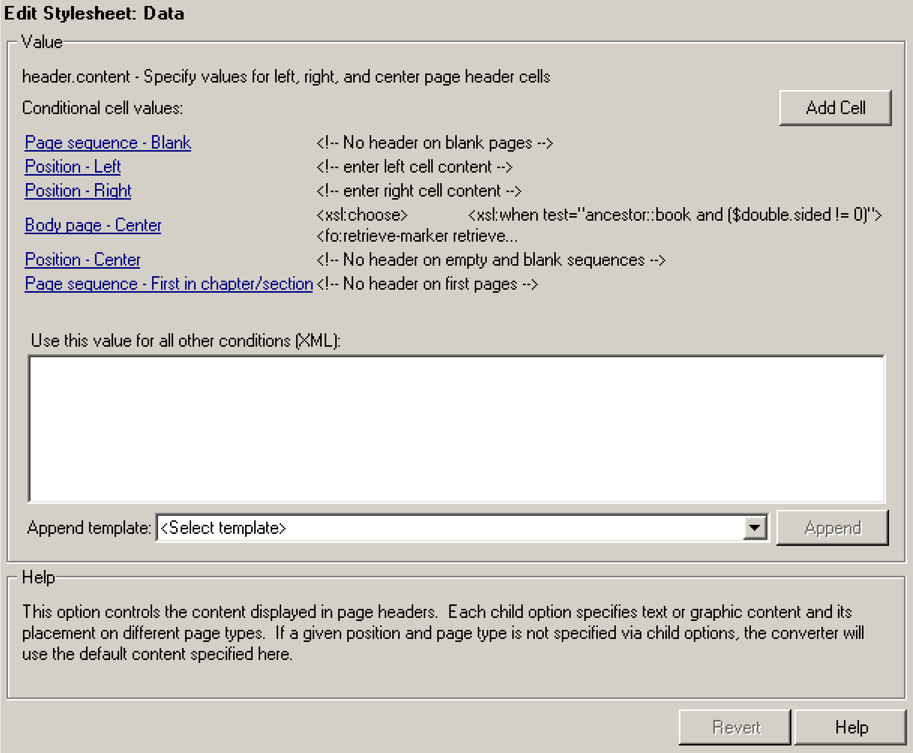
Use *stylesheet cells* to specify content of headers and footers in PDF reports.

The MATLAB Report Generator software defines a page as six *cells*. These cells correspond to the left, right, and center of the page's header and the left, right, and center of the page's footer.

A *cell group* consists of one or more stylesheet cells. Two cell groups are available for PDF reports: **Header Content** and **Footer Content**.

The Properties pane for each cell in a cell group lists the group's current stylesheet cell definitions. These definitions appear in a two-column list of **Conditional cell values**. The first column displays the name of a *condition*. The second column displays *content* that appears in the report if the specified condition is met.

For example, the stylesheet cell `Page sequence - Blank` specifies the content for a blank page; by default, the content is empty. Similarly, `Cell - Right Side` specifies the content for the right side of the header on every page.



Edit Stylesheet: Data

Value

header.content - Specify values for left, right, and center page header cells

Conditional cell values:

Page sequence - Blank	<!-- No header on blank pages -->
Position - Left	<!-- enter left cell content -->
Position - Right	<!-- enter right cell content -->
Body page - Center	<xsl:choose> <xsl:when test="ancestor::book and ($double.sided != 0)"> <fo:retrieve-marker retrieve...
Position - Center	<!-- No header on empty and blank sequences -->
Page sequence - First in chapter/section	<!-- No header on first pages -->

Use this value for all other conditions (XML):

Append template: <Select template>

Help

This option controls the content displayed in page headers. Each child option specifies text or graphic content and its placement on different page types. If a given position and page type is not specified via child options, the converter will use the default content specified here.

You can use many combinations of conditions and values to customize content of headers and footers. The MATLAB Report Generator software provides several predefined conditions that are frequently used. These predefined cells appear in the Properties panes for the Header Content and for Footer Content cell groups.

Working with Headers and Footers

Adding Content That Satisfies Specified Conditions

You can use the Properties pane of a stylesheet cell to specify content that satisfies specified conditions. The Properties pane for a stylesheet cell includes the following.

Label	Definition	Description
Condition	Condition that must be met for content to appear in the report	This is a selection list of frequently used and predefined conditions. Select a condition and click **Edit** to view or change a condition's XML code
Value (XML)	Content to appear in the report if the condition is met	Modify or create XML code for header or footer content
Append Template	Name of the template you use to add content	Templates containing XML code that you can use to add content. For more information, see "Using Templates to Add Content to Headers and Footers" on page 6-25.

When the File Converter processes a page, it evaluates settings that are relevant to each of the six cells on the page and adds content accordingly. If there are no conditions in effect for a given cell, the File Converter uses the default values for the cell group.

Possible conditions and their values as coded in XML are shown in the following table.

Name of Condition	Possible Values for the Condition	Sample XML Code

`$position`	`right`
	`center`
	`left`
		`$position='right'`
		`$position='center'`
		`$position='left'`

`$sequence`	`odd`
	`even`
	`first`
	`blank`
		`$sequence=odd`
		`$sequence=even`
		`$sequence=first`
		`$sequence=blank`

`$double-sided`	`0`
	`1`
		`$double-sided=0`
		`$double-sided=1`

`$pageclass`	`$titlepage`
	`$lot`
	`$body`
		`$pageclass=$titlepage`
		`$pageclass=$lot`
		`$pageclass=$body`

Use standard logical operators (such as `=` , `!=` , `and`, `or`) and nested expressions (characters between parentheses are an expression within an expression) to specify *complex conditions*. You can use complex conditions to set the position of headers and footers on pages. You can also use them to specify other settings, such as where in the report the content appears.

Using Templates to Add Content to Headers and Footers

Templates are available for adding the following items to headers and footers:

- Text
- Author names
- Page numbers
- Titles for chapters and sections

- Chapter numbering
- Draft information
- Comments
- Graphics

Templates used by the File Converter are Extensible Style Language Transformations (XSLT), which is a language for transforming XML documents into other XML documents. For details about XSLT, see the Web site for the World Wide Web Consortium (W3C®) at `http://www.w3.org/TR/xslt`.

To use a template to specify content for a header or footer:

1. In the **Append template** list, select the type of content you want to add.
2. Click **Append**.

   The Properties pane on the right displays default content for the type you select. Edit the XML code to change the default content.

For example, to specify `text` as the content:

1. Select **Text** from the **Append template** list.
2. Click **Append**.
3. The default value for `xsl:text` is `Confidential`. Edit the value as needed.

Inserting Graphics Files

To add a graphics file to headers or footers in a report, you must:

1. Specify the name of the file in the **Header Content** or **Footer Content** stylesheet cell.
2. Edit the values of the `Region Before Extent` and `Region After Extent` data items. These are located in the **Pagination and General Styles** folder of the **Options** pane for PDF formatting.

For an example of adding a graphic file to a header, see "Adding Graphics to Headers in PDF Reports" on page 6-29.

Note PDF reports support bitmap and gif images (`.bmp` and `.gif` files) in headers and footers.

Modifying Fonts and Other Properties

You cannot use stylesheet cells to modify the font family or other such properties of headers and footers. To specify the style of the content in headers and footers, use the **Header Content Properties** and **Footer Content Properties** attribute sets.

Each of these attribute sets is a pagination style data item for PDF stylesheets. You can edit a particular attribute in the set by selecting it in the Outline pane on the left.

For an example of modifying font size and other properties of a PDF report, see "Changing Font Size, Page Orientation, and Paper Type of a Generated Report" on page 6-34.

Examples

In this section...

Numbering Pages in a Report

This example shows how to edit a stylesheet cell to number the upper-right side of all pages in the generated report.

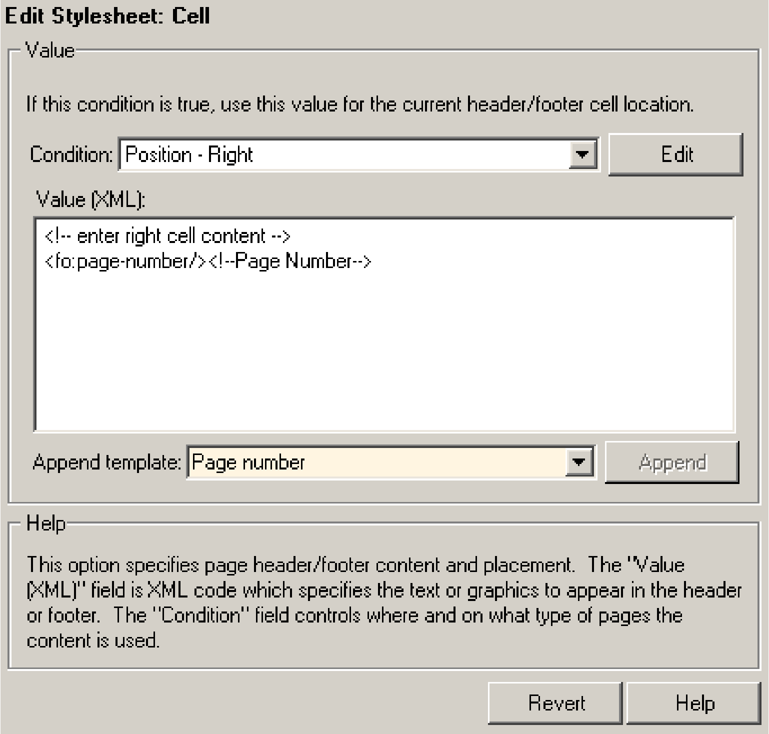
1 Define a basic stylesheet cell in the Header Content cell group with a condition of `right`.

   a Open a PDF stylesheet in the Report Explorer.

   b Double-click **Header Content** (under **Pagination and General Styles**) in the Options pane in the middle.

   c Click **Position - Right** in the Properties pane on the right.

2 Set the header content to the current page number by selecting `Page number` from the **Append template** selection list.



Edit Stylesheet: Cell

Value

If this condition is true, use this value for the current header/footer cell location.

Condition: Position - Right

Value (XML):

```
<!-- enter right cell content -->
<fo:page-number/><!--Page Number-->
```

Append template: Page number

Help

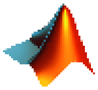
This option specifies page header/footer content and placement. The "Value (XML)" field is XML code which specifies the text or graphics to appear in the header or footer. The "Condition" field controls where and on what type of pages the content is used.

3 Click **Append**.

Adding Graphics to Headers in PDF Reports

This example shows how to include an image in the center of the header of each page in a PDF report, excluding the report's title page and the first page of each chapter. You do this by editing default header content for a PDF stylesheet. This example uses the report template `mfile-report.rpt`.

You can use any bitmap or jpeg file as image content. You must know the size of the image so that you can allow enough room for it in the header. This example uses the `sample_logo.bmp` image, which is shown here.



Note PDF reports support only bitmap (`.bmp`) and jpeg (`.jpg`) images.

To include this image file in the center of each header in the body of a PDF report:

1 Open `mfile-report.rpt` by entering the following at the MATLAB command prompt:

```
setedit mfile-report
```

2 Create a custom stylesheet.

   a Select **Tools > Edit Stylesheet** in the menu bar of the Report Explorer.

   b Click **New FO (PDF)** in the Properties pane on the right.

   c As the **Display name**, enter `Logo stylesheet for PDF`.

   d As **Description**, enter `Company logo in center of header`.

   e Save the stylesheet as `logo_stylesheet.rgs` in a folder on your MATLAB path.

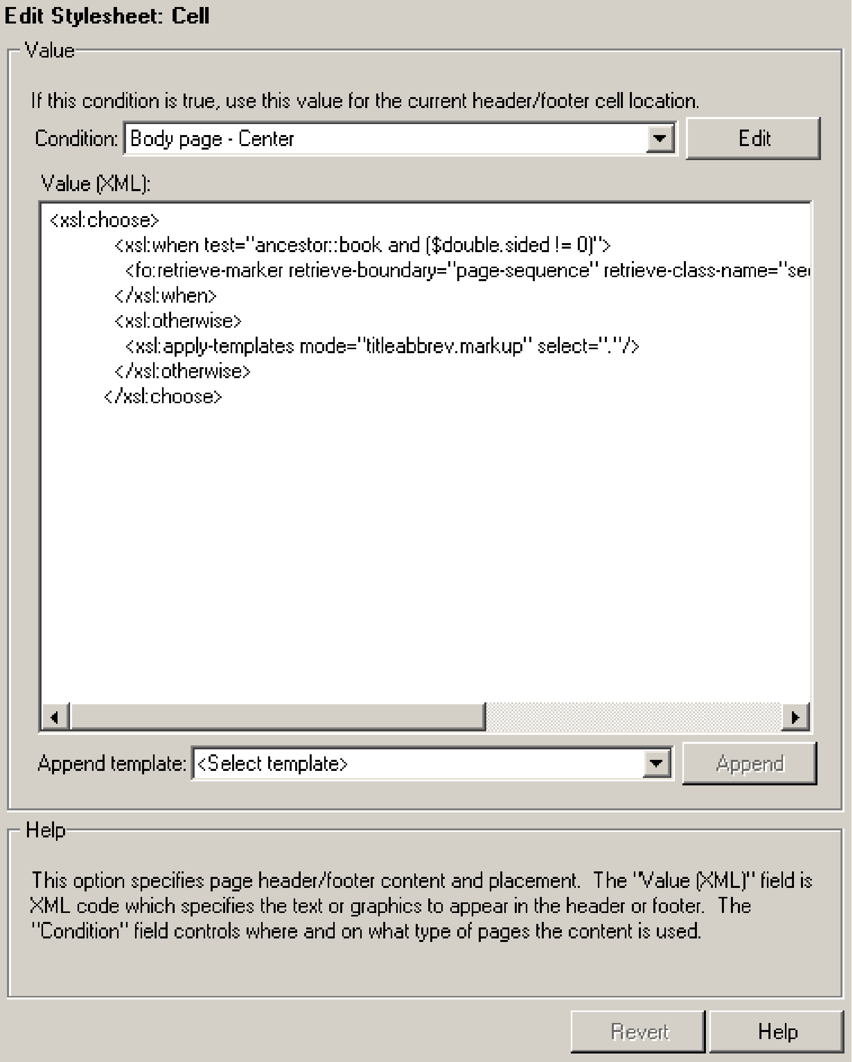
3 Open the cell group for editing.

   a Scroll through the Options pane on the left to the **Pagination and General Styles** folder.

   b Double-click **Header Content** in the Options pane.

   c Click **Body page – Center** from the list of cells in the Properties pane on the right.

   The Properties pane appears as shown.



Edit Stylesheet: Cell

Value

If this condition is true, use this value for the current header/footer cell location.

Condition: Body page - Center

Value (XML):

```
<xsl:choose>
          <xsl:when test="ancestor::book and ($double.sided != 0)">
            <fo:retrieve-marker retrieve-boundary="page-sequence" retrieve-class-name="se
          </xsl:when>
          <xsl:otherwise>
            <xsl:apply-templates mode="titleabbrev.markup" select="."/>
          </xsl:otherwise>
        </xsl:choose>
```

Append template: <Select template>

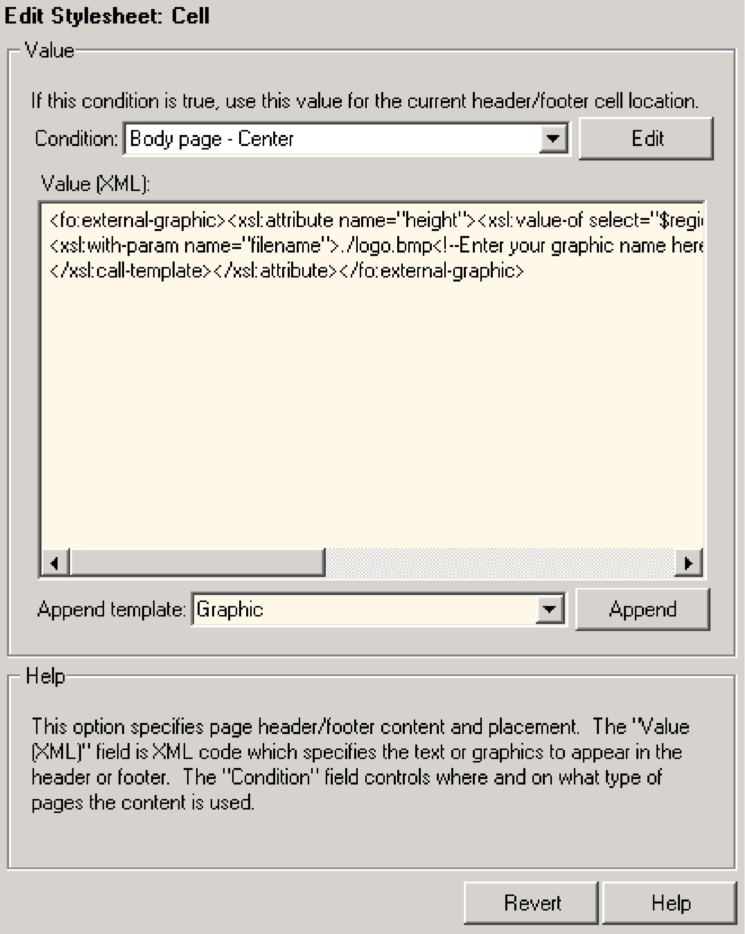
Help

This option specifies page header/footer content and placement. The "Value (XML)" field is XML code which specifies the text or graphics to appear in the header or footer. The "Condition" field controls where and on what type of pages the content is used.

d Delete the text in the **Value (XML)** field.

e Select **Graphic** from the **Append template** selection list and click **Append**.

The Properties pane on the right shows the XML code that tells the File Converter to include the graphic.



Edit Stylesheet: Cell

Value

If this condition is true, use this value for the current header/footer cell location.

Condition: Body page - Center

Value (XML):

```
<fo:external-graphic><xsl:attribute name="height"><xsl:value-of select="$regi
<xsl:with-param name="filename">./logo.bmp<!--Enter your graphic name here
</xsl:call-template></xsl:attribute></fo:external-graphic>
```

Append template: Graphic

Help

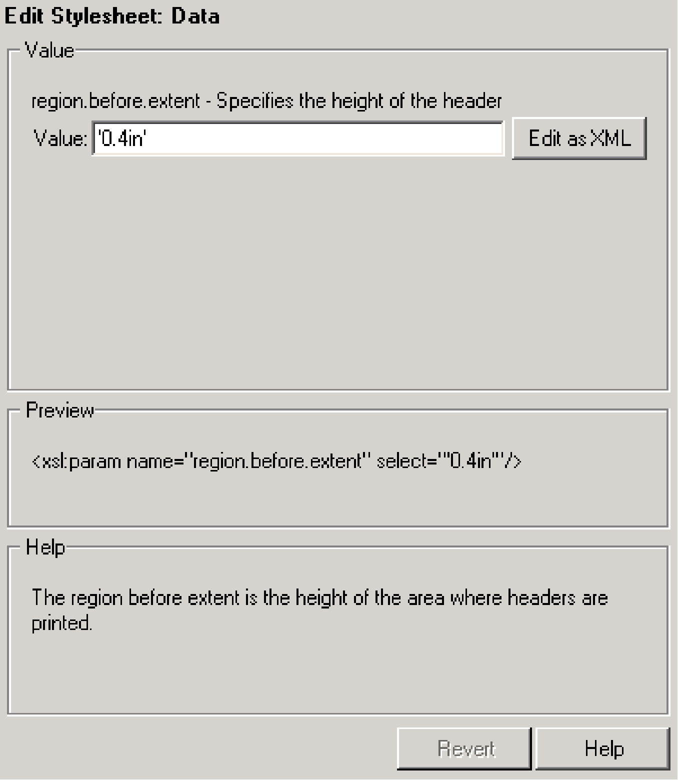
This option specifies page header/footer content and placement. The "Value (XML)" field is XML code which specifies the text or graphics to appear in the header or footer. The "Condition" field controls where and on what type of pages the content is used.

4 By default, the name of the graphic is `logo.bmp`. Change all instances of this name to `sample_logo.bmp` in the **Value (XML)** field.

5 Save the stylesheet.

6 Make sure that the amount of room available in the header is large enough to accommodate the image file.

a In the Options pane in the middle, double-click **Region Before Extent**, which is in the **Pagination and General Styles** folder.



Edit Stylesheet: Data

Value

region.before.extent - Specifies the height of the header

Value: '0.4in'

Preview

<xsl:param name="region.before.extent" select="'0.4in'"/>

Help

The region before extent is the height of the area where headers are printed.

b By default the value for the height of the header is `0.4` inch. Replace this value with `1.0in`.

c Save the stylesheet.

7 Generate the report with the new styles.

a Select **mfile-report.rpt** in the Outline pane on the left.

b In the selection lists under the **Report Format and Stylesheet** area of the Properties pane on the right:

- Specify `Acrobat (PDF)` for **File format**
- Specify `Logo stylesheet for PDF`.

c Click **Report** on the toolbar to generate the report.

Changing Font Size, Page Orientation, and Paper Type of a Generated Report

This example shows how to:

- Generate an XML source file without converting it to a supported report format
- Make section headers in a report larger
- Change the report page orientation to landscape
- Change the report paper type to A4

Create a custom stylesheet by editing an existing stylesheet to change the appearance of the `wsvar-report` report, which is provided with the MATLAB Report Generator software.

1 Generate a source file for the report.

   a Open the report by entering the following command in the MATLAB Command Window:

   ```
   setedit wsvar-report
   ```

   b In the **Report Format and Stylesheet** area of the Properties pane, change the format to **DocBook (no transform)**.

   c Check the **If report already exists, increment to prevent overwriting** check box.

   d Select **File > Report** to generate the report.

   The report-generation process creates an XML source file in the MATLAB Editor.

2 Convert the report to PDF format.

   a Select **Tools > Convert Source File** from the Report Explorer menu bar to open the File Converter.

   b From the **Source file** selection list, enter **wsvar-report0.xml**.

   c From the **File format** selection list, select **Acrobat (PDF)**.

- **d** From the **Stylesheet** selection list, select **Unnumbered Chapters and Sections**.
- **e** Click **Convert File**.

  The MATLAB Report Generator software converts the XML source file for `wsvar-report` to PDF format, and then opens the PDF document.

3 Make the report headers more prominent.

- **a** In the File Converter, click **Edit**.

  The Report Explorer displays the **Unnumbered Chapters and Sections** stylesheet.

- **b** In the Properties pane on the right, enter `Custom Large Section Headers` as the stylesheet name.
- **c** Enter the description `No chapter and section numbering, larger section titles`.
- **d** In the Outline pane on the left, select the **Custom Large Section Headers** stylesheet.
- **e** In the Options pane in the middle, select **Section Title Level 1 Properties**.
- **f** In the Properties pane on the right, click **Add to current stylesheet**.

  The **Section Title Level 1 Properties** data item appears in the Outline pane on the left as a child of the `Custom Large Section Headers` stylesheet.

- **g** In the Properties pane on the right, select the **Font Size** attribute.

  The Properties pane on the right displays an XML expression specifying font size as a multiple of the Body Font Size attribute.

- **h** Click **Edit as string**.

  The MATLAB Report Generator software converts the XML expression to a simple string, which appears in a pane labeled **Value**.

- **i** Enter the value `18pt`.

The size of the font is now fixed at 18 points, rather than being a multiple of the body font size attribute.

- j Select **File > Save** to save the stylesheet.
- k Save the stylesheet as `customheader.rgs`, in a folder in your MATLAB path.

  The `customheader.rgs` stylesheet appears as an available stylesheet in the Options pane in the middle of the Report Explorer. It also appears as an option in the File Converter.

4 Use the new stylesheet to convert the current XML source file.

- a In the **Stylesheet Editor: Main** Properties pane on the right, click **Send to File Converter**

  The File Converter appears, with the `customheader.rgs` stylesheet selected.
- b Click **Convert file**.

5 Change page orientation and paper type.

- a On the File Converter Properties pane, click **Edit**.
- b In the Options pane on the left, double-click the **Page Orientation** data item.
- c In the Properties pane on the right, use the selection list to change the value of the data item to `Landscape`.
- d In the Options pane in the middle, double-click **Paper Type** in the **Pagination and General Styles** folder.
- e In the Properties pane on the right, select **A4** from the selection list.
- f Save the stylesheet.

6 Generate the report `wsvar-report.xml` in PDF format using `customheader.rgs`..

The PDF report appears with horizontally oriented pages of slightly different dimensions.

Editing Font Size as a Derived Value in XML

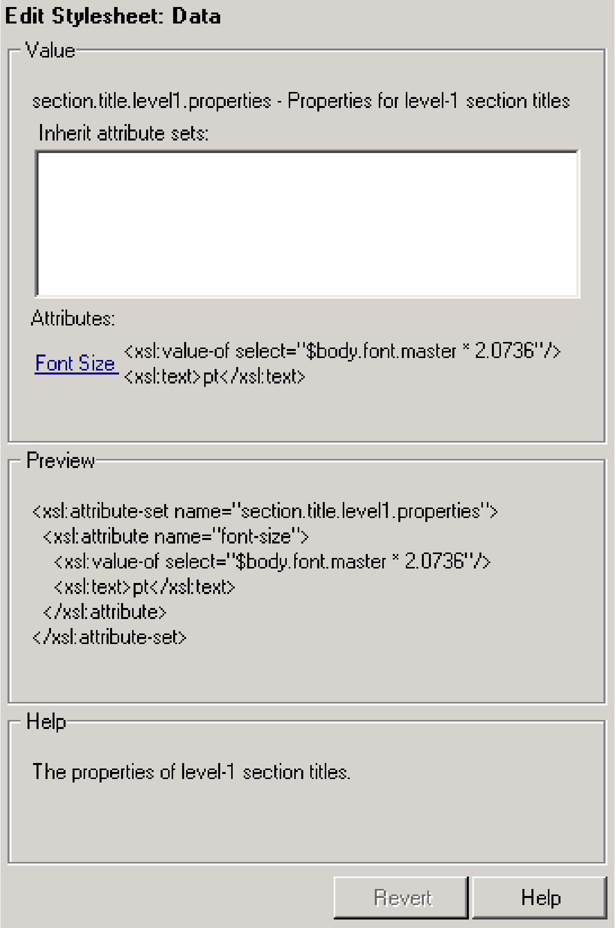
This example shows how to change the font size in a report to a value derived from other values. You do this by editing the PDF report's XML source directly.

1. Open the default print stylesheet for PDF documents.

2. In the Options pane in the middle, select and expand the **Property Sets** folder.

3. In the Options pane, double-click the **Section Title Level1 Properties** data item.

   The Properties pane on the right appears as follows.



Edit Stylesheet: Data

Value

section.title.level1.properties - Properties for level-1 section titles

Inherit attribute sets:

Attributes:

Font Size <xsl:value-of select="$body.font.master * 2.0736"/> <xsl:text>pt</xsl:text>

Preview

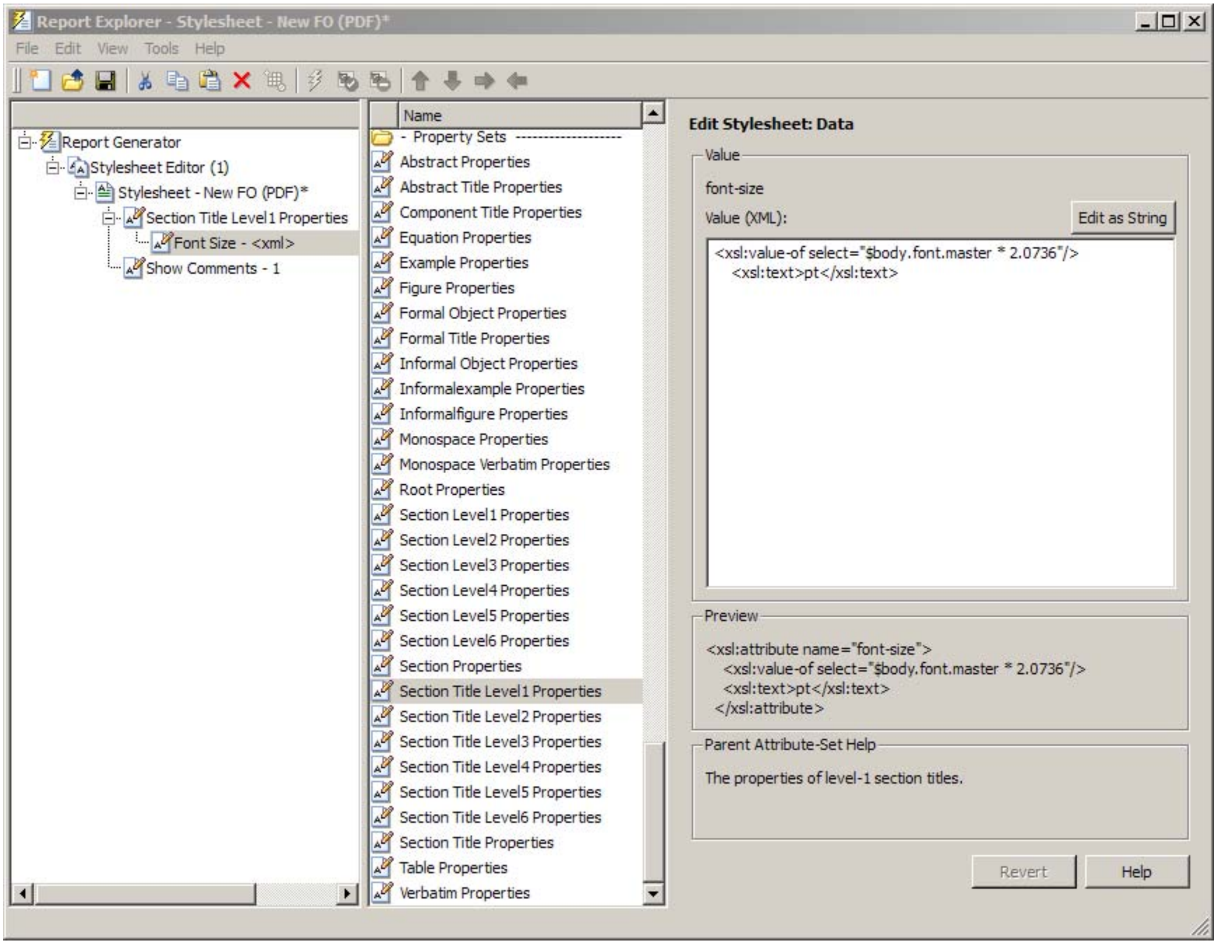
```
<xsl:attribute-set name="section.title.level1.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 2.0736"/>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Help

The properties of level-1 section titles.

4 In the **Attributes** area of the Properties pane on the right, click **Font Size - <xml>**.

The Report Explorer looks as follows.


Report Explorer - Stylesheet - New FO (PDF)*
File Edit View Tools Help
Report Generator
Stylesheet Editor (1)
Stylesheet - New FO (PDF)*
Section Title Level1 Properties
Font Size - <xml>
Show Comments - 1
Name
- Property Sets ------------------
Abstract Properties
Abstract Title Properties
Component Title Properties
Equation Properties
Example Properties
Figure Properties
Formal Object Properties
Formal Title Properties
Informal Object Properties
Informalexample Properties
Informalfigure Properties
Monospace Properties
Monospace Verbatim Properties
Root Properties
Section Level1 Properties
Section Level2 Properties
Section Level3 Properties
Section Level4 Properties
Section Level5 Properties
Section Level6 Properties
Section Properties
Section Title Level1 Properties
Section Title Level2 Properties
Section Title Level3 Properties
Section Title Level4 Properties
Section Title Level5 Properties
Section Title Level6 Properties
Section Title Properties
Table Properties
Verbatim Properties
Edit as String
Revert
Help

Edit Stylesheet: Data

Value

font-size

Value (XML):

```
<xsl:value-of select="$body.font.master * 2.0736"/>
  <xsl:text>pt</xsl:text>
```

Preview

```
<xsl:attribute name="font-size">
  <xsl:value-of select="$body.font.master * 2.0736"/>
  <xsl:text>pt</xsl:text>
</xsl:attribute>
```

Parent Attribute-Set Help

The properties of level-1 section titles.

The `font size` value is a product of `$body.font.master` and `2.0736`. To change the font size to a larger size, change the multiplication factor to `3.0736`.

Tip You specify the value for the `$body.font.master` data item in the **Body Font Master** property. This property is in the **Pagination and General Styles** category in the Options pane in the middle. The default value of this data item is 10. Changing this value causes the derived values to change accordingly.

Comparing XML Files

How to Compare XML Files

In this section...

Introduction

You can use MATLAB Report Generator software to compare a pair of XML text files.

The XML comparison tool processes the results into a report that you can use to explore the file differences.

You can access the XML comparison tool from:

- The MATLAB Current Folder browser context menu
- The MATLAB File and Folder Comparisons Tool
- The MATLAB command line.

The XML comparison tool compares the files using the "Chawathe" algorithm, as described in this 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`.

XML comparison reports display in the File and Folder Comparisons Tool. For more information about the File and Folder Comparisons Tool, see "Comparing Files and Folders" in the MATLAB Desktop Tools and Development Environment documentation.

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.

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 MATLAB Report Generator implementation of Chawathe's algorithm uses a comparison pattern that defines the thresholds. For more information, see "How The Matching Algorithm Works" on page 7-11.

Selecting Files to Compare

- "From the Current Folder Browser" on page 7-3
- "From the File and Folder Comparisons Tool" on page 7-4
- "From the Command Line" on page 7-4

From the Current Folder Browser

To compare two files in the same folder:

1. From the Current Folder browser, select the files you want to compare.

2. Right-click, and from the context menu, select **Compare Selected Files**.

If the selected files are XML files, the XML comparison tool performs a Chawathe analysis on the files and displays a report in the File and Folder Comparisons window.

The file you right-click to launch the XML comparison tool displays on the right side of the report.

For more information about comparisons of other file types with the File and Folder Comparisons Tool, such as text, MAT or binary, see “Comparing Files and Folders” in the MATLAB Desktop Tools and Development Environment documentation.

To compare files in different directories, you can use the File and Folder Comparisons Tool or command line access methods.

From the File and Folder Comparisons Tool

To compare files using the File and Folder Comparisons Tool, launch the tool from the MATLAB **Desktop** menu.

If the files you select to compare in the File and Folder Comparisons Tool are XML files, the XML comparison tool performs a Chawathe analysis of the XML files, and generates an HTML report.

From the Command Line

To compare XML files from the command line, enter

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

where `filename1` and `filename1` are XML files. This XML comparison functionality is an extension to the MATLAB `visdiff` function. For other file types, `visdiff` produces a MATLAB differences report.

If you want the MATLAB differences report for XML files, you can do this at the command line as follows:

```
xmlcomp.textCompare(filename1, filename2)
```

XML Comparison Demos

For an example with instructions, see the `mlxml_testplan` demo.

See Also

For explanations on how to use and understand the report and the XML comparison functionality, refer to:

- “Using the XML Comparison Report” on page 7-6

- "Known Limitations and Workarounds" on page 7-9

Using the XML Comparison Report

In this section...

Navigating the XML Comparison Report

The XML comparison report shows changes only. The report is a hierarchical view of the differences between two XML text files, and is not a hierarchical view of the original XML data.

To *step through changes*, use the toolbar or the **XML** menu. To move to the next or previous group of changes, either:

- Click the toolbar buttons Next ↓ or Previous ↑.
- Select **XML > Next** or **XML > Previous**

You can also click to select items in the hierarchical trees.

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

Report icons indicate the nature of each difference:

- ↷ Modified.
- ↦ Moved.
- ▢ Unmatched (items found only on one side).
- ↧ Reordered. This means items at the same level in the XML hierarchy have changed order.

Note Hierarchical node tags appear twice in the tree as nested nodes.

Hierarchical nodes appear twice 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.

Use the toolbar buttons or the **XML** menu for the following functions:

- **Expand All** — Expands every item in the tree.
- **Collapse All** — Collapses all items in the tree to the most compact view possible.
- **Reload Report** — Redraws the report. This does *not* run the Chawathe analysis again.

  To run the analysis again, click the **Refresh** button at the top of the File and Folder Comparisons window.
- **Printable Report** — Opens the Export Printable Report dialog box, where you can select a filename and location to save the report. The report is a noninteractive HTML document of the differences detected by the Chawathe algorithm for printing or archiving a record of the comparison.

See also "XML Comparison Demos" on page 7-4.

Saving Comparison Files in a Zip File

Temporary XML comparison files accumulate in `tempdir`, in subfolders with names starting with `mwxml`.

To zip the temporary files (such as log files) created during XML comparisons, for sharing or archiving, 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
```

Known Limitations and Workarounds

In this section...

Unexpected Results

If you see unexpected results within an XML comparison report, try reading the documentation section on "How The Matching Algorithm Works" on page 7-11.

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

The **Find** tool in the File and Folder Comparisons toolbar only searches visible report items. To search the entire report, click **Expand All** before searching.

Setting File and Memory Sizes

You can use the following command line options to manage browser file size and JVM memory.

These options are persistent across sessions.

First create an `xmlcomp.options` object.

```
obj = xmlcomp.options
```

You might need to increase the browser file size limit in order to display some comparisons. For example, to increase the limit in bytes for the size of file that you can display in the browser (from the default `7e6`), enter:

```
obj.setMaxBrowserFileSize(7e7)
```

To reset the amount of memory available to the external JVM process used for comparisons (the default is 1024 MB), enter:

```
obj.setJVMMemLimit(1024)
```

To restore factory values, use

```
obj.reset
```

Errors from UNC Paths

On a Windows platform, you should not start a MATLAB session, from a Universal Naming Convention (UNC) path, such as `\\mycompany\bigserver\matlab`. You will see errors if you do this. UNC paths are not supported for *`matlabroot`*.

You can work around this by mapping a drive to the location.

Resolving Cosmetic Issues with the HTML Browser

You might see rendering problems for certain sizes of window, due to issues with the version of ICE browser that ships with 64-bit Versions of MATLAB software. Try the **Redraw** button if you see missing or extra scroll bars. Next, try undocking the comparison window if you see double scroll bars.

How The Matching Algorithm Works

In this section...

Why Do I See Unexpected Results?

The core of the XML file comparison engine is Chawathe’s matching algorithm. This matching algorithm is a heuristic method based on a scoring system. This means that comparison results could be unexpected when many elements in each document are very similar.

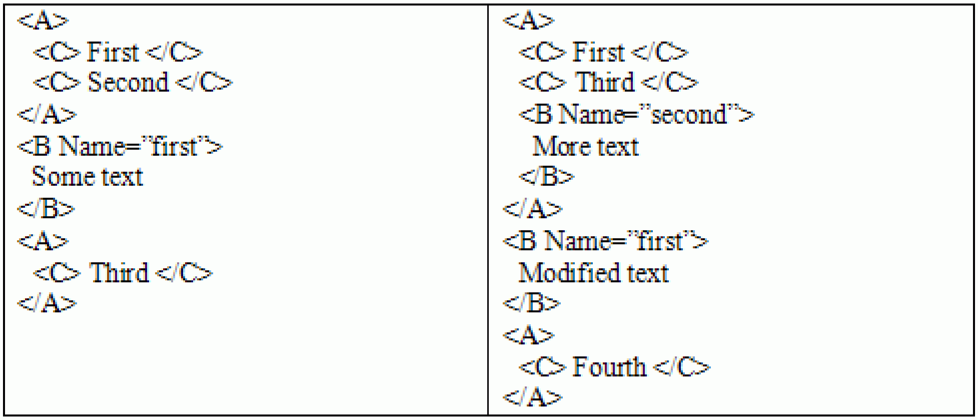
See the following sections for some examples.

How the Chawathe Algorithm Works

XML text documents are hierarchical data structures. Users can insert, delete, or reorder elements, modify their contents, or move elements across different parts of the hierarchy. The Chawathe algorithm can detect these different types of changes within the hierarchy of the document. As with conventional text differencing utilities, the Chawathe algorithm detects local text that is added, deleted, or changed, and additionally can prepare an edit script that can be used to create a report of the hierarchical location of detected differences.

The Chawathe algorithm attempts to match elements that are of the same category. The Chawathe paper refers to these categories as *labels*. In the following XML example documents (with labels `A`, `B`, and `C`):

- The three `C` elements on the left are compared with the three `C` elements on the right
- The single `B` element on the left is compared with the two `B` elements on the right



```
<A>
  <C> First </C>
  <C> Second </C>
</A>
<B Name="first">
  Some text
</B>
<A>
  <C> Third </C>
</A>
```

```
<A>
  <C> First </C>
  <C> Third </C>
  <B Name="second">
    More text
  </B>
</A>
<B Name="first">
  Modified text
</B>
<A>
  <C> Fourth </C>
</A>
```

The Chawathe algorithm matches a particular label by extracting a flat sequence of elements from the hierarchical document and attempting to match the elements in the sequences. In the example above, elements of the sequence

```
(<C> First </C>, <C> Second </C>, <C> Third </C>)
```

are matched against elements of the sequence

```
(<C> First </C>, <C> Third </C>, <C> Fourth </C>)
```

Sequences are matched using a Longest Common Subsequence (LCS) algorithm. For example, if `C` elements are matched on their text content, the LCS of the above sequences is given by:

```
(<C> First </C>, <C> Third </C>)
```

You can define a *score* for matching elements of a particular label in different ways. For instance, in the above example, `C` elements can be matched on text content, `B` can be matched on text content and on `Name`, and `A` on the number of `B` and `C` elements they have in common. To determine whether elements match or not, the Chawathe algorithm compares the score to a threshold.

The implementation can specify scoring methods, thresholds, the definition of labels, and the order in which labels are processed. These can be defined separately for each problem domain or type of XML file. The XML comparison tool provides suitable definitions for a set of common XML file types, and uses a default definition for any type of XML document it does not recognize.

Why Use a Heuristic Algorithm?

Chawathe's algorithm is a heuristic. That is, it cannot guarantee to return the optimal matching between two sequences. It is the use of a threshold mechanism in combination with an LCS algorithm that makes the algorithm a heuristic. A heuristic algorithm is preferable to an optimal matching because the heuristic is much faster.

An algorithm can only guarantee a mathematically optimal matching by exhaustively computing the score between all pairs of elements in the two sequences and choosing those pairs that maximize an overall matching score between the two sequences. This exhaustive approach is computationally very expensive because its running time increases exponentially with the length of the sequences to be matched.

Also a user's expectations can depend on context information that is not available to the matching algorithm (e.g., prior knowledge of the precise sequence of changes applied). This means even a mathematically optimal algorithm might match elements unexpectedly from a user's perspective.

In contrast with the mathematically optimal approach, Chawathe's algorithm guarantees linear running time for sequences that are the same or very similar. The worst-case scenario is quadratic running time for sequences that are entirely different.

The XML comparison tool performs best when the files to be compared are mostly similar. It becomes slower for files that contain more differences.

Examples of Unexpected Results

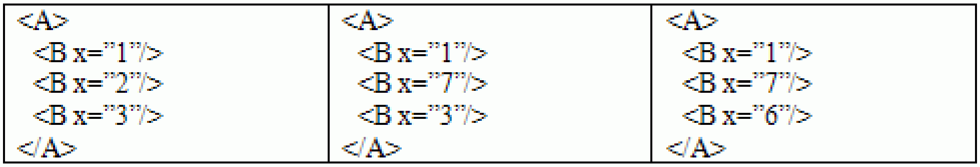
- "Elements Matched in Previous Comparisons Fail to Match" on page 7-13
- "Elements Matched Across Different Parts of the Hierarchy" on page 7-14
- "Two Sequences of Elements Are Cross-Matched" on page 7-16

Elements Matched in Previous Comparisons Fail to Match

Elements could fail to match even if they were matched in comparisons of previous versions of the documents. A seemingly small change in one of the properties used for matching can cause this to happen if it tips the score under the threshold.

Consider the following example where

- B elements are scored on the value of x
- A elements are scored on the ratio of matching B elements
- For both A and B the score is compared with a threshold of 0.5.



<A>
  <B x=”1”/>
  <B x=”2”/>
  <B x=”3”/>
</A>
	<A>
	  <B x=”1”/>
	  <B x=”7”/>
	  <B x=”3”/>
	</A>
		<A>
		  <B x=”1”/>
		  <B x=”7”/>
		  <B x=”6”/>
		</A>

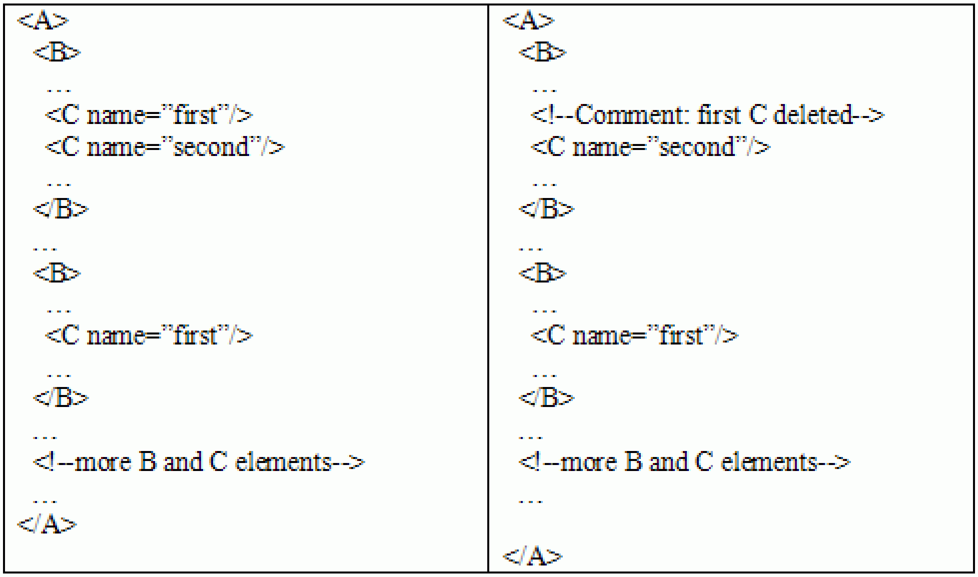
The left A and the middle A have two out of three B elements in common, resulting in a matching score of 2/3=0.66. The XML comparison tool marks the A elements as matched and the report shows that their contents have been *modified*.

When a user makes a further change to the middle document (resulting in the right document), and this new document is compared again to the left document, the matching score for A drops to 1/3=0.33. The algorithm considers the A elements unmatched this time. In this case, the difference between the two documents is marked as a *deletion* of A from the left document and an *insertion* of a new A into the right document.

This problem is likely to occur when there is little information available inside a single element to score a match. A seemingly small change in one of the properties used for matching could tip the score under the threshold, and therefore result in a large change in the outcome of the comparison.

Elements Matched Across Different Parts of the Hierarchy

Sometimes unexpected matches of similar items occur across different parts of the hierarchy. In the following example, C elements are matched on name:



```
<A>
  <B>
   …
   <C name=”first”/>
   <C name=”second”/>
   …
  </B>
  …
  <B>
   …
   <C name=”first”/>
   …
  </B>
  …
  <!--more B and C elements-->
  …
</A>
```

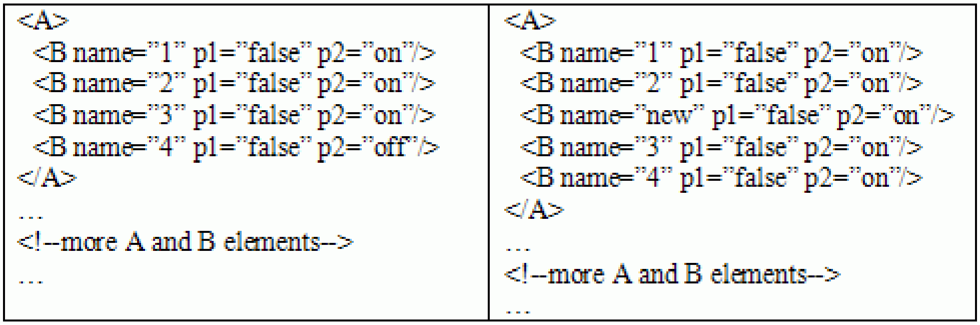
```
<A>
  <B>
   …
   <!--Comment: first C deleted-->
   <C name=”second”/>
   …
  </B>
  …
  <B>
   …
   <C name=”first”/>
   …
  </B>
  …
  <!--more B and C elements-->
  …

</A>
```

In this case, the user might expect to see the very *first* C element on the left marked as deleted, with the second and third C elements matched to the corresponding C element on the right. However, this might not happen, if the first C on the left is matched to the second C on the right, even though these two C elements exist in very different parts of the document hierarchy. This mismatch would result in the third C element on the left being marked as deleted, which the user might find unexpected.

This case is likely to occur when there are several potential matching candidates for a particular element. In other words, when elements of a particular label tend to be very similar. Whether such a spurious cross-matching occurs or not depends on all of the other C elements within the two documents. The LCS algorithm used for matching the two sequences favors *local* matches over *distant* ones. In other words, sub-sequences of elements that are close together in the first sequence tend to be matched to sub-sequences of elements that are close together in the second sequence. However, this locality is not always guaranteed, and the outcome depends on how other elements in the sequence are matched.

Two Sequences of Elements Are Cross-Matched

It is difficult to distinguish many similar potential matches and this could produce unexpected results. In the following example, B elements are scored on name, p1, and p2, and the score is compared to a threshold of 0.5.



```
<A>
  <B name=”1” p1=”false” p2=”on”/>
  <B name=”2” p1=”false” p2=”on”/>
  <B name=”3” p1=”false” p2=”on”/>
  <B name=”4” p1=”false” p2=”off”/>
</A>
...
<!--more A and B elements-->
...
```

```
<A>
  <B name=”1” p1=”false” p2=”on”/>
  <B name=”2” p1=”false” p2=”on”/>
  <B name=”new” p1=”false” p2=”on”/>
  <B name=”3” p1=”false” p2=”on”/>
  <B name=”4” p1=”false” p2=”on”/>
</A>
...
<!--more A and B elements-->
...
```

The right document contains one B element more than the left document, and therefore one of the B elements on the right must remain unmatched and the tool will mark one as inserted. However, since most B elements on the left potentially match most B elements on the right, it is impossible to predict exactly how the sequences will be matched. For instance, the comparison could generate the following result:

```
B name= 1      >  B name= 2
B name= 2      >  B name= new
B name= 3      >  B name= 3
B name= 4      >  B name= 4
```

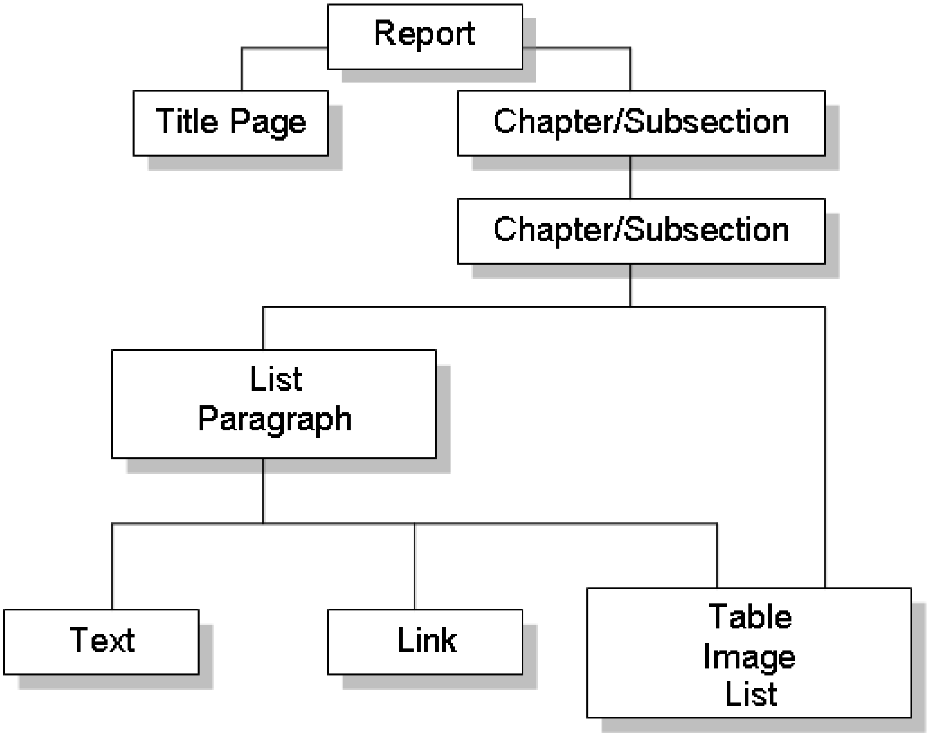
In this case, “B name= “1” on the right remains unmatched. As in the previous example, this depends on how all of the other B elements in the two documents are matched. This situation is likely to occur when elements have several potential matching candidates.

8

Component Reference

Formatting Components

Use Formatting components to insert documentation elements into your report. Formatting components must have the following parent/child relationships.



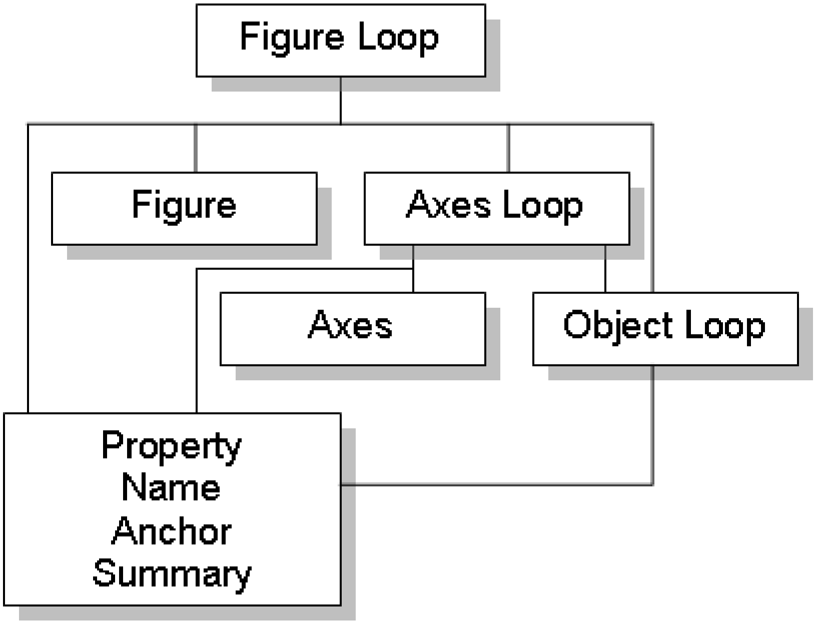
The following table describes the Formatting components.

Chapter/Subsection	Group portions of report into sections with titles
Empty Component	Group components so that you can easily move, activate, or deactivate them, or create a blank space in a list

Image	Insert image from external file into report
Link	Insert linking anchors or pointers into report
List	Create bulleted or numbered list from cell array or child components
Paragraph	Insert paragraph text into report
Table	Convert rectangular cell array into table and insert it into report
Text	Format and insert text into report
Title Page	Insert title page at beginning of report

Handle Graphics Components

Handle Graphics components report on Handle Graphics figures. The following figure shows the Handle Graphics component hierarchy.



The following table describes the Handle Graphics components.

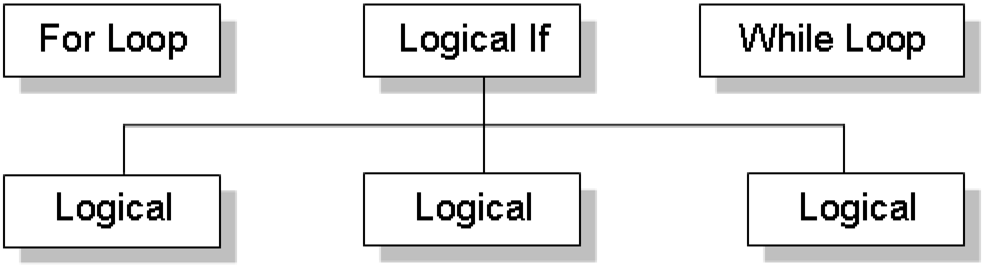
Axes Loop	Run child components for all axes objects in the MATLAB workspace
Axes Snapshot	Insert image of selected MATLAB axes objects into the generated report
Figure Loop	Apply child components to specified graphics figures
Figure Snapshot	Insert snapshot of Handle Graphics figure into report

Graphics Object Loop	Run child components for each Handle Graphics object that is currently open in the MATLAB workspace
Handle Graphics Linking Anchor	Designate location to which links point
Handle Graphics Name	Insert name of Handle Graphics object into the report
Handle Graphics Parameter	Insert property name/property value pair from Handle Graphics figure, axes, or other object
Handle Graphics Property Table	Insert table that reports on property name/property value pairs
Handle Graphics Summary Table	Insert table that summarizes Handle Graphics object properties

Logical and Flow Components

Logical and Flow Control components execute conditionally, enabling you to decide when a child component executes or how many times a child component executes.

These components have the following parent/child relationships.



For Loop	Iteratively executes child components
Logical Else	Specify an `else` condition for a `Logical If` component
Logical Elseif	Specify an `elseif` condition for the `Logical If` component
Logical If	Specify `logical if` condition
Logical Then	Specify a `then` condition for the `Logical If` component
While Loop	Iteratively execute child components while a specified condition is true

MATLAB Components

Use MATLAB components to interact with the MATLAB workspace.

Evaluate MATLAB Expression	Evaluate specified MATLAB expression
Insert Variable	Insert variable values into report
MATLAB Property Table	Insert table that includes MATLAB object property name/property value pairs
MATLAB/Toolbox Version Number	Insert table that shows version and release numbers and release date of MathWorks™ products
Variable Table	Insert table that displays all the variables in the MATLAB workspace

MATLAB Report Generator Components

This category includes “general utility” components.

Comment	Insert comment into XML source file created by the report-generation process
Import File	Import `ASCII` text file into report
Nest Setup File	Allow one report template (`.rpt` file) to run inside another
Stop Report Generation	Halt report generation
Time/Date Stamp	Insert time and date of report generation into report

9

Components — Alphabetical List

Nest Setup File
Paragraph
Stop Report Generation
Table
Text
Time/Date Stamp
Title Page
Variable Table
While Loop

Purpose

Run child components for all axes objects in the MATLAB workspace

Description

The Axes Loop component runs its child components for all axes objects in the MATLAB workspace. For information about working with looping components, see "Working with Looping Components" on page 4-3.

Object Selection

- **Loop type**:
  - `All axes`: Loops on all axes objects.
  - `Current axes`: Loops on the currently selected axes object.
- **Exclude objects which subclass axes**: Excludes objects, such as legends and color bars, from the loop.
- **Loop Menu**:
  - `Loop on axes with handle visibility "on"`: Loops only on visible axes objects.
  - `Loop on all axes`: Loops on all axes objects.
- **Search terms**: Specifies search terms for the loop. For example, to search for `Tag` and `My Data`, enter `"Tag", "My Data"`.

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_hg.chg_ax_loop`

See Also

Axes Snapshot, Figure Loop, Figure Snapshot, Graphics Object Loop, Handle Graphics Linking Anchor, Handle Graphics Name, Handle Graphics Parameter, Handle Graphics Property Table, Handle Graphics Summary Table

Purpose

Insert image of selected MATLAB axes objects into the generated report

Description

Inserts an image of selected MATLAB axes objects into the generated report.

Format

- **Image file format**: Specifies the image file format. Select `Automatic HG Format` to automatically choose the format best suited for the specified report output format. Otherwise, choose an image format that your output viewer can read. `Automatic HG Format` is the default option. Other options include:
  - `Automatic HG Format` (uses the Handle Graphics 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`

- **Capture figure from screen**: Captures the figure for the generated report directly from the screen. Options include:
  - `Client area only`: Captures part of the figure.
  - `Entire figure window`: Captures the entire figure window.

Print Options

- **Paper orientation**: Options include:
  - `Landscape`
  - `Portrait`
  - `Rotated`
  - `Use figure orientation`

  For more information about paper orientation, see `orient` command in the MATLAB documentation.
- **Image size**: Options include:
  - `Use figure PaperPositionMode setting`: Sets the image size in the report to the `PaperPositionMode` property of the figure. For more information about paper position mode, see `orient` in the MATLAB documentation.
  - `Automatic (same size as onscreen)`: Sets the image in the report to the same size as it appears on screen.
  - `Custom`: Specifies a custom image size. Specify the image size in the **Size** field and **Units** list.
- **Size**: Specifies the size of the figure snapshot in the format `[w h]` (width height). This field is active only if you choose `Custom` in the **Image size** selection list.

- **Units**: Specifies the units for the size of the figure snapshot. This field is active only if you choose `Set image size` in the **Custom** selection list.
- **Invert hardcopy**: Sets the `InvertHardcopy` property of Handle Graphics figures. This property inverts colors for printing; that is, it changes dark colors to light colors and vice versa. Options include:
  - `Automatic`: Automatically changes dark axes colors to a light axes color. If the axes color is a light color, it is not inverted.
  - `Invert`: Changes dark axes colors to light axes colors and vice versa.
  - `Don't invert`: Does not change the colors in the image displayed 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**: Options include:
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, the maximum size, and the units of measure.
  - `Use image size`: Sets the size of the image in the report to the same size as it appears on screen.
- **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 the units for the size of the snapshot. This field is active only if you choose `Zoom` or `Fixed size` in the **Image size** selection list.
- **Alignment**: Options are:
  - `Auto`
  - `Right`
  - `Left`
  - `Center`

  Only `PDF` and `RTF` reports support this option.
- **Title**: Specifies text to appear above the snapshot.
- **Caption**: Specifies text to appear under the snapshot.

Insert Anything into Report?

Yes. Image.

Class

`rptgen_hg.chg_ax_snap`

See Also

`Axes Loop`, `Figure Loop`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

Chapter/Subsection

Purpose

Group portions of report into sections with titles

Description

This component groups portions of the report into sections. Each section has a title and content.

The following rules apply to this component:

- Child components appear inside the section created by this component.
- Selecting the **Get title from first child component** check box prevents this component from accepting paragraph-level children. In this case, this component's first child must be a `Text` component.
- This component can have Chapter/Subsection components as its children.
- Sections can be nested. There are seven levels of nesting possible. The seventh nested section in the report is untitled, although the child components of this section include information into the report.

Chapter Numbering

By default, chapters are numbered and sections are not numbered. Specify chapter and section numbering using a stylesheet. For more information about chapter and section numbering options in Web and print stylesheets, see "Setting the Report Format" on page 3-4.

Section Title

- **Title**: Specifies a title to display in the generated report:
  - `Automatic`: Automatically generates a title.
  - `Custom`: Specifies a custom title.

- **Numbering**: Specifies a numbering style for the report:
  - `Automatic`: Numbers by context.
  - `Custom`: Allows you to create your own numbering style.
- **Section Type**: Shows you in which level a selected section resides.

Insert Anything into Report?

Yes. Chapter or section.

Class

`rptgen.cfr_section`

See Also

`Empty Component`, `Image`, `Link`, `List`, `Paragraph`, `Table`, `Text`, `Title Page`

Purpose

Insert comment into XML source file created by the report-generation process

Description

This component inserts a comment into the XML source file created by the report-generation process. This comment is not visible in the generated report.

This component can have children. Child components insert their output into the XML source file, but this does not appear in the generated report.

To make comment text appear in the report:

1. Edit the XML source file (which has the same name as your report file, but has a `.xml` extension).
2. Find the `comment` area in the XML source file by locating the comment tags `<--` and `-->`.
3. Remove the comment tags.
4. Convert the XML source file using the `rptconvert` command.

Properties

- **Comment text**: Specifies comments to include in the report.
- **Show comment in Generation Status window**: Displays comments in the **Generation Status** tab while the report generates.
- **Status message priority level**: Specifies the priority level of the status messages that appear during report generation. Priority options range from `1) Error messages only` to `6) All messages`. The default is `3) Important messages`. This option is only available if you select the **Show comment in Generation Status window** option.

Insert Anything into Report?

No. This component inserts comments, which can appear in the report, into the report's XML source file.

Class

`rptgen.crg_comment`

See Also

"Converting XML Documents" on page 3-11, `Import File`, `Nest Setup File`, `Stop Report Generation`, `Time/Date Stamp`

Purpose

Group components so that you can easily move, activate, or deactivate them, or create a blank space in a list

Description

This component does not insert anything into the generated report. It can have any component as a child. You can use it to group components together so that you can easily move, activate, or deactivate them, or create a blank space in a list.

If the MATLAB Report Generator software does not recognize a given component when loading a report template, it replaces the unrecognized component with this component.

Insert Anything into Report?

No.

Class

`rptgen.crg_empty`

See Also

`Chapter/Subsection`, `Image`, `Link`, `List`, `Paragraph`, `Table`, `Text`, `Title Page`

Purpose

Evaluate specified MATLAB expression

Description

This component evaluates a specified MATLAB expression. You can include code and/or command-line output in the report.

Properties

- **Insert MATLAB expression in report**: Causes the MATLAB expression that this component evaluates to appear in the report.
- **Display command window output in report**: Includes the command window output that results from the evaluation of the specified MATLAB expression.
- **Expression to evaluate in the base workspace**: Specifies the expression to evaluate in the MATLAB workspace. You can include this attribute in a loop component to obtain a handle to the object in whose context the loop is executing. For example:
  - Enter the following if you want to obtain a handle to the block in whose context a `Block Loop` is executing:

    ```
    %Access to block
    blockname = RptgenSL.getReportedBlock;
    blockHandle = get_param(blockname,'handle');
    ```

  - Enter the following if you want to obtain a handle to the model in whose context a `Model Loop` is executing:

    ```
    %Access to model
    modelname = RptgenSL.getReportedModel;
    modelHandle = get_param(Modelname,'handle');
    ```

  - Enter the following if you want to obtain a handle to the signal in whose context a `Signal Loop` is executing:

    ```
    %Access to signal
    signalname = RptgenSL.getReportedSignal;
    signalHandle = get_param(signalname,'handle');
    ```

- Enter the following if you want to obtain a handle to the system in whose context a `System Loop` is executing:

  ```
  %Access to system
  systemname = RptgenSL.getReportedSystem;
  systemHandle = get_param(systemname,'handle');
  ```

  Similarly, you can use `set` calls in place of `get` calls in the previous examples to set your current block, model, signal, or system within the context of a loop component.

- **Evaluate this expression if there is an error**: Evaluates another MATLAB expression if the specified expression produces an error. You must enter in this field the expression to evaluate in case of an error.

Insert Anything into Report?

Inserts text only if you select one of the following options:

- `Insert MATLAB expression string in report`
- `Display command window output in report`

Class

`rptgen.cml_eval`

See Also

`Insert Variable`, `MATLAB Property Table`, `MATLAB/Toolbox Version Number`, `Variable Table`

Figure Loop

Purpose

Apply child components to specified graphics figures

Description

This component applies each child component to specified figures in the report. For more information about working with this component, see “Working with Looping Components” on page 4-3.

Figure Selection

- **Include figures**
  - `Current figure only`: Includes only the current figure in the report.
  - `Visible figures`: Loops on all visible figures. The `Data figures only` option is checked by default and excludes figures with `HandleVisibility = 'off'` from the loop.
  - `All figures with tags`: Loops on figures with specified tags, select When you select a given tag, all figures with that tag appear in the loop, regardless of whether each figure is visible or whether its `HandleVisibility` attribute is `'on'` or `'off'`.

    The tag field (located under `All figures with tags`) shows selected tags. To add tags to this field, type in the tag names, separating them with new lines.
- **Loop Figure List**: Shows all figures that are included in the loop. If the report template generates new figures or changes existing figures, figures in the **Loop Figure List** are not the figures that are reported on.

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_hg.chg_fig_loop`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

Figure Snapshot

Purpose

Insert snapshot of Handle Graphics figure into report

Description

This component inserts a snapshot of a Handle Graphics figure into the report.

Format

- **Image file format**: Specifies the image file format. Select `Automatic HG Format` to automatically choose the format best suited for the specified report output format. Otherwise, choose an image format that your output viewer can read. `Automatic HG Format` is the default option. Other options include:
  - `Automatic HG Format` (uses the Handle Graphics 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`
- **Capture picture from screen**:
  - `Client area only`: Captures a portion of the figure window.
  - `Entire figure window`: Captures the whole window.

Print Options

- **Paper orientation**: Options include:
  - `Landscape`
  - `Portrait`
  - `Rotated`
  - `Use figure orientation`

  For more information about paper orientation, see the `orient` command in the MATLAB documentation.
- **Image size**:
  - `Use figure PaperPositionMode setting`: Uses the figure's `PaperPositionMode` property as 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)`: Causes the image in the report to appear the same size as on screen.
  - `Custom`: Specifies a custom image size. Set the image size in the **Size** field and **Units** list.
- **Size**: Specifies the size of the figure snapshot in the form `w h` (width times height). This field is active only if you choose `Custom` from the **Image size** selection list.

- **Units**: Specifies units for the size of the figure snapshot. This field is active only if you choose `Custom` in the **Image size** selection list.
- **Invert hardcopy**: Sets print colors using the figure's `InvertHardcopy` property, which inverts colors for printing. Options include:
  - `Automatic`: Automatically changes dark axes colors to a light axes color. If the axes color is a light color, it is not inverted.
  - `Invert`: Changes dark axes colors to light axes colors and vice versa.
  - `Don't invert`: Does not change the colors in the image.
  - `Use figure's InvertHardcopy setting`: Uses the value of the `InvertHardcopy` property set in the Handle Graphics image.
  - `Make figure background transparent`: Makes the image background transparent.

Display Options

- **Scaling**: Options include:
  - `Fixed size`: Specifies the number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
  - `Use image size`: Causes the size of the image in the report to appear the same size as on screen.
- **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** selection list.

- **Alignment**: Options include:
  - `Auto`
  - `Right`
  - `Left`,
  - `Center`

  Only `PDF` and `RTF` reports support this option.
- **Title**: Specifies a title for the figure:
  - `Custom`: Specifies a custom title.
  - `Name`: Specifies the figure name as the title.
- **Caption**: Specifies text to appear under the snapshot.

Insert Anything into Report?

Yes. Image.

Class

`rptgen_hg.chg_fig_snap`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

For Loop

Purpose

Iteratively executes child components

Description

This component functions like the MATLAB `for` loop, except that instead of executing a statement, it executes its child components. It must have at least one child component to execute.

Loop Type

The loop type can have incremented indices or a vector of indices. For more information on for loops and indices, see `for` in the MATLAB documentation.

- **Incremented indices**: Executes a `for` loop of the form:

  ```
  for varname=x:y:z
  ```

  - **Start**: Corresponds to `x` in the previous expression.
  - **Increment**: Corresponds to `y` in the previous expression.
  - **End**: Corresponds to `z` in the previous expression.

- **Vector of Indices**: Executes a `for` loop of the form:

  ```
  for varname=[a b c ...]
  ```

  Specify appropriate values in the **Vector** field in the form `a b c ...`.

Workspace Variable

- **Show index value in base workspace**: Displays the loop index in the MATLAB workspace while other components execute.
- **Variable name**: Allows you to specify the variable name. The default is `RPTGEN_LOOP`.
- **Remove variable from workspace when done**: Removes the loop index from the MATLAB workspace. This option is only available if you select the **Show index value in base workspace** option.

Insert Anything into Report?

No.

Class

`rptgen_lo.clo_for`

See Also

`Logical Else`, `Logical Elseif`, `Logical If`, `Logical Then`, `While Loop`

Graphics Object Loop

Purpose

Run child components for each Handle Graphics object that is currently open in the MATLAB workspace

Description

This component runs its child components for each Handle Graphics object that is currently open in the MATLAB workspace. The component inserts a table into the generated report.

Select Objects

- **Exclude GUI objects (uicontrol, uimenu, ...)**: Excludes GUI objects, such as `uicontrol` and `uimenu`, from the loop.
- **Loop list**: Specifies the loop level for Handle Graphics objects:
  - `Loop on objects with handle visibility "on"`
  - `Loop on all objects`
  - **Search for**: Allows you to enter space-delimited search terms.

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_hg.chg_obj_loop`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Figure Snapshot`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

Handle Graphics Linking Anchor

Purpose

Designate location to which links point

Description

This component designates a location to which links point. It should have a looping component as its parent.

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 figure, axes, or object as a linking anchor. If the `Figure Loop` component is this component's parent, the linking anchor points to the current figure. Similarly, if the `Graphics Object Loop` is this component's parent, the linking anchor points to the current object.
  - `Figure`: Sets the linking anchor to the current figure.
  - `Axes`: Sets the linking anchor to the current axes.
  - `Object`: Sets the linking anchor to the current object.

Insert Anything into Report?

Yes. Anchor.

Class

`rptgen_hg.chg_obj_anchor`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

Handle Graphics Name

Purpose

Insert name of Handle Graphics object into the report

Description

This component inserts the name of a Handle Graphics object as text into the report. You can use this component to create a section title based on the current figure by making it the first child component of a `Chapter/Subsection` component, and then selecting the `Chapter/Subsection` component's **Get title from first child component** option.

Properties

- **Display name as**:
  - `Type Name`
  - `Type  Name`
  - `Type:  Name`
- **Show name of current**: Options include:
  - `Figure`: Shows the name of the current figure. The first nonempty figure parameter determines the name.
  - `Axes`: Shows the name of the current axes. The first nonempty axes parameter determines the name.
  - `Other Object`: Sets the name of the current object to the figure's `CurrentObject` parameter and its first nonempty parameter.

Insert Anything into Report?

Yes. Text.

Class

`rptgen_hg.chg_obj_name`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Parameter`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

Handle Graphics Parameter

Purpose

Insert property name/property value pair from Handle Graphics figure, axes, or other object

Description

This component inserts a property name/property value pair from a Handle Graphics figure, axes, or other object.

Property Selection

- **Get property from current**: Reports on a specified Handle Graphics object:
  - `Figure`: Inserts a figure's property name/property value pair.
  - `Axes`: Inserts an axes' property name/property value pair.
  - `Object`: Inserts an object's property name/property value pair.
- **Figure property**: Specifies the type of property to include. The `All` option shows every parameter for the current object.

Display Options

- **Title**: Specifies a title for the generated report:
  - `None` (default): No title.
  - `Automatic`: Automatically generates the title from the parameter.
  - `Custom` Specifies a custom title.
- **Size limit**: Limits the width of the display in the generated report. Units are in pixels. The size limit of a given table is the hypotenuse of the table width and height [`sqrt(w^2+h^2)`]. The size limit of a text string equals its number of characters squared. If you exceed the size limit, the variable appears in condensed form, such as `[64x64 double]`. Setting a size limit of `0` displays the variable in full, no matter how large it is.
- **Display as**: Choose a display style:
  - `Auto table/paragraph`: Displays as a table or paragraph.
  - `Inline text`: Displays in line with the surrounding text.
  - `Paragraph`: Displays as a paragraph.
  - `Table`: Displays as a table.

- **Ignore if value is empty**: Excludes empty parameters from the generated report.

Insert Anything into Report?

Yes. Text.

Class

`rptgen_hg.chg_property`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Property Table`, `Handle Graphics Summary Table`

Handle Graphics Property Table

Purpose

Insert table that reports on property name/property value pairs

Description

This component inserts a table that reports on property name/property value pairs.

For more information on using this component, see “Working with Property Table Components” on page 4-9.

Select Graphics Object

- **Object type**: Specifies an object type for the generated report:
  - `Figure`
  - `Axes`
  - `Object`
- **Filter by class**: Specifies a class or classes for the table.

Table

You can select a preset table, which is already formatted and set up, from the list in the upper-left corner of the attributes page.

To create a custom table, edit a preset table, such as `Blank 4x4`. Add and delete rows and add properties. To open the Edit Table dialog box, click **Edit**.

For details about creating custom property tables, see “Working with Property Table Components” on page 4-9.

- **Preset table**: Specifies the type of table to display the object property table.
  - `Defaults`
  - `Callbacks`
  - `Graphics`
  - `Printing`
  - `Blank 4x4`

  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 in the table, select the **Split property/value cells** check box. In this case, the table is in split mode and there can be only one property name/property value pair in a 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. In this case, the table is in nonsplit mode, which supports more than one property name/property value pair per cell. It also supports text.

  Before switching from nonsplit mode to split mode, make sure that you have only one property name/property value pair per table cell. If you have more than one property name/property value pair or any text, only the first property name/property value pair appears in the report; subsequent pairs and text are omitted.

- **Display outer border**: Displays the outer border of the table in the generated report.

Table Cells

Select table properties to modify. The selection in this pane affects the availability of fields in the **Title Properties** pane.

If you select `Figure Properties`, only the **Contents** and **Show** options appear. If you select any other object in the **Table Cells** pane, the **Lower border** and **Right border** options appear.

Title Properties

- **Contents**: Enables you to modify the contents of the table cell selected in the **Table Cells** pane. Options include:
  - `Left`
  - `Center`
  - `Right`
  - `Double justified`

- **Show as**: Enables you to choose the format for the contents of the table cell. Options include:
  - `Value`
  - `Property Value`
  - `PROPERTY Value`
  - `Property:  Value`
  - `PROPERTY: Value`
  - `Property - Value`
  - `PROPERTY - Value`
- **Alignment**: Aligns text in the table cells. Options are:
  - `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.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_hg.chg_prop_table`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Summary Table`

Handle Graphics Summary Table

Purpose

Insert table that summarizes Handle Graphics object properties

Description

This component inserts a table that summarizes Handle Graphics object properties. Each row in the table represents an object. Each column in the table represents a property. You can specify object properties to include in the report.

Properties

- **Object type**: Specifies the object type to display in the generated report. Options include:
  - `figure`
  - `axes`
  - `object`

  The available options in the **Select Objects** pane depend on your selection in the **Object type** menu.
- **Table title**: Specifies a title for the table in the generated report. Options include:
  - `Automatic`: Generates a title automatically.
  - `Custom`: Specifies a custom title.

Property Columns

- **Property columns**: Specifies object properties to include in the table in the generated report.
  - To add a property:
    1. Select the appropriate property level in the menu.
    2. In the list under the menu, select the property to add and click **Add**.
  - To delete a property, select its name and click **Delete**.

  Some entries in the list of available properties (such as `Depth`) are “virtual” properties which 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 table in the generated report.

Object Rows

Insert anchor for each row: Inserts an anchor for each row in the summary table.

Figure Selection

The options displayed in the **Figure Selection** pane depend on the object type selected in the **Object type** list:

- If **Object type** is `figure`, the following options appear:
  - **Include figures**
    - `Current figure only`: Includes only the current figure in the report.
    - `Visible figures`: Executes child components for figures that are currently open and visible. The `Data figures only` option is checked by default. This option excludes figures with `HandleVisibility = 'off'` from the loop.
    - `All figures with tags`: Includes all figures with a specified tag regardless of whether they are visible or their `HandleVisibility` parameter is `'on'` or `'off'`. The tag selection list, located under this option, shows available tags. You can add tag names to this list.
  - **Data figure only (Exclude applications)**: Shows only data figures.
  - **Loop Figure List**: Shows figures within the current set of figures to display.
- If **Object type** is `axes`, the following options appear:
  - **Loop type**:
    - `All axes`: Loops on all axes objects.

    - `Current axes`: Loops on the selected axes object.
  - **Exclude objects which subclass axes**: Excludes objects such as legends and color bars.
  - **Loop Menu**:
    - `Loop on axes with handle visibility "on"`: Loops on visible axes objects.
    - `Loop on all axes`: Loops on all axes objects.
  - **Search terms**: Specifies search terms for the loop. For example, to search for `Tag` and `My Data`, enter `"Tag", "My Data"`.
- If **Object type** is `object`, the following options appear:
  - **Exclude GUI objects (uicontrol, uimenu, ...)**: Excludes GUI objects, such as `uicontrol` and `uimenu`, from the loop.
  - **Loop menu**: Specifies the loop level:
    - `Loop on objects with handle visibility "on"`
    - `Loop on all objects`
  - **Search for**: Specifies space-delimited search terms.

Insert Anything into Report?

Yes. Table.

Class

`rptgen_hg.chg_summ_table`

See Also

`Axes Loop`, `Axes Snapshot`, `Figure Loop`, `Figure Snapshot`, `Graphics Object Loop`, `Handle Graphics Linking Anchor`, `Handle Graphics Name`, `Handle Graphics Parameter`, `Handle Graphics Property Table`

Purpose

Insert image from external file into report

Description

This component inserts an image from an external file into the report. It can have the `Chapter/Subsection` or `Paragraph` component as its parent. If the `Paragraph` component is its parent, you must select the **Insert as inline image** check box.

Class

- **File name**: Specifies the image file name. You can enter this name manually, or use the **Browse** button **(...)** to find the image file.

  The image must be in a format that your viewer can read. An error like the following appears if you specify the name of an image file that does not exist:

  ```
  No file name.  Could not create graphic.
  ```

  This field supports `%<VariableName>` notation. For more information about this notation, see "%<VariableName> Notation" on page 9-68 on the `Text` component reference page.
- **Copy to local report files directory**: Saves a copy of the image to a local report files folder.

Display Options

- **Scaling**:
  - `Fixed size`: Specifies number and type of units.
  - `Zoom`: Specifies the percentage, maximum size, and units of measure.
  - `Use image size`: Causes the image in the report to appear the same size as on screen.
- **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** 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** selection list.
- **Alignment**: Options are:
  - `Auto`
  - `Right`
  - `Left`
  - `Center`

  Only `PDF` and `RTF` reports support this format.
- **Title**: Specifies text to appear above the snapshot.
- **Caption**: Specifies text to appear under the snapshot.

Preview

The image that you specify in the **Image file name** field appears in this pane. You cannot preview Adobe® PostScript® images, or images with formats that the `imread` function does not support, such as `.gif`.

Clicking an image causes it to display in full size.

Insert Anything into Report?

Yes. Image.

Class

`rptgen.cfr_image`

See Also

`Chapter/Subsection`, `Empty Component`, `List`, `Paragraph`, `Table`, `Text`, `Title Page`

Import File

Purpose

Import `ASCII` text file into report

Description

This component imports an `ASCII` text file into the report.

Properties

- **File name**: Specifies the name of the file to import into the text field. You can enter a name, or use the **Browse** button **(...)** to find the file.
- **Import file as**: Specifies formatting for the imported file. Options include:
  - `Plain text (ignore line breaks)`: Imports the file as plain text without any line breaks (no paragraphs). If you select this option, the `Import File` component acts like the `Text` component, so it should have the `Paragraph` component as its parent.

    The examples in this section use the following text as the input file:

    ```
    This is the first row of text from the imported file.
      The second row follows a line break in the first row.

    There is a blank line above the third row.
    ```

    `Plain text (ignore line breaks)` produces the following formatting for the example file:

    ```
    This is the first row of text from the imported file.
    The second row follows a line break in the first row.

    There is a blank line above the third row.
    ```

  - `Paragraphs defined by line breaks`: Imports the file as text, in paragraphs with line breaks (hard returns or carriage returns). This option produces the following formatting for the example file:

    ```
    This is the first row of text from the imported file.
    The second row follows a line break in the first row.

    There is a blank line above the third row.
    ```

- `Paragraphs defined by empty rows`: Imports the file as text, in paragraphs with empty rows (rows that include no text). This option produces the following formatting for the example file:

  ```
  This is the first row of text from the imported file.
  The second row follows a line break in the first row.

  There is a blank line above the third row.
  ```

- `Text (retain line breaks)` (default): Imports the file as plain text with line breaks. This option produces the following formatting for the example file:

  ```
  This is the first row of text from the imported file.
  The second row follows a line break in the first row.

  There is a blank line above the third row.
  ```

- `Fixed-width text (retain line breaks)`: Imports the file as fixed-width text (all letters have the same width or size), including line breaks. This option is useful for importing MATLAB files. This option produces the following formatting for the example file:

  ```
  This is the first row of text from the imported file.
  The second row follows a line break in the first row.

  There is a blank line above the third row.
  ```

- `DocBook XML`: Inserts an XML source file, and makes no changes to its format.
- `Formatted Text (RTF/HTML)`: Inserts an `RTF` or `HTML` source file, and makes no changes to its format.
- `Syntax highlighted MATLAB code`: Inserts a MATLAB file.

The **File Contents** field displays the first few lines of the file to be imported.

Insert Anything into Report?

Yes.

- Inserts text if you select one of the following options:
  - `Plain text (ignore line breaks)`
  - `Text (retain line breaks)`
  - `Fixed-width text (retain line breaks)`
- Inserts paragraphs if you select one of the following options:
  - `Paragraphs defined by line breaks`
  - `Paragraphs defined by empty rows`
- Inserts the contents of an XML file if you select the `DocBook XML` option.
- Inserts the contents of the `RTF` or `HTML` file if you select the `Formatted text (RTF/HTML)` option.
- Inserts a link to a file if you import the file into an HTML report.

Class

`rptgen.crg_import_file`

See Also

`Comment`, `Nest Setup File`, `Stop Report Generation`, `Time/Date Stamp`

Insert Variable

Purpose

Insert variable values into report

Description

This component inserts the value (and optionally, the name) of each the following variables into the report:

- A variable from the MATLAB workspace
- A variable from a MAT-file
- A global variable

Source

- **Variable name**: Specifies the name of the variable:
  - `%<VariableName>`: Inserts the value of a variable from the MATLAB workspace into the report.
- **Variable location**: Options include:
  - `Base Workspace`: Gets a variable from the MATLAB workspace.
  - `MAT File`: Gets a variable from a binary file with a `.mat` extension.
  - `Global variable`: Gets a global variable.

Display Options

- **Title**: Specify a title for the report:
  - `Automatic`: Generates a title automatically.
  - `Custom`: Specifies a custom title.
  - `None`: Specifies 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 the hypotenuse of the table width and height [`sqrt(w^2+h^2)`]. The size limit of a given text string is the number of characters squared. If you exceed the size limit, the variable appears in condensed form, such as `[64x64 double]`. Setting a size limit of `0` displays the variable in full, regardless of its size.
- **Display as**: Choose a display style from the menu:
  - `Table`: Displays as a table.

  - `Paragraph`: Displays as a paragraph.
  - `Inline text`: Displays in line with the surrounding text.
  - `Auto table/paragraph`: Displays as a table or paragraph.
- **Ignore if value is empty**: Excludes empty parameters from the generated report.

Insert Anything into Report?

Yes. Text.

Class

`rptgen.cml_variable`

See Also

`Evaluate MATLAB Expression`, `MATLAB Property Table`, `MATLAB/Toolbox Version Number`, `Variable Table`

Link

Purpose

Insert linking anchors or pointers into report

Description

This component inserts linking anchors or pointers into the report.

Properties

- **Link type**: Select the type of link to insert into the report. Options include:
  - `Linking anchor`: Specifies a link to an anchor.
  - `Internal document link`: Specifies a location in the report (as specified by an anchor).
  - `URL (external) link`: Specifies a link to a Web site.
- **Link identifier**: Indicates the location to which the link points. It can contain only `ASCII` characters, and it is not visible in the generated report.

  The link identifier options are context sensitive; their formats differ depending on the link type you select. For example, to link to an external file `foo.txt`, specify the link identifier as follows:
  - On UNIX systems:

    ```
    file:///home/janedoe/foo.txt
    ```
  - OnMicrosoft Windows systems:

    ```
    H:\foo.txt
    ```
- **Link text**: Specifies text to use in the link.
- **Emphasize link text**: Italicizes the link text.

Insert Anything into Report?

Yes. Text or anchor.

Class `rptgen.cfr_link`

See Also `Chapter/Subsection`, `Empty Component`, `List`, `Paragraph`, `Table`, `Text`, `Title Page`

Purpose

Create bulleted or numbered list from cell array or child components

Description

This component creates a bulleted or numbered list from a cell array or child components.

List Content

- **Create list from workspace cell array**: Creates the list from of the 1-by-n or n-by-1 cell array. This option is not available when this component has child components — in this case, the list automatically generates from the child components.
- **List title**: Specifies the title of the list.

List Formatting

- **List style**:
  - `Bulleted list`
  - `Numbered list`.
- **Numbering style**: Specifies a numbering style for numbered lists. This setting is supported only in the RTF/DOC report format. Options include:
  - `1,2,3,4,...`
  - `a,b,c,d,...`
  - `A,B,C,D,...`
  - `i,ii,iii,iv,...`
  - `I,II,III,IV,...`
- **Show parent number in nested list (1.1.a)**: Displays all level numbers in a nested list. You can create a nested list by putting one cell array inside another or by nesting one `List` component inside another. Following is an example of how a list appears when you select this option:

```
1. Example
2. Example
 2.1. Example
 2.2. Example
```

```
  2.2.a. Example
  2.2.b. Example
3. Example
```

This option is not available if you select `Show only current list value (a)`.

- **Show only current list value (a)**: Displays only the current list value. Following is an example of how a list appears when you select this option:

```
1. Example
2. Example
 1. Example
 2. Example
  1. Example
  2. Example
3. Example
```

This option is not available if you select `Show parent number in nested list (1.1.a)`.

Example 1: Creating a Nested List

Consider the following report template, which includes a nested list created by putting a `List` component inside another `List` component:

```
[-] Report - Unnamed.rpt
   [-] Bulleted list from child components
      [ ] Text - sky
      [ ] Table - varname
      [ ] Image - test.jpg
      [ ] Text - grass
      [-] Bulleted list from child components
         [ ] Text - clouds
         [ ] Text - sun
      [-] Paragraph - information
```

This report template generates a report that includes the following bulleted lists:

- sky
- varname, the table from the variable
- test.jpg, a snapshot of the image
- grass
  - clouds
  - sun
- information

Example 2: Creating a List Using Child Components

To generate a report that includes the following bulleted list:

- red
- green
- blue

Use the following report template:

```
[-] Report - Unnamed.rpt
   [-] Bulleted list from child components
      [ ] Text - red
      [ ] Text - green
      [ ] Text - blue
```

Creating a List Using a Cell Array

To generate the same bulleted list as in the previous example, configure a report template to call a cell array, colors:

```
[-] Report - Unnamed.rpt
   [-] Bulleted list from cell array called colors
```

Wherecolors is:

```
colors={'red','green','blue'}
```

Insert Anything into Report?

Yes. List.

Class

`rptgen.cfr_list`

See Also

`Chapter/Subsection`, `Empty Component`, `Link`, `Paragraph`, `Table`, `Text`, `Title Page`

Logical Else

Purpose

Specify an `else` condition for a `Logical If` component

Description

This component acts as an `else` when it is the child of the `Logical If` component. You can specify this component in one of the following ways:

-

  ```
  if
   then
   else
  ```

-

  ```
  if
   then
   elseif
   elseif
      .
      .
      .
   else
  ```

Properties

If component has no children, insert text: Inserts specified text into your report when the `Logical Else` component has no child components. In this case, this component acts like the `Text` component.

Insert Anything into Report?

Yes, when `if` or `elseif` statement is false.

Class

`rptgen_lo.clo_else`

See Also

`For Loop`, `Logical Elseif`, `Logical If`, `Logical Then`, `While Loop`

Purpose

Specify an `elseif` condition for the `Logical If` component

Description

This component acts as an `elseif` when it is the child of the `Logical If` component. You must specify this component as follows:

```
if
 then
 elseif
 elseif
    .
    .
    .
 else
```

Properties

- **Test expression**: Specifies a MATLAB expression to evaluate.
- **If component has no children, insert text**: Inserts the specified text into the report when the `Logical Elseif` component has no child components. In this case, this component acts like the `Text` component.

Insert Anything into Report?

Yes, when parent `if` statement is false.

Class

`rptgen_lo.clo_else_if`

See Also

`For Loop`, `Logical Else`, `Logical If`, `Logical Then`, `While Loop`

Logical If

Purpose

Specify `logical if` condition

Description

This component acts as a logical `if`; it can have the `Logical Then`, `Logical Elseif`, or `Logical Else` components as children components. This component executes its child components when the specified workspace expression is true. It displays a specified string when it has no child components. You can specify this component as follows:

-
  ```
  if
   then
  ```

-
  ```
  if
   then
   else
  ```

-
  ```
  if
   then
   elseif
   elseif
       .
       .
       .
   else
  ```

Properties

- **Test expression**: Specifies a MATLAB expression to evaluate.
- **If component has no children, insert text**: Inserts specified text into the report when the `Logical If` component has no children.

Insert Anything into Report?

Depends on specified attribute values.

Class

`rptgen_lo.clo_if`

See Also

`For Loop`, `Logical Else`, `Logical Elseif`, `Logical Then`, `While Loop`

Logical Then

Purpose

Specify a `then` condition for the `Logical If` component

Description

This component acts as a `then` when it is the child of the `Logical If` component. You can specify this component as follows:

-
  ```
  if
   then
  ```

-
  ```
  if
   then
   else
  ```

-
  ```
  if
   then
   elseif
   elseif
      .
      .
      .
   else
  ```

Attributes

If component has no children, insert text: Inserts specified text into the report when the `Logical Then` component has no children. In this case, this component acts like the `Text` component.

Insert Anything into Report?

Yes, when parent `if` statement is true.

Class

`rptgen_lo.clo_then`

See Also

`For Loop`, `Logical Else`, `Logical Elseif`, `Logical If`, `While Loop`

MATLAB Property Table

Purpose

Insert table that includes MATLAB object property name/property value pairs

Description

This component inserts a table that includes MATLAB object property name/property value pairs.

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**: Choose a type of table:
  - `Default`
  - `Blank 4x4`

  To apply the preset table, select the table and click **Apply**.

- **Split property/value cells**: Splits property name/property value pairs into separate cells. Select the **Split property/value cells** check box for the property name and property value to appear in adjacent cells. In this case, the table is in split mode; only one property name/property value pair per cell is allowed. If more than one name/property pair exists in a cell, only the first pair appears in the report; subsequent pairs are ignored.

  Clear the **Split property/value cells** check box for a given property name and property value to appear together in one cell. In this case, the table is in nonsplit mode, which supports more than one property name/property value pair. It also supports text.

  Before switching from nonsplit mode to split mode, make sure that you have only one property name/property value pair per table cell.

- **Display outer border**: Displays the outer border of the table in the generated report.

- **Table Cells**: Modifies table properties. The selection in this pane affects the available fields in the **Cell Properties** pane.

Cell Properties

Available options in the **Cell Properties** pane depend what you select for **Table Cells**. If you select `Workspace Properties`, only the **Contents** and **Show** options appear. If you select any other option, the **Lower border** and **Right border** options appear.

- **Contents**: Modifies the contents of the table cell selected in the **Table Cells** pane.
- **Show as**: Specifies the format for the contents of the table cell. Options include:
  - `Value`
  - `Property Value`
  - `PROPERTY Value`
  - `Property:  Value`
  - `PROPERTY: Value`
  - `Property - Value`
  - `PROPERTY - Value`
- **Alignment**: Specifies how to align the contents of the selected table cell in the **Table Cells** field. Options include:
  - `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 `Blank 4x4`. You can add and delete rows and add properties. To open the Edit Table dialog box, click **Edit**.

For details about using this dialog box to create custom property tables, see "Working with Property Table Components" on page 4-9.

Insert Anything into Report?

Yes. Table.

Class

`rptgen.cml_prop_table`

See Also

`Evaluate MATLAB Expression`, `Insert Variable`, `MATLAB/Toolbox Version Number`, `Variable Table`

Purpose

Insert table that shows version and release numbers and release date of MathWorks products

Description

Using the Table Filter, specify whether this component reports version information for all installed MathWorks products or just those products required for a model.

For the specified set of products, this component inserts a table showing any of these columns that you specify:

- Version number
- Release number
- Release date
- Is required for model

You can list all your MathWorks products by typing `ver` at the MATLAB command line.

Table Title

Table title: Specifies the table title. The default is `version number`.

Table Filter

Show only toolboxes required for model: When you select this option, the report shows version information for only those products required for a model or chart. By default, the report shows version information for all installed MathWorks products.

Table Columns

- **Version number**: Includes the product version number (for example, 3.4) for all installed MathWorks products or for only those products required for a model or chart.

  or

- **Release number**: Includes the MathWorks release number (for example, R2009b) for all installed MathWorks products or for only those products required for a model or chart.
- **Release date**: Includes the release date of for all installed MathWorks products or for only those products required for a model.

- **Is required for model**: Indicates "Yes" for each MathWorks product required for a model or chart.

Insert Anything into Report?

Yes. Table.

Class

`rptgen.cml_ver`

See Also

`Evaluate MATLAB Expression`, `Insert Variable`, `MATLAB Property Table`, `Variable Table`

Nest Setup File

Purpose

Allow one report template (`.rpt` file) to run inside another

Description

This component runs another report template at the point where the `Nest Setup File` component is located in the current report template.

The components of the inserted report template are stored in the current report template at the same level as the `Nest Setup File` component. Thus, inserted components have the same parent component as the `Nest Setup File` component.

Properties

- **Setup file to run**: Specifies the name of the report template to import and run. You can enter a name or use the **browse** button **(...)** to find the file.
- **Inline nested report in this report**: Inserts the nested report in the original report template where this component is located.
- **Insert link to external report**: Creates two separate reports, one using the original report template and one using the nested report template.
- **Recursion limit**: Allows you to nest a report template inside itself by setting a recursion limit in this field. The recursion limit sets a limit on the number of times the report template can run itself.
- **Nest all reports with specified file name**: Nests all reports with the same name as specified in the **Setup file to run** option.

Example

In the following example, the report template `R2.rpt` is nested in `R1.rpt`:

```
[-] Report - R1.rpt                [-] Report - R2.rpt
 [ ] Chapter                               [ ] 1
      [-] B                                [ ] 2
         [ ] Nest Setfile - R2.rpt        [-] Chapter
         [ ] C                               [ ] 4
      [ ] D                                  [ ] 5
```

The generated report is identical to the one generated by the following report template:

```
[-] Report - R1.rpt
  [ ] Chapter
  [-] B
     [ ] 1
     [ ] 2
     [-] Section 1
       [ ] 4
       [ ] 5
     [ ] C
  [ ] D
```

Components that determine their behavior from their parents, such as `Chapter/Subsection`, are affected by components in the parent report template.

Insert Anything into Report?

Yes, if the nested report template produces a report.

Class

`rptgen.crg_nest_set`

See Also

`Comment`, `Import File`, `Stop Report Generation`, `Time/Date Stamp`

Purpose

Insert paragraph text into report

Description

This component inserts a paragraph into the report. The paragraph text is taken from a child text component, or from text that you enter in the **Paragraph Text** field.

Title Options

- **No paragraph title**(default): Specifies no title for the paragraph.
- **Get title from first child** : Gets the title of the paragraph from its first child component, which should be a `Text` component.
- **Custom title**: Specifies a custom title for the paragraph.

Paragraph Text

Enter paragraph text into this field. If the `Paragraph` component has child components, the paragraph content is taken from the child components; otherwise, the `Paragraph` component inserts text from this field. If the `Paragraph` component does not have any child components and you do not enter any text into this field, nothing appears in the report.

Use the `%<VariableName>` notation in this field if you want to insert the value of a variable from the MATLAB workspace. For more details about this notation, see "%<VariableName> Notation" on page 9-68 on the `Text` component reference page.

Style

- **Bold**: Makes the text bold.
- **Italic**: Makes the text italic.
- **Underline**: Underlines the text.
- **Strikethrough**: Strikes through the text.
- **Retain spaces and carriage returns**: Formats text in the report in the same way as it is entered.
- **Show text as syntax-highlighted MATLAB code**: Displays the text as syntax-highlighted MATLAB code.
- **Color**: Specifies the color of the text.

Insert Anything into Report?

Yes, depending on child components.

Class

`rptgen.cfr_paragraph`

See Also

`Chapter/Subsection`, `Empty Component`, `Image`, `Link`, `List`, `Table`, `Text`, `Title Page`

Stop Report Generation

Purpose

Halt report generation

Description

This component acts like **Stop** during report generation. You can use this component inside an `if/then` statement by using Logical and Flow Control components to halt the report-generation process when the specified condition is `true`. When report generation halts, an XML source file is created, but not converted.

Confirmation Properties

- **Confirm before stopping generation**: Generates a confirmation dialog box before stopping report generation.
- **Confirmation question**: Specifies a confirmation question for the prompt. The default is `Stop generating the report?`
- **Halt button name**: Specifies a name for the button that stops report generation. The default is **"Halt Generation"**.
- **Continue button name**: Specifies a name for the button that continues report generation. The default is **"Continue Generation"**.

Example

This example creates a simple report that takes a snapshot of the current figure. If there is no current figure, the report generation automatically halts:

```
[-] Report - figure-report.rpt
 [-] if (isempty(get(0,'CurrentFigure')))
  [ ] Stop Generation
 [-] Figure Loop - current
  [-] Chapter - <Title from SubComponent1>
   [ ] Figure Name
   [ ] Graphics Figure Snapshot
   [ ] Figure Prop Table - Figure Properties
```

Insert Anything into Report?

No.

Stop Report Generation

Class `rptgen.crg_halt_gen`

See Also `Comment`, `Import File`, `Nest Setup File`, `Time/Date Stamp`

Purpose

Convert rectangular cell array into table and insert it into report

Description

This component converts a rectangular cell array into a table and inserts the table into the report.

Table Content

- **Workspace variable name**: Specifies the workspace variable name with which to construct the table.
- **Collapse large cells to a single description**: Consolidates large cells into one description.

Formatting Options

- **Table title**: Specifies the title of your table.
- **Cell alignment**: Options are:
  - `left`
  - `center`
  - `right`
  - `double justified`

- **Column widths**: Inputs a vector with *m* elements, where *m* equals the number of columns in the table. Column sizing is relative and normalized to page width. For example, say that you have a 2-by-3 cell array and input the following into the **Column widths** field:

  ```
  [1 2 3]
  ```

  The report output format for the cell array is such that the second column is twice the width of the first column, and the third column is three times the width of the first column. If the vector is greater than the number of columns in the table, the vector is truncated so that the number of elements equals the number of columns. If *m* is less than the number of columns in the table, the vector is padded with `1`s so that the number of elements equals the number of columns.

  If you use this field, it is recommended that you specify a width for each column. Any width not specified defaults to 1. MATLAB displays a warning when defaulting any unspecified column width to 1.

- **Table grid lines**: Displays grid lines, which create borders between fields, in the table.
- **Table spans page width (HTML only)**: Sets the table width to the width of the page on which it appears.

Header/Footer Options

Designating a row as a header or footer row causes the contents of the row to appear in boldface.

- **Number of header rows**: Specifies the number of header rows.
- **Footer list**:
  - `No footer`: Specifies no footers for the report.
  - `Last N rows are footer`: Enables you to select a footer that is different from your header.

Example

Consider the following cell array in the MATLAB workspace:

```
{'foo','bar';[3],[5]}
```

Its cell table in the report appears as follows.

foo	bar
3	5

Note that the table has no headers or footers and no title.

Insert Anything into Report?

Yes. Table.

Class

`rptgen.cfr_table`

See Also

`Chapter/Subsection`, `Empty Component`, `Image`, `Link`, `List`, `Paragraph`, `Text`, `Title Page`

Text

Purpose

Format and insert text into report

Description

This component formats and inserts text into the report. It must have the `Paragraph` component as its parent.

Properties

Text to include in report: Specifies text to include in the report.

%<VariableName> Notation

You can enter `%<VariableName>` in this field (and in any field where the text appears blue) to include the value of a variable from the base MATLAB workspace. You cannot enter more than one variable in `%<>`. If you enter an invalid variable name, the report includes the text `%<VariableName>` instead of the value of the variable.

Example

1. Enter the following text:

```
I have a %<ObjName> and it has %<NumLeaves> leaves.
The word '%<ObjName>' has %<size(ObjName)> letters.
```

2. Set `ObjName` = `"plant"` and `NumLeaves` = 3.

3. Generate the report. It looks as follows:

```
I have a plant and it has 3 leaves.
The word 'plant' has 5 letters.
```

Style

- **Bold**: Makes the text bold.
- **Italic**: Makes the text italic.
- **Underline**: Underlines the text.
- **Strikethrough**: Strikes through the text.
- **Retain spaces and carriage returns**: Formats the text in the report as you entered it.

- **Show text as syntax-highlighted MATLAB code**: Shows the text as syntax-highlighted MATLAB code.
- **Color**: Specifies the color of the text from the menu.

Insert Anything into Report?

Yes. Text.

Class

`rptgen.cfr_text`

See Also

`Chapter/Subsection`, `Empty Component`, `Image`, `Link`, `List`, `Paragraph`, `Table`, `Title Page`

Time/Date Stamp

Purpose

Insert time and date of report generation into report

Description

This component inserts the time and date of the report generation into your report as text. It must have the `Paragraph` or `Chapter/Subsection` component as its parent.

Prefix

Include text before stamp : Includes text before the time/date stamp. Specify the text in the corresponding field.

Time Stamp Properties

- **Include current time in stamp**: Inserts the current time into the time/date stamp.
- **Time display**: Specifies the appearance of the time display. Options include:
  - `12-hour`
  - `24-hour`
- **Time Separator**: Specifies a separation marker between hours, minutes, and seconds. Options include:
  - `Blank space ( )`: Formats time as `Hour Minute Second`
  - `Colon (:)`: Formats time as `Hour:Minute:Second`
  - `Period (.)`: Formats time as `Hour.Minute.Second`
  - `None ()` : Formats time as `HourMinuteSecond`
- **Include seconds in time stamp**: Displays seconds in the time/date stamp.

Date Stamp Properties

- **Include current date in stamp**: Inserts the current date in the time/date stamp.
- **Date order**: Specifies the order in which the day, month, and year appear. Options include:
  - `Day Month Year`
  - `Month Day Year`

  - `Year Month Day`
- **Date separator**: Specifies a separation marker between day, month, and year. Options include:
  - `Blank space ( )`: Displays date as `Day Month Year`
  - `Colon (:)`: Displays date as `Day:Month:Year`
  - `Slash (/)`: Displays date as `Day/Month/Year`
  - `Period (.)`: Displays date as `Day.Month.Year`
  - `None ()`: Displays date as `DayMonthYear`
- **Month display**: Specifies how the month displays. Options include:
  - `Long (December)`
  - `Short (Dec)`
  - `Numeric (12)`
- **Year display**: Specifies how the month displays. Options include:
  - `Long (2007)`
  - `Short (07)`

Preview

This pane displays the time/date stamp to appear in the report.

Insert Anything into Report?

Yes. Text.

Class

`rptgen.crg_tds`

See Also

`Comment`, `Import File`, `Nest Setup File`, `Stop Report Generation`

Title Page

Purpose

Insert title page at beginning of report

Description

This component inserts a title page at the beginning of the report. You can use it in a report template as a child of a `Chapter/Subsection` component or alone.

Properties

The text fields on this property pane support the `%<VariableName>` notation. For more details about this notation, see "%<VariableName> Notation" on page 9-68 on the `Text` component reference page.

Main Tab

Title

- **Title**: Specifies the title of the report. The title appears in large font.
- **Subtitle**: Specifies the subtitle of the report. The subtitle appears in smaller font under the title.

Options

- **Author**: Options include:
  - `Custom`(default): Specifies the author of the report.
  - `No author`: Specifies no author's name.
  - `Automatic author`: Automatically includes your user name as the author name.

  The author name appears under the subtitle, in a smaller font than the subtitle.
- **Include report creation date**: Includes the report creation date. Choose the date format in the corresponding list.
- **Include copyright holder and year**: Includes copyright holder and year information.

Image Tab

File

- **File name**: Specifies the file name of an image to appear under the subtitle, on the title page.
- **Copy to local report files directory**: Copies the image file into the folder in which the report file is located.

Display Options

- **Scaling**:
  - `Fixed size`: Specifies number and type of units.
  - `Zoom`: Specifies percentage, maximum size, and units of measure.
  - `Use image size`: Causes the image to appear in the report as the same size it does on screen.
- **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** list
- **Alignment** Options are:
  - `Auto`
  - `Right`
  - `Left`
  - `Center`

  Only `PDF` and `RTF` reports support this option.

Abstract Tab

Abstract Text: Specifies an optional abstract for the report.

Style

- **Bold**: Makes the text bold.
- **Italic**: Makes the text italic.
- **Underline**: Underlines the text.
- **Strikethrough**: Strikes through the text.

- **Retain spaces and carriage returns**: Formats the text in the generated report as you entered it.
- **Show text as syntax-highlighted MATLAB code**: Shows the text as syntax-highlighted MATLAB code.
- **Color**: Specifies the color of the text.

Legal Notice Tab

Legal Notice Text: Specifies an optional legal notice for the report.

Style

- **Bold**: Makes the text bold.
- **Italic**: Makes the text italic.
- **Underline**: Underlines the text.
- **Strikethrough**: Strikes through the text.
- **Retain spaces and carriage returns**: Formats the text in the generated report as you entered it.
- **Show text as syntax-highlighted MATLAB code**: Shows the text as syntax-highlighted MATLAB code.
- **Color**: Specifies the color of the text.

Insert Anything into Report?

Yes. Title page.

Class

`rptgen.cfr_titlepage`

See Also

`Chapter/Subsection`, `Empty Component`, `Image`, `Link`, `List`, `Paragraph`, `Table`, `Text`

Purpose

Insert table that displays all the variables in the MATLAB workspace

Description

This component inserts a table that displays all the variables in the MATLAB workspace.

Tip Find all the variables in the MATLAB workspace by typing `whos` at the command line.

Source Workspace

Read variables from:

- `Base workspace`: Reads variables from the MATLAB workspace.
- `MAT-file`: Reads variables from a binary file with a `.mat` extension. Use the `%<VariableName>` notation. For more details about this notation, see "%<VariableName> Notation" on page 9-68 on the `Text` component reference page.

Table Title

- **Table title**:
  - `Automatic (Variables from MATLAB workspace)`: Sets the table title to the name of a MATLAB variable.
  - `Custom`: Specifies a custom title.
- **Table Columns**: Options include:
  - `Variable dimensions (MxN)`: Includes the size of the variable.
  - `Variable memory bytes`: Includes the number of bytes of memory consumed by the variable.
  - `Variable class`: Includes the variable class.
  - `Variable value`: Includes the value of the variable.

Note Large variable arrays collapse to `[MxN CLASS]`. For example, if you have a 300-by-200 double array, it appears in the report as [`300x200 DOUBLE`].

Example

The following is an example of a variable table that includes size, memory bytes, and value information in the table columns.

Name	Size	Bytes	Value
`aCell`	1x2	238	{ [1 2 3 4] Speed (kph) }
`aNumber`	1x1	8	1
`aString`	1x11	22	Speed (kph)
`aStructure`	1x1	302	[struct w/ fields. Inputs, Outputs]
`aVector`	1x4	32	[1 2 3 4]

Insert Anything into Report?

Yes. Table.

Class

`rptgen.cml_whos`

See Also

`Evaluate MATLAB Expression`, `Insert Variable`, `MATLAB Property Table`, `MATLAB/Toolbox Version Number`

While Loop

Purpose

Iteratively execute child components while a specified condition is true

Description

This component iteratively executes its child components while a specified condition is true. The `While Loop` component must have at least one child component; the purpose of this component is to run its children several times. If it does not have any children, this component does not add anything to the report.

Tip Limit the number of repetitions to prevent infinite loops.

Logic Properties

- **Continue looping if this expression is true**: Specifies a string to evaluate. This string must be a valid MATLAB expression that evaluates to `1` or `0` (`true` or `false`).

  For example, if `a` = `1`, `b` = `2`, and `c` = `3`, the following command:

  ```
  d=(a>b/c)
  ```

  returns:

  ```
  d = 1
  ```

  Because `1` is greater than `b/c` (`2/3`), this expression is `true` and evaluates to `1`.

- **Limit number of loops to**: Allows you to prevent infinite loops. Use the left and right arrows to increase or decrease the number of loops.
- **Initialize with this expression**: Initializes the loop with a valid MATLAB expression.

Insert Anything into Report?

Yes, if it has a child component.

While Loop

Class `rptgen_lo.clo_while`

See Also `For Loop`, `Logical Else`, `Logical Elseif`, `Logical If`, `Logical Then`

10

Function Reference

Customization

`compwiz`	Create custom MATLAB Report Generator components

GUI

`rptlist`	Retrieve list of all report templates in the MATLAB path
`setedit`	Start the Report Explorer

Report Formatting

`rptconvert`	Convert DocBook XML files into supported document formats

Report Generation

`report`	Generate reports from report templates

11

Functions – Alphabetical List

compwiz

Purpose

Create custom MATLAB Report Generator components

Syntax

```
compwiz
compwiz (' browse')
compwiz (' v1browse')
```

Description

The Component Creation tool creates a framework for custom report components. For more information, see Chapter 5, "Creating Custom Components" in this documentation.

- `compwiz` with no arguments displays the Component Editor in the Report Explorer GUI.
- `compwiz (' browse')` displays a list of components from which to derive a new component.
- `compwiz (' v1browse')` displays a list of legacy (`v1.x`) components from which to derive a new component.

See Also

Chapter 5, "Creating Custom Components"

Purpose

Generate reports from report templates

Syntax

```
report
report (filename)
[report1, report2, ...] = report (rptfile1, rptfile2, ...)
report (...,-oOPATH,...)
report (...,-fFORMAT, ...)
```

Description

- `report` with no arguments opens the Report Explorer. For more information on the Report Explorer, see "What Is the Report Explorer?" on page 1-3
- `report (filename)` generates a report from the specified report template. When specifying the name of the report template, omit the `.rpt` file name extension. You can specify one or more report templates.
- `[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.

The `report` command also accepts the following flags, which set options for the report during the report-generation process. You can pass these flags to the `report` command anywhere among the input arguments.

- `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`)

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 template:

  ```
  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')
```

See Also

Chapter 3, "Generating Reports"

Purpose

Convert DocBook XML files into supported document formats

Syntax

```
rptconvert
rptname = rptconvert (source)
rptname = rptconvert (source, format)
rptname = rptconvert (source, format, stylesheet)
...=rptconvert(...,'-view')
...=rptconvert(...,'-quiet')
...=rptconvert(...,'-verbose')
sheetlist = rptconvert('-stylesheetlist')
sheetlist = rptconvert('-stylesheetlist',format)
FORMATLIST = rptconvert('-formatlist')
```

Description

This function converts a DocBook XML source file created by the report-generation process to a supported document format. For information about supported output formats, see "Supported Report Formats" on page 1-10.

`rptconvert` with no input arguments launches the converter GUI. When input arguments are passed to this function, `rptconvert` converts the XML document to the specified format and displays status messages to the MATLAB Command Window.

In the following commands:

- `rptname = rptconvert (source)`
- `rptname = rptconvert (source, format)`
- `rptname = rptconvert (source, format, stylesheet)`

`source` is the name of the DocBook XML file created by the report-generation process. You can specify this file name with or without its file extension.

`format` is a unique identifier code for each output format type. If you omit this argument, the XML file is converted to HTML format by default.

`stylesheet` is a unique identifier for a given stylesheet. If you omit this argument, the default stylesheet for the selected `format` is used.

You can also pass the following flags to the input arguments:

- `...=rptconvert(...,'-view')` displays the converted document.
- `...=rptconvert(...,'-quiet')` suppresses status messages.
- `...=rptconvert(...,'-verbose')` shows detailed status messages.
- `sheetlist = rptconvert('-stylesheetlist')` returns a two-column cell array. The first column of this array includes valid stylesheet identifiers. The second column includes descriptions of each stylesheet.
- `sheetlist = rptconvert('-stylesheetlist',format)` returns an array like that returned by `sheetlist = rptconvert('-stylesheetlist')`. The first column of this array includes stylesheet identifiers for the specified `format`.
- `FORMATLIST = rptconvert('-formatlist')` returns a two-column cell array. The first column of this array includes valid `format` values, the second column includes descriptions of each format.

Examples

Retrieve a list of available `HTML` stylesheets:

```
rptconvert('-stylesheetlist','html')
```

See Also

"Converting XML Documents" on page 3-11

Purpose Retrieve list of all report templates in the MATLAB path

Syntax `rptlist`

Description `rptlist` with no arguments opens the Report Explorer, which lists available report templates in the MATLAB path.

See Also `setedit`

Purpose Start the Report Explorer

Syntax

```
setedit (filename)
```

Description `setedit (filename)` opens the Report Explorer and loads the report template named `filename`. If a file with the specified name does not exist, Report Explorer opens an empty report template with that name.

See Also `rptlist`, “What Is the Report Explorer?” on page 1-3

Examples

Use this list to find examples in the documentation.

Generating Reports from MATLAB Files

Working with Components

Customizing Components

Customizing Stylesheets

Generating Reports from XML files

Index