RML for Idiots

An easy introduction to
Report Markup Language

ReportLab Europe Ltd.
Thornton House
Thornton Road
Wimbledon
London SW19 4NG, UK

Introduction

About RML

• Report Markup Language is an XML-style language for describing the layout of documents.
• You define and manipulate any aspect of a document, including the content and the styling, by using tags. Many of the tags are similar to HTML.
• The PDF document output is generated from the RML using the python module 'rml2pdf'.

About this document

This document is designed to get you up and running with RML quickly. It demonstrates the basics of laying out simple documents and formatting the contents, with examples throughout. The idea is to show you how to do the stuff you are most likely to want to do, without getting bogged down in complexities; so each example presents a basic concept but doesn't go into too much detail about the niceties - for that, you can use the User Guide and the test documents. Work through this guide from start to finish, and you will cover all the major concepts, and end up with some nice looking pdf output.

Running the examples

You need a text editor and a working ReportLab PLUS setup - if you don't already have this, follow the installation instructions on our website, or ask a Reportlab employee. At the command line, create a directory where you want to keep your code, and cd into it. Cut and paste each example from this document into a text file, save it in the directory you just created, then run it through rml2pdf.pyc like this:

UNIX:

```
user@computer:~/rml-for-idiots$ python ~/path/to/rml2pdf.pyc example_xx.rml
```

WINDOWS:

```
C:\rml-for-idiots> C:\path\to\rml2pdf.pyc example_xx.rml
```

The result should be a document in the same directory called 'example_xx.pdf'.

The python module rml2pdf.pyc can also be used inside python scripts, like any other module. Import the module, read your RML into a string, and pass it into the 'go' method as follows:

```
>>> from rlextra.rml2pdf import rml2pdf
>>> rml=open('inputfile.rml','r').read()
>>> rml2pdf.go(rml,'outputfile.pdf')
```

A pdf named 'outputfile.pdf' will be created in the current directory.

Reference Help

• Use the rml2pdf-userguide.pdf located in rlextra/rml2pdf/doc to expand on what is presented here.
• Browse the test cases in rlextra/rml2pdf/test/ for a huge range of examples of RML functionality.
• The ultimate reference document for the RML tags is the Document Type Definition; located in 'rlextra/rml2pdf/rml.dtd'. This tells you about all the available RML tags, and their required and optional parameters. See the next section for more details.

Using the DTD

A 'Document Type Definition' is the standard way to define an XML dialect. It formally sets out the building blocks of an XML document, with a list of legal elements and attributes. You can find plenty of information on

the web about how to read a DTD - a good place to start is http://www.w3schools.com/DTD/default.asp

Every tag available in RML is defined and described in rlextra/rml2pdf/rml.dtd - so if you're unsure about a tag or its parameters, search in this document for the tag name and you will see it's definition. There's more information about how to read the DTD in the FAQ at the end of this document.

Essential concepts

The structure of a document.

When designing a document in RML, the main concept to grasp is the difference between the static elements on the page, and 'flowables'.

The component parts that make up a document, ie. text, pictures, or primitives (like lines, rectangles and circles) can be either static or flowable, depending on whereabouts in the RML document they are described.

A static element is fixed into position on a page. You specify the x and y coordinates and the element appears at that position.

With flowables, however, you specify the position of a 'frame', which can be thought of as a box which contains flowable elements. The elements then flow into the frame until that frame is full. Then they start filling up the next frame. So the frames are static elements, but what the frames contain is 'flowable'.

The other main element of a document is the styling: the fonts, line weights and colours, etc..

These three elements are defined in the three main sections of a basic RML document like this:

```
<template>
    this section contains the STATIC elements of the document, and the layout of the
    frames on each page.
</template>
<stylesheet>
    this section contains the STYLE information for formatting text and tables.
</stylesheet>
<story>
    this section contains the FLOWABLE elements of the document.  This is
    usually where most of the text goes.  This content will flow into
    the frames specified in the <template> section.
</story>
```

As an example of the above concepts, imagine the front page of a newspaper. In terms of RML, the masthead (eg. "The Times") would be a static, fixed element - it's always the same text in the same place. There might be a solid line under the top banner, along with the price and the date, all of which would be fixed position elements. Most of the centre of the page would then be divided into frames containing text in various styles: headlines, subsection headings, and plain text. The text of these stories would 'fill up' each of these frames from top to bottom, in a series of paragraphs. Finally there might be an advert in a fixed position at the bottom of the page.

So, the <template> section would contain details about the masthead and the position of the frames, the <stylesheet> section would define the fonts used for headlines and body text, and the actual text of the news would be in the <story> section. The news would flow from top to bottom, left to right, into each frame on the page.

This is a simplified version of the truth, and there are exceptions to all this, but that comes later...

Common Conventions in RML

1. Colours can be specified in three ways in RML: as html-style 'named' colours (eg. "red"); as rgb values (eg. "#004A8D") or as CMYK (eg. "#ff99001f")

2. X & Y co-ordinates generally refer to the bottom left corner of an element.

3. Measurements default to points, but can be millimetres(mm), centimetres(cm), or inches(in). They can be mixed and matched as you like eg(x="5" y="10mm" height="2cm" width="1in").

Lesson 1: Hello World : One simple frame on a page

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_01.pdf">

    <template>
        <!--this section contains elements of the document -->
        <!--which are FIXED into position.                 -->
        <pageTemplate id="main">
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
        </pageTemplate>
    </template>

    <stylesheet>
        <!--this section contains the STYLE information for            -->
        <!--the document, but there isn't any yet.  The tags still     -->
        <!--have to be present, however, or the document won't compile.-->
    </stylesheet>

    <story>
            <!--this section contains the FLOWABLE elements of the -->
            <!--document.  These elements will fill up the frames  -->
            <!--defined in the <template> section above.           -->
        <para>
            Welcome to RML!
        </para>
        <para>
             This is the "story". This is the part of the RML document where
             your text is placed.
        </para>
        <para>
             It should be enclosed in "para" and "/para" tags to turn it into
             paragraphs.
        </para>
    </story>

</document>
```

Run the above example as outlined in the section above called 'Running the examples'. Open the output file ('example_01.pdf') and you will see a basic document with some text in the middle of the page. Note the following about the RML:

The <template> section is where the layout is defined. In this case it defines one <frame> by giving it an id ("main"), followed by the x and y coordinates of the bottom left corner of the frame, and then its width and height.

All these values are specified in points, which is the default unit of measurement in RML, but you can mix and match mm, cm, inches and points like this: <x="10mm" y="2cm" width="5in" height="50">.

This frame is the fixed 'box' into which the text from the <story> section will flow. Remember, coordinates in RML refer to the bottom left corner of the element. However, the text starts flowing from the top of the frame, so if you change the height of the frame, the text will start further up the page.

As it says in the comments, the <stylesheet> section is empty. It has to be there because the document won't compile without it.

The <story> section contains three paragraphs, each of which is contained inside <para> tags. Note that the whitespace is ignored in the PDF output.

So for a basic bit of text on a page, you define a frame in a position you like, and then the text in the <story> section fills up the frame.

Lesson 2: Two frames, text flowing between them.

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_02.pdf">

    <template showBoundary="1"> <!--Debugging is now turned on, frame outlines -->
                                <!--will appear on the page                    -->
        <pageTemplate id="main">
            <!-- two frames are defined here: -->
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>
        </pageTemplate>
    </template>

    <stylesheet>
        <!-- still empty...-->
    </stylesheet>

    <story>
        <para>
             Welcome to RML.
        </para>
        <para>
             This is the part of the RML document where
             your text is placed.
        </para>
        <para>
             It should be enclosed in &lt;para&gt; and &lt;/para&gt; tags to turn
             it into paragraphs.
        </para>
        <para>
             You should now be able to see borders around the two frames on this
             page, because showBoundary="1" in the template section at the top.
             Set it to 0 and the black borders will disappear.
        </para>
        <para>
             This is an extra paragraph which is a bit longer.  It is supposed to
             fill up the rest of the current frame, and demonstrates what happens
             when a frame fills up with text. It should start flowing into the next
             frame. blah blah waffle waffle lorem ipsum etc etc
        </para>
    </story>
</document>
```

Lesson number two demonstrates two new features.

Firstly, the <template> tag now has an attribute, showBoundary, set to 1. This draws an outline around the frames on the page, which is useful for debugging.

Secondly, we have added another frame tag which defines a second frame, next to the first. The extra <para> tag in the <story> section contains more text, which overflows from the first frame into the second.

So now you see how text flows from one frame to the next.

Lesson 3: Adding text in fixed positions

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_03.pdf">

    <template showBoundary="0">
        <pageTemplate id="main">
            <pageGraphics>
                <!-- Use drawString to place a fixed title on the page -->
                <drawString x="100" y="700"> Hello!</drawString>
            </pageGraphics>
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>
        </pageTemplate>
    </template>

    <stylesheet>
    </stylesheet>

    <story>
        <para>
             Welcome to RML.
        </para>
        <para>
             This is the part of the RML document where
             your text is placed.
        </para>
        <para>
             It should be enclosed in &lt;para&gt; and &lt;/para&gt; tags to turn
             it into paragraphs.
        </para>
        <para>
             You can no longer see borders around the two frames on this
             page, because showBoundary="0" in the template section at the top.
             Set it to 1 and the black borders will reappear.
        </para>
        <para>
             This is an extra paragraph which is a bit longer.  It is supposed to
             fill up the rest of the current frame, and demonstrates what happens
             when a frame fills up with text. It should start flowing into the next
             frame. blah blah waffle waffle lorem ipsum etc etc
        </para>
    </story>
</document>
```

The <pageTemplate> tag now contains a new section, <pageGraphics>, inside which we have used the <drawString> tag to draw some text in a fixed position on the page. Again, the coordinates are defined in points, and represent the bottom left corner of the text string itself. The <pageGraphics> tag is the place where you define the directly drawn, fixed elements, which can be lines or circles, text or images.

Now you have seen the two ways of laying out text on the page: flowing in frames, or fixed in position in the <pageTemplate>. But they look rather plain, so let's see how to style them.

Lesson 4: First steps in styling fixed and flowing text

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_04.pdf">

    <template>
        <pageTemplate id="main">
            <pageGraphics>
                <fill color="red"/>
                <setFont name="Times-Roman" size="24"/>
                <drawString x="100" y="700">
                    Different types of Text Styling
                </drawString>
            </pageGraphics>
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>
        </pageTemplate>
    </template>

    <stylesheet>
        <paraStyle name="textstyle1" fontName="Helvetica" fontSize="9" leading="10"/>
    </stylesheet>

    <story>
        <para>
             Welcome to RML.  This paragraph has not been styled.
        </para>
        <para style="textstyle1">
            This paragraph HAS been styled, using a style defined in the
            &lt;stylesheet&gt; section.
        </para>
        <para style="textstyle1" fontSize="12">
            This paragraph uses the same style as the first paragraph, but the
            fontsize has been overridden.  paraStyles can inherit from each other.
            You can also use html-style <b>bold</b> and <i>italic</i> tags.
        </para>
        <para textColor="blue">
             This paragraph has been styled, but not using the stylesheet.
             It just uses style attributes inside it's own &lt;para&gt; tag.
        </para>
    </story>
</document>
```

The new tags in the <pageGraphics> section demonstrate how to change the font and its colour for fixed elements, using <setFont> and <fill> tags. The code demonstrates an important new concept: the pageGraphics section acts like a 'state machine' when you start changing the style of the text. This means that you set the 'pen' to draw in a particular way, and it stays in that state until you change it.

The other new section is the <paraStyle> tag in the <stylesheet> section. Styles are defined here, and then used in the <para> tags in the <story> section. Any element of a <paraStyle> tag can be overridden inside the <para> tag itself. You can also style a paragraph by just adding to it's <para> tag, as in paragraph four.

Note also that although the style has been changed in the <pageGraphics> section, the changes we made to the font there do not affect the style of the text in the story, which begins, unstyled, with the default font and text size, until you change it.

Now you know how to style fixed and flowable text.

Lesson 5: Inserting images

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_05.pdf">

    <template showBoundary="1">
        <pageTemplate id="main">
            <pageGraphics>
                <image file="files/logo_bg.gif" x="300" y="700"/>
                <fill color="red"/>
                <setFont name="Times-Roman" size="24"/>
                <drawString x="100" y="700"> Hello!</drawString>
            </pageGraphics>
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>

        </pageTemplate>

    </template>

    <stylesheet>
    </stylesheet>

    <story>
        <para>
            This is a paragraph above an illustration.
        </para>

        <!-- The <illustration> tag below defines a box (with a border turned -->
        <!-- on, for visibility)  inside which an image is placed.            -->
        <illustration width="50" height="100" borderStrokeWidth="1"
                      borderStrokeColor="black">
        <!-- x and y coordinates are from the bottom left corner              -->
        <!-- of the containing illustration                                   -->
            <image file="files/rml2pdf.gif" x="10" y="10"/>
        </illustration>
        <para>
            This paragraph is below the illustration.  Note that if you
            change the height and width of the illustration, the text
            in the first paragraph may become hidden by the graphic image,
            which overflows the borders of the illustration in which it is
            contained.
        </para>
    </story>
</document>
```

You will frequently want to include images in your documents. RML can use images in most popular formats, including EPS and PDF. The images can be shrunk and expanded, and the aspect ratio can be fixed, using various attributes of the tags - see the <image> tag in the RML DTD for a list of these.

Lesson 5 shows two images. The ReportLab logo is added to the <pageGraphics> section with an x and y coordinate, and so is fixed at the top right of the page.

The second logo, however, is included in the story section as a flowable element. In order for it to be treated as a flowable, the image has to be surrounded by an <illustration> tag, and this is also the case when drawing graphics primitives like lines and circles.

The <illustration> tag is like another <pageGraphics> section, but specified as a 'flowable'; i.e the entire contents of the <illustration> tag are one flowable element in the <story>. This means that the positions of the graphics specified inside it are relative to the bottom left corner of the illustration itself, not the whole page.

One thing to note about this example is that the image can be bigger than the illustration containing it. The illustration is narrower than the image inside it (the border of the illustration is visible in the background) but the image overflows the boundaries. If you change the sample so that the height is also smaller than that of the image, you will see that the text in the paragraph above the picture disappears. That is because it is now hidden behind the image.

Now you have seen how to include images: fixed in the <pageTemplate>, or included as a flowable in the story.

Lesson 6: Drawing lines and shapes

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_06.pdf">

    <template showBoundary="1">
        <pageTemplate id="main">
            <pageGraphics>
                <fill color="red"/>
                <stroke color="purple"/>
                <lines> 50 50 100 100</lines>
                <lineMode width="20"/>
                <stroke color="blue"/>
                <circle x="150" y="650" radius="25" fill="yes" stroke="yes"/>
                <lineMode width="5"/>
                <stroke color="green"/>
                <rect x="300" y="625" width="25" height="60" fill="yes" stroke="yes"/>
            </pageGraphics>
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>

        </pageTemplate>

    </template>

    <stylesheet>
    </stylesheet>

    <story>
        <para>
            The two illustrations above, and the diagonal line at the bottom of
            the page, are fixed position graphics drawn in the pageGraphics
            section of the pageTemplate.  Note that rml operates like a 'state
            machine': you set the state of the pen, and it remains in that
            state until you change it.  The two shapes both have very fat stroke
            settings.
        </para>
        <para>
            However, the black square below is defined inside an illustration tag
            in the story.  The 'pen state' set in the pageTemplate doesn't
            apply here.
        </para>
        <!-- The <illustration> tag below defines a box -->
        <!-- inside which a rectangle is drawn.         -->
        <illustration width="50" height="100">
            <!-- note that we didn't define a pen or stroke colour in the <story>
            yet, so they have defaulted to black-->
            <rect x="0" y="0" width="25" height="60" fill="yes" stroke="yes"/>
        </illustration>
        <para>
            This paragraph is below the illustration.  Try increasing and
            decreasing the height of the illustration tag above, and see
            how it affects the flow of the elements in the story.
        </para>
    </story>
</document>
```

This example introduces the drawing of graphics primitives like lines, circles and squares.

Several of the concepts already introduced apply here. Firstly, as with included images, graphics primitives can be either fixed into position in the <pageTemplate> section, or included in the <story> as a flowable element by wrapping them in the <illustration> tag. Secondly, graphics primitives have the same styling principles as with text - you set the colour, line style etc., and the pen stays like that until you change it. Like text state, they are also set seperately within the <story> and the <pageTemplate>.

The sample code draws two shapes at the top of the page and a diagonal line at the bottom. These are defined in the <pageGraphics> section. Shapes use the concept of 'stroke' and 'fill' - the stroke is the outline of the shape and the fill is the area within the stroke. Stroke and fill are switched on and off using the parameters in the shape tags (eg fill="yes"), and the colour is set using the <stroke> and <fill> tags.

The <lines> tag uses four values which represent the x and y co-ordinates of the start and end of the line.

The black square within the <story> has to be surrounded by an <illustration> tag, and it is then treated as a flowable. As with the image in the <story> in Lesson 5, the <illustration> can be smaller than the graphic it contains, and this will have an effect on the flow of the elements in the frame.

Lesson 7: Tables

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_07.pdf">

    <template showBoundary="1">
        <pageTemplate id="main">

            <pageGraphics>
                <place x="300" y="500" width="250" height="250">
                    <blockTable>
                        <tr>
                            <td>A block </td><td>table</td>
                        </tr>
                        <tr>
                            <td>inside </td><td>a place tag.</td>
                        </tr>
                    </blockTable>
                </place>
            </pageGraphics>

            <frame id="first" x1="100" y1="400" width="150" height="200"/>

        </pageTemplate>

    </template>

    <stylesheet>
    </stylesheet>

    <story>

        <blockTable>
            <tr>
                <td>A block</td>
                <td>table</td>
            </tr>
            <tr>
                <td>inside </td>
                <td>the story tag.  This is a long string of text
                    which won't wrap by default.</td>
            </tr>
        </blockTable>

    </story>
</document>
```

Tables are drawn in RML using the <blockTable> tag. They are very similar to HTML tables.

An important thing to note is that by default, the text in a cell will not wrap. The output from the example above probably doesn't look like you'd expect: that's because the table in the <story> has a long string of text, and the entire table is centred inside the containing frame. To make the table fit inside the frame, you have to set the width of the columns in the <blockTable> tag (see lesson 10).

This example also introduces the <place> tag. This tag allows you to insert a 'flowable' section inside the pageTemplate section, (it is the converse of the <illustration> tag, which allows you to include static elements in the 'flowable' section). To use a table, or other flowable element like a <para>, inside the pageTemplate section,

wrap it inside a <place> tag; the contents of the <place> tag are then treated like a single block inside which elements will be 'flowable'.

Now you know the basics of tables and how to place them.

Lesson 8: Table Styles - pt. 1: the basics

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_08.pdf">

    <template showBoundary="1">
        <pageTemplate id="main">

            <pageGraphics>
                <place x="10" y="500" width="600" height="250">
                    <blockTable style="blocktablestyle1">
                        <tr>
                            <td>This table</td><td>is defined</td>
                        </tr>
                        <tr>
                            <td>inside a place tag in the graphics section,</td>
                            <td>
                               <para style="textstyle1">
                                   and can be styled using blockTableStyles
                                   and para tags.
                               </para>
                            </td>
                        </tr>
                    </blockTable>
                </place>
            </pageGraphics>

            <frame id="first" x1="100" y1="400" width="150" height="200"/>

        </pageTemplate>

    </template>

    <stylesheet>
        <!-- A blockTableStyle tag for styling a table-->
        <blockTableStyle id="blocktablestyle1">
            <blockFont name="Courier-Bold"/>
            <blockFont name="Helvetica-BoldOblique" size="8"/>
            <blockTextColor colorName="green"/>
        </blockTableStyle>
        <paraStyle name="textstyle1"
                   fontName="Helvetica"
                   fontSize="9"
                   textColor="blue"/>
    </stylesheet>

    <story>

        <!-- a table styled using only blockTableStyle>-->
        <blockTable>
            <tr>
                <td><para textColor="green">A block</para></td>
                <td>table</td>
            </tr>
            <tr>
                <td>inside </td>
                <td>
```

```
                    the story tag.  This is a long string of text
                    which won't wrap by default.
                </td>
            </tr>
        </blockTable>

        <!-- a table styled using both blockTableStyle from -->
        <!-- the stylesheet, and parastyles in table cells. -->
        <blockTable style="blocktablestyle1">
            <tr>
                <td><para textColor="red">This is </para></td>
                <td>the same table</td>
            </tr>
            <tr>
                <td>but with</td>
                <td>
                    <para style="textstyle1">
                        &lt;para&gt; tags inside the table cells, so the
                        long string of text is wrapped!
                    </para>
                </td>
            </tr>
        </blockTable>

    </story>
</document>
```

By default, as in example 7, tables are unstyled with invisible borders, and text doesn't wrap. This is the default because it can be rendered very fast. However, styles can be applied to tables using the <blockTableStyle> tag in the <stylesheet> section.

You can also style the text in individual cells using <para> tags inside the <td> tags.

In this example, the table at the top right is from the <pageGraphics> section. Because it is inside a set of <place> tags, it can use styles from the stylesheet in exactly the same way as a flowable element in the <story>.

The second table is a default, unstyled table. Note that it's centred inside it's containing <frame> and the text doesn't wrap inside the cells. It uses the default PDF styling.

The third table demonstrates the use of the <blockTableStyle> tag. Again, it's centred inside it's containing <frame>, and the text doesn't wrap inside the cells; however, the text is styled according to "blocktablestyle1" which is defined in the stylesheet.

The third table also demonstrates how you can mix blockTableStyles and <para> tags. The <para> tag in the bottom right cell causes the text to wrap. You can use <paraStyle> tags from the stylesheet and override their properties in exactly the same way you would outside a table.

Lesson 9: Table Styles - pt. 2: styling blocks of cells

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_09.pdf">

    <template showBoundary="1">
        <pageTemplate id="main">

            <pageGraphics>

            </pageGraphics>

            <frame id="first" x1="200" y1="400" width="150" height="200"/>

        </pageTemplate>

    </template>

    <stylesheet>
        <blockTableStyle id="blocktablestyle3">
            <!-- This sets a font for the second row -->
            <blockFont name="Helvetica-BoldOblique" size="16" start="0,1" stop="2,1"/>

            <!-- This sets a textColor for all the text in the table ...-->
            <blockTextColor colorName="purple"/>
            <!-- ...and this overrides it for the second column -->
            <!-- using -1 to refer to the last item.            -->
            <blockTextColor colorName="black" start="1,0" stop="1,-1"/>
            <!-- Use of linestyles for left, right, top and bottom gridlines -->
            <lineStyle kind="LINEBELOW" colorName="red" start="0,3" stop="-1,3" thickness="2"/>
            <lineStyle kind="LINEAFTER" colorName="maroon" start="0,3" stop="-1,3" thickness="2"/>
            <lineStyle kind="LINEBEFORE" colorName="black" start="0,3" stop="-1,3" thickness="2"/>
            <lineStyle kind="LINEABOVE" colorName="yellow" start="0,3" stop="-1,3" thickness="2"/>
            <!-- Use of grid lineStyle, for 'all in one' formatting-->
            <lineStyle kind="GRID" colorName="green" thickness="1" start="0,0" stop="-1,0" />
        </blockTableStyle>
    </stylesheet>

    <story>

        <blockTable style="blocktablestyle3">
            <tr>
                <td>This is </td>
                <td>a table with</td>
                <td>three columns</td>
            </tr>
            <tr>
                <td>demonstrating how</td>
                <td>to style each cell</td>
                <td>differently.</td>
            </tr>
            <tr>
                <td>Have a look </td>
                <td>at blocktablestyle3 </td>
                <td>to see how it works.</td>
            </tr>
            <tr>
```

```
                <td>Use line styles for different</td>
                <td>table borders; note that </td>
                <td>padding is on by default.</td>
            </tr>
        </blockTable>

    </story>
</document>
```

This example demonstrates the use of start and stop values in the <blockTableStyle> element tags. These can be used to specify particular 'blocks' of rows or columns to which you want to apply a style.

The 'start' and 'stop' tags are used to describe the beginning and end of 'blocks' of cells within the table, where the style should be applied, using an (column,row) notation with the numbering starting from zero. So, for example, the top left-hand cell is referred to as (0,0), and (2,1) refers to the third column on the second row. Negative numbers can also be used to count cells backwards from the end of the last cell, so (-1,-1) always refers to the bottom right hand cell, and e.g. (-1,1) refers to the last column on the second row.

(0,0)	(1,0)	(2,0)
(0,1)	(1,1)	(2,1)
(0,2)	(1,2)	(2,2)

So, to apply a style to the second row of the above table, you would specify start="0,1" stop="2,1".

Another important point to note about tables is that cell padding is on by default. You have to explicitly set it to zero using blockTableStyles -otherwise, it is set to 5 points in all directions. This can be a source of confusion when using the start/stop notation, because if you set padding for a particular range of cells, it will revert to the default values outside of that range - not go to zero as you might expect.

This example also shows how to apply styles to the gridlines, using lineStyles.

Lesson 10: Table Styles - pt. 3: fixed columns and rows

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_10.pdf">

    <template showBoundary="1">
        <pageTemplate id="main">

            <pageGraphics>
            </pageGraphics>
            <frame id="first" x1="200" y1="400" width="150" height="400"/>

        </pageTemplate>

    </template>

    <stylesheet>
        <blockTableStyle id="blocktablestyle4">
            <!-- show a grid: this also comes in handy for debugging your tables.-->
            <lineStyle kind="GRID" colorName="green" thickness="1" start="0,0" stop="-1,-1" />
        </blockTableStyle>
        <paraStyle name="textstyle1" fontName="Helvetica" fontSize="9" textColor="blue"/>
    </stylesheet>

    <story>

        <blockTable style="blocktablestyle4">
            <tr>
                <td>This is </td>
                <td>a table with</td>
                <td>three columns</td>
            </tr>
            <tr>
                <td>demonstrating how automatic </td>
                <td>cell sizes </td>
                <td>work</td>
            </tr>
            <tr>
                <td>Have a look </td>
                <td>at blocktablestyle4</td>
                <td><para style="textstyle1">to see how it works. This is yet more long text to
                demonstrate wrapping etc. And supercalifragilistic is a long word</para></td>
            </tr>
        </blockTable>
        <spacer length="1cm"/>
        <para>Below is the same table, with fixed row heights and column widths.</para>
        <spacer length="1cm"/>

        <blockTable style="blocktablestyle4" rowHeights="2cm,2cm,2cm" colWidths="2cm,2cm,2cm">
            <tr>
                <td>This is </td>
                <td>another table </td>
                <td>with three columns</td>
            </tr>
            <tr>
                <td>demonstrating how </td>
                <td>rowHeight and colWidth</td>
```

```
                <td>affect the cells.</td>
            </tr>
            <tr>
                <td>Have a look </td>
                <td>at lesson 10</td>
                <td><para style="textstyle1">to see how it works. This is yet more long text to
                demonstrate wrapping etc. And supercalifragilistic is a long word</para></td>
            </tr>
        </blockTable>

    </story>
</document>
```

Now you know how to add a table, and style the cell contents and gridlines. But what about column and row sizes? Up until now, these have been automatic. If you dont use any <para> tags in any of the cells in a column, then the width of the column will expand to fit the longest piece of text in the column. If any cell in a column does have a <para> tag, (which causes text in the cell to wrap), then the column will expand to the width of whichever is longer - the longest piece of text which isn't in a <para> tag, or the longest word in a column with <para> tags.

You can also fix the row heights and column widths using rowHeights and colWidths elements in the <blockTable> tag. You will get an error if you add more rows or columns than are declared. Text in fixed rows and columns will simply overflow if it doesn't fit.

Lesson 11: Controlling the flow

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_11.pdf">

    <template showBoundary="1">
        <pageTemplate id="Layout1">
            <!-- three frames are defined here: -->
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>
            <frame id="third" x1="100" y1="100" width="150" height="200"/>
            <frame id="fourth" x1="300" y1="100" width="150" height="200"/>
        </pageTemplate>

    </template>

    <stylesheet>
        <!-- empty...-->
    </stylesheet>

    <story>

        <nextFrame name="second"/>

        <para>
             ..this text will appear in the second frame, even though the first frame isn't full yet.
        </para>
        <para>
            It also appears first in the RML file, before the text in the frame on the left.
        </para>
        <para>
             Don't forget that you would normally see output without borders around the frames.
             This has been set to appear just to clarify where the frames are.
        </para>
        <para>
            When I use the &lt;nextFrame&gt; tag without specifying a frame name....
        </para>

        <nextFrame/>

        <para>
            ...we just move into the next frame listed in the &lt;pageTemplate&gt; section.
        </para>

        <nextFrame name="first"/>

        <para>
             Welcome to RML.
        </para>
        <para>
             This is a paragraph of 'flowing' text.
        </para>
        <para>
             I've used the &lt;nextFrame&gt; tag to force this text to appear in this frame, by
             specifying the frame id.
        </para>
```

```
        <!-- use <keepInFrame> to force text to stay in a particular frame. -->
        <!-- In this case, it's set to shrink the text to fit. -->
        <keepInFrame frame="fourth" onOverflow="shrink">
            <para>
                 This frame demonstrates how to force text to stay inside a frame.  Use the
                 &lt;keepInFrame&gt; tag to either shrink the text to fit, truncate it,
                 overflow the frame, or raise an error.
            </para>
            <para>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
                tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
                quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
                consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
                cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
                non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
            </para>
            <para>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
                tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
                quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
                consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
                cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
                non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
            </para>
        </keepInFrame>

    </story>
</document>
```

You can force flowable text to flow into particular frames as you wish. Use the <nextFrame> tag, in between your paragraphs, to force the text to move into the next frame. This doesn't have to be the next frame listed in the <pageTemplate> ; use the name="" element to specify the id of the frame you want to use.

You can also force text to stay inside a particular frame - and then shrink it to fit, let it overflow, truncate it, or raise an error (to see how to do each one, look up the tag in the DTD). Use <keepInFrame> tags to surround the relevant text.

Lesson 12: Different pages with different layouts

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="example_12.pdf">

    <template showBoundary="1">
        <pageTemplate id="Layout1">
            <pageGraphics>
                <image file="files/logo_bg.gif" x="300" y="100"/>
                <fill color="red"/>
                <setFont name="Helvetica-Bold" size="24"/>
                <drawString x="100" y="100"> This is Layout #1</drawString>
            </pageGraphics>
            <!-- two frames: -->
            <frame id="first" x1="100" y1="400" width="150" height="200"/>
            <frame id="second" x1="300" y1="400" width="150" height="200"/>
        </pageTemplate>
        <pageTemplate id="Layout2">
            <pageGraphics>
                <image file="files/logo_bg.gif" x="300" y="700"/>
                <fill color="red"/>
                <setFont name="Helvetica-Bold" size="24"/>
                <drawString x="100" y="700"> This is Layout #2</drawString>
            </pageGraphics>
            <!-- two frames: -->
            <frame id="first" x1="100" y1="100" width="150" height="200"/>
            <frame id="second" x1="500" y1="100" width="150" height="200"/>
        </pageTemplate>
    </template>

    <stylesheet>
        <!-- empty...-->
    </stylesheet>

    <story firstPageTemplate="Layout1">
        <para>
             Welcome to RML.
        </para>
        <para>
             This is page layout number 1.
        </para>
        <para>
            Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
            tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
            quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
            consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
            cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
            non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."
        </para>
        <para>
            Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
            tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
            quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
            consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
        </para>
        <para>
             We will now set the next page template to layout 2, and use nextFrame to force
```

```
            text into the next frame, thereby creating page 2.
        </para>
        <setNextTemplate name="Layout2"/>
        <nextFrame/>

        <para>
            In this example, there is more than one frame in the template for page one.  If there
            hadn't been enough text to fill up all the frames, we might have needed more than
            one &lt;nextFrame&gt; tag to push the text onto a new page.  That's because the
            &lt;setNextTemplate&gt; doesn't actually start using the next template immediately;
            you have to either fill up the frames with text, or specify enough &lt;nextFrame&gt;
            tags to account for any empty frames, before a new page will be created.
        </para>

    </story>
</document>
```

You can easily use different layouts for different pages in your document. Declare each different layout as a <pageTemplate> with an id in the <template> section. Normally, the flowables will fill up the frames on a page and then flow into a new page with the same layout. However, you can set the layout of the next page using the <setNextTemplate name="templateID"/> tag. This will ensure that once the current page is full, the next page will have the layout specified.

Lesson 13: Putting it all together - "The Daily Planet"

```
<!DOCTYPE document SYSTEM "rml.dtd">
<document filename="planet.pdf">

    <docinit>
        <registerTTFont faceName="ClassizismAntiqua" fileName="files/ClassizismAntiquaBook.ttf"/>
    </docinit>
    <template showBoundary="0">
        <pageTemplate id="frontpage">
            <pageGraphics>
                <setFont name="ClassizismAntiqua" size="48"/>
                <drawString x="125" y="700"> The Daily Planet</drawString>
                <lineMode width="5"/>
                <lines> 50 675 545 675</lines>
                <lines> 50 775 545 775</lines>
                <lines> 50 25 545 25</lines>
                <lineMode width="1"/>
                <lines> 178 30 178 340</lines>
                <setFont name="ClassizismAntiqua" size="12"/>
                <drawString x="50" y="650">$1.50</drawString>
                <drawString x="225" y="650">13 December 2003</drawString>
                <drawString x="415" y="650"> Quality News since 1972</drawString>
                <setFont name="ClassizismAntiqua" size="8"/>
                <drawString x="445" y="10"> Published in Metropolis City</drawString>
            </pageGraphics>
            <frame id="first" x1="50" y1="340" width="250" height="275"/>
            <frame id="second" x1="50" y1="40" width="125" height="300"/>
            <frame id="third" x1="180" y1="40" width="125" height="300"/>
        </pageTemplate>
            <pageTemplate id="page2">
            <pageGraphics>
                <lineMode width="5"/>
                <lines> 50 770 545 770</lines>
                <lines> 50 25 545 25</lines>
                <lineMode width="1"/>
                <lines> 178 30 178 750</lines>
                <setFont name="ClassizismAntiqua" size="8"/>
                <drawString x="50" y="775">13 December 2003</drawString>
                <drawString x="475" y="775">The Daily Planet</drawString>
                <drawString x="50" y="10">Page 2</drawString>
            </pageGraphics>
            <frame id="first" x1="50" y1="20" width="125" height="740"/>

        </pageTemplate>

    </template>

    <stylesheet>
       <paraStyle name="headline" alignment="LEFT" fontSize="40"/>
       <paraStyle name="strapline" alignment="LEFT"/>
       <paraStyle name="normal" alignment="LEFT" spaceBefore="5"/>
       <paraStyle name="caption" alignment="LEFT" fontSize="8"/>
    </stylesheet>

    <story>
        <para style="headline" leading="40">
```

```
            REPORTLAB RELEASE MARKUP LANGUAGE!
        </para>

        <para style="strapline">
            <b>LONDON - </b>Reportlab Inc, the software company, today announced the release of
            a new markup language promising to revolutionise the production of documents
            across the known world.
        </para>

        <para style="normal">
            Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum
            has been the industry's standard dummy text ever since the 1500s, when an unknown printer
            took a galley of type and scrambled it to make a type specimen book.
        </para>

        <para>
            It has survived not only five centuries, but also the leap into electronic typesetting,
            remaining essentially unchanged. It was popularised in the 1960s with the release of
            Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing
            software like Aldus PageMaker including versions of Lorem Ipsum.
        </para>

        <illustration width="125" height="140" align="CENTER">
            <image file="files/bob.jpg" x="20" y="0"/>
            <setFont name="Times-Roman" size="6"/>
            <drawString x="20" y="-20"> Bob Dobbs, CEO of ReportLab Inc.</drawString>
        </illustration>

        <para>
            Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum
            has been the industry's standard dummy text ever since the 1500s, when an unknown printer
            took a galley of type and scrambled it to make a type specimen book. It has survived not
            only five centuries, but also the leap into electronic typesetting, remaining essentially
            unchanged. It was popularised in the 1960s with the release of Letraset sheets containing
            Lorem Ipsum passages, and more recently with desktop publishing software like Aldus
            PageMaker including versions of Lorem Ipsum.
        </para>

        <para>
            Why do we use it?
        </para>
        <setNextTemplate name="page2"/>

        <para>
            It is a long established fact that a reader will be distracted by the readable content
            of a page when looking at its layout. The point of using Lorem Ipsum is that it has a
            more-or-less normal distribution of letters, as opposed to using 'Content here, content
            here', making it look like readable English. Many desktop publishing packages and web
            page editors now use Lorem Ipsum astheir default model text, and a search for 'lorem
            ipsum' will uncover many web sites still in their infancy. Various versions have
            evolved over the years, sometimes by accident, sometimes on purpose (injected humour
            and the like).
        </para>

    </story>
</document>
```

The RML code above shows how to use most of the main concepts we've discussed, in a single document layout. It contains fixed elements - the masthead, date, etc. , and a set of frames into which the text and images from the <story> section flow. It also demonstrates graphics primitives (the horizontal lines), how to use custom fonts, and using different templates for different pages.

Lesson 14: An introduction to dynamic documents.

```
<!-- Paste the following into a text file named 'example_14.prep'-->
<!-- Look inside the file 'example_14.py' to see how this   -->
<!-- template is merged with values in a csv file to create -->
<!-- a series of dynamic documents.  The values in curly    -->
<!-- brackets are dynamically substituted by ReportLab's    -->
<!-- templating system, 'preppy'.                           -->

{{def(name,location)}}

<!DOCTYPE document SYSTEM "rml.dtd">
<!-- this filename isn't used - instead the filename -->
<!-- is created dynamically inside 'example_14.py'   -->
<document filename="example_14.pdf">

    <template>
        <pageTemplate id="main">
            <pageGraphics>
                <setFont name="Helvetica-Oblique" size="16"/>
                <drawString x="100" y="700">Using RML with a templating system.</drawString>

                <setFont name="Helvetica" size="16"/>

                <!-- this demonstrates how to use python code directly inside your rml template. -->
                {{script}}
                    import datetime
                    today=datetime.datetime.today()
                    today=today.strftime("%A %B %d, %Y")
                {{endscript}}

                <drawString x="350" y="650">{{today}}</drawString>

            </pageGraphics>

            <frame id="first" x1="100" y1="400" width="400" height="200"/>
        </pageTemplate>
    </template>

    <stylesheet>
        <paraStyle name="textstyle1" fontName="Helvetica" fontSize="16" leading="16" spaceBefore="16"/>
    </stylesheet>

    <story>

        <para style="textstyle1">
            Hi {{name}},
        </para>

        <para style="textstyle1">
            How are things in {{location}}?
        </para>

        <para style="textstyle1">
            All the best, from ReportLab
        </para>
```

```
    </story>

</document>
```

```
# Paste the following into a text file named 'example_14.py'
# This python script loops through
# the lines in a csv file, and merges the
# values into an rml template named 'example_14.prep'
#
# This results in pure rml, which is then
# run through rml2pdf to create pdf files.

import csv
from rlextra.preppy import preppy
from rlextra.rml2pdf import rml2pdf

def main():

    #get the csv file into a python csv reader object
    csvdata  = open('names.csv')
    reader = csv.reader(csvdata)

    # The rml template is loaded into preppy
    template = preppy.getModule('example_14.prep')

    # loop through the lines in the csv file
    for row in reader:
        name= row[0]
        location=row[1]

        # Now merge the variables into the template.
        # you can pass in any Python objects or variables
        # as long as they evaluate.
        rmlText = template.get(name,location)

        #convert to PDF
        pdfFileName = name + '.pdf'
        rml2pdf.go(rmlText, outputFileName=pdfFileName)

        print 'saved %s' % pdfFileName


if __name__=='__main__':

    main()
```

Copy the following into a text file called 'names.csv'

```
Mrs. Brown,Birmingham
Mr. White,London
Ms. Green,Manchester
Mr. Pink,Brighton
```

While RML can be used to create static documents like the Daily Planet example you've already seen, it's most useful as part of a dynamic document templating system. In its simplest terms, you can think of it like a 'mail merge' function writ large - a simple real life example is a 'direct mail' sales letter, which typically uses fixed text for most of the body of the letter, with custom text for each particular recipient - e.g. 'Dear Mr. X'.

A simple but very powerful templating system is bundled with ReportLab PLUS. It's a python module called 'preppy', which can be found in /rlextra/preppy/. Preppy is very simple to use - the dynamic parts of the document, and any python code you might need, are enclosed in double curly brackets inside your template. The template is loaded into preppy along with any dynamic data. Preppy then evaluates any python expressions and substitutes any variables, and outputs the result as text - so in this case, you are left with complete rml text which only has to be run through rml2pdf to create a pdf document.

To use the example above, paste the first part into a text file named 'example_14.prep' and the second part into a text file name 'example_14.py'. Then create a file called 'names.csv' containing the names and towns ,as shown above. Run 'python example_14.py' at the command line and you should see a number of dynamic pdfs being created. Read the comments inside 'example_14.py' to see how preppy and rml2pdf are used inside python programs.

Frequently Asked Questions

Q. When I try to run the rml file through RML2PDF, I get errors. Help!

A. Most errors should have a reasonably self-explanatory error message. The most common cause of compilation errors is failing to close tags properly; e.g. a <para> without a </para>. Another common mistake is failing to escape certain characters within your <para> tags - e.g. left and right angle brackets (substitute with < and >), ampersands (substitute with &) etc. - refer to Google for how to escape other characters in XML. Also, you will frequently find yourself using the wrong elements in a tag - refer to the DTD for the correct tag syntax, and remember that tags are case sensitive.

Q. I specified a table/ some text / an image in my rml document, but when I compile it to PDF, it's not there.

A. This is a common problem for beginners. If the document complies into PDF without errors, the element is probably not visible because it's hidden behind another element, or the frame containing it is too small. Try changing the x/y co-ordinates of the element, or expanding the size of the frames, or shrinking the size of the fonts/images/text etc, then recompile until the element becomes visible. You can sometimes see if a hidden element is present at all by selecting the whole page with the mouse in Acrobat reader.

Q. My image/frame/table is in the wrong place. Why?

A. Remember that in RML, co-ordinates are specified from the bottom left corner of any element, including the page itself. This is in line with the way PostScript works, but it may be somewhat unfamiliar.

Q. I'm having trouble making sense of the DTD. Can you help?

A. A (partial) sample entry for the image tag is as follows:

```
<!ELEMENT image EMPTY>
<!ATTLIST image
    file CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    showBoundary (yes | no | 0 | 1) #IMPLIED
>
```

From this entry, you can see the following:

• using this tag requires a filename, along with x and y coordinates (these are marked as CDATA #REQUIRED).
• the width and height are optional 'chardata' parameters (marked as CDATA #IMPLIED).
• the showBoundary parameter is optional (#IMPLIED) and shows a list of the permitted values.
So a sample use of this tag would be <image file="pic.jpg" x="0" y="0" showBoundary="yes">, which would place the a picture 'pic.jpg' at the bottom left corner of a document, with a surrounding border.

Q. How do I specify the page size & orientation?

A. The page size defaults to A4, portrait. You can change it for the whole document within the <template> tag, and you can override it for individual pages using the <pageTemplate> tag.

Q. How can I use my own fonts?

A. By default, you can use the 14 standard fonts supplied with Acrobat Reader. However, RML also accepts TrueType and Type1 fonts. Examples of how to use them are in test_005_fonts.rml and test_023_ttfonts.rml in the rml2pdf/tests/ directory.

Q. Sorry, I still don't get it.

A. Try reading the manual in rml2pdf/docs; look at the test cases in rml2pdf/test/ , or have a look at the code snippets on the ReportLab website.

Q. I'm a bit confused about how to use 'feature X'

A. Try reading the manual in rml2pdf/docs; look at the test cases in rml2pdf/test/ , or have a look at the code snippets on the ReportLab website.

Q. I really need to use 'feature Y'. Does RML support it?

A. Try reading the manual in rml2pdf/docs; look at the test cases in rml2pdf/test/ , or have a look at the code snippets on the ReportLab website.

Q. I've discovered how to do something amazing in RML. How can I tell the world?

A. Post a code snippet on the ReportLab website! Go to http://www.reportlab.com/snippets