Application Programmer's Interface (API) Reference Document

Version 2.0.1 – June 2004

For Java 1.4.2+ with WebObjects 5.2+ on Windows, Mac OS X, Solaris, and Linux

TABLE OF CONTENTS

1.1 – Introduction

RCMPDFKit is WEBAPPZ's solution for printing documents according to Adobe's® Portable Document Format (PDF). PDF is the universal standard for the distribution of electronic documents. The problems generally encountered with electronic file sharing are avoided as PDF is not limited to an application, font types, formatting, graphics, or software and printer limitations.

RCMPDFKit is an easy to use, powerful software tool that generates web browser embedded PDF files, including table-based PDF reports. The ability to add images to any of the pages and to add multiple tables consecutively are just some of the features. The tables use a cell-based architecture enabling many color, font, and image combinations to meet different user requirements.

The Kit's generation of PDF output has been optimized for binary compression. This means reduced PDF download times and enhanced performance.

We have recently (v1.4) improved how PDF is presented through the client browser. If the Acrobat plug-in is found, the PDF will appear within the loading window; otherwise, the PDF will be downloaded and opened by a client-side application – outside of the browser - and the downloading window will close itself once downloading has completed. You also have the option now to specify file names for client side downloading: e.g. “statement2003.pdf” or “timesheet20040202.pdf”.

Further, in v2.0.1, we improved the kit’s font support. Our PDFkit currently supports 300 server side fonts with Adobe Font Metrics (afm) files, and regardless of the operating system, every client system supports at least the four standard PDF fonts: Courier, Helvetica, Times, and Symbol.

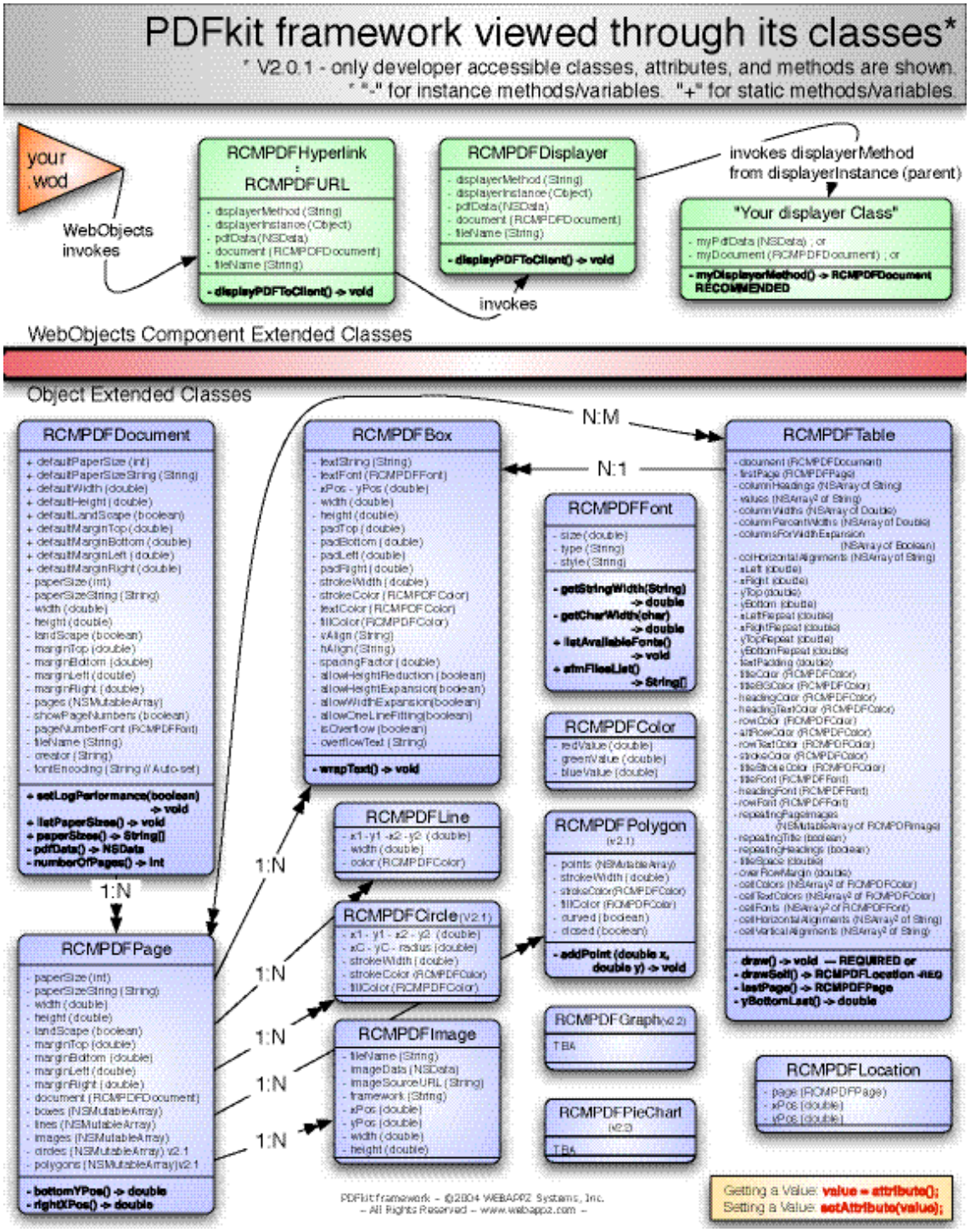
In addition, v2.0.1 now provides support for international paper sizes. So, wherever you are printing your documents, there are no limitations.

If your company exchanges documents electronically, then RCMPDFKit is the answer to consistent, professional, hassle-free printing.

Super easy installation: On Mac OS X, double click the PDFKit installer package. On other platforms, just place the RCMPDFKit.framework directory, along with your other WebObjects frameworks. Then, include it in Xcode or Project Builder. Now, you will have full access to its API. Please read the section “Integrating RCMPDFKit with your application” below.

1.2 – Overview of Classes

Please note that all attributes are accessed as accessor methods. E.g. if an attribute is called “width”, you would call x = width() and setWidth(x). Similarly, with “marginBottom”, you would call x = marginBottom() and setMarginBottom(x).

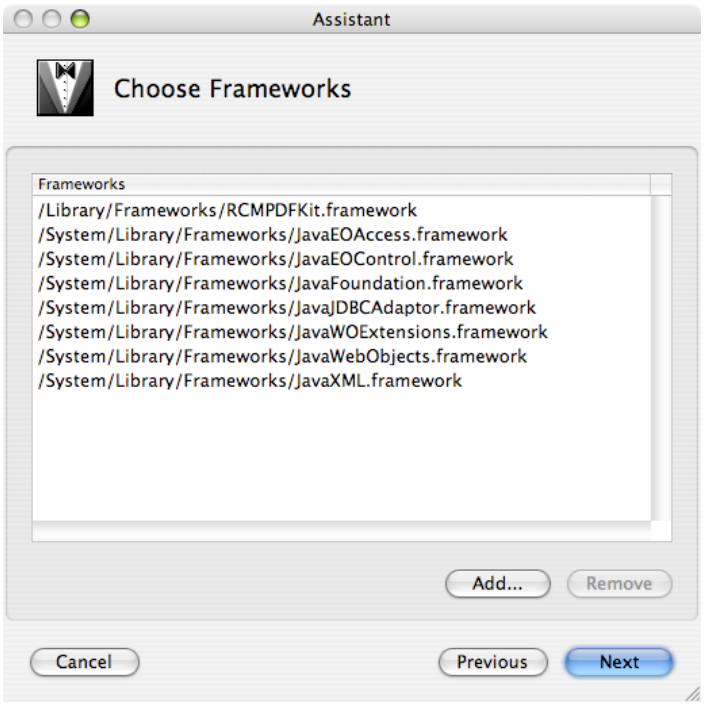


2.1 – Integrating RCMPDFKit with your application

RCMPDFKit.framework is a WebObjects Framework directory and should be installed in (/Apple)/Local/Library/Frameworks on Windows/Solaris/Linux, and /Library/Frameworks on Mac OS X along with any other frameworks that may already exist at that location.

Xcode – Mac OS X from version 10.3:

A: When you create a new "WebObjects Application" or "WebObjects Framework" type of project, add the RCMPDFKit.framework as shown below.



B: With an existing Xcode project:

- 1 Click on the "Frameworks" folder in the left folder bar.
- 2 Click on the "Project" menu, and then "Add Frameworks…".
- 3 Select RCMPDFKit.framework in /Local/Library, and click "Add"
- 4 Please ensure that the target is "Application Server".

ProjectBuilder – Windows and Mac OS X prior to version 10.3:

In the browser of your PB.project, double-click the Frameworks suitcase to bring up the Add Frameworks panel. Browse to the RCMPDFKit.framework and click the Open button. If the path to (/Apple)/Local/Library/Frameworks is not already in your search order, you may agree with the dialog box suggesting to add it.
From Your Java Classes: `import com.webappz.pdfkit.*;`

2.2 – Simple Example

To ensure that you have installed the RCMPDFKit framework correctly, you may try the "Hello World" example below. For more examples, please see the included RCMPDFKitTutorials project. Imagine you are working with a WebObjects page (WOComponent) called "Main.wo". This is also the default page.

In the class Main.java, you add the following method:

```
import com.webappz.pdfkit.*;

public RCMPDFDocument helloWorld() {
    RCMPDFDocument doc = new RCMPDFDocument();  // create a new pdf docum.
    RCMPDFPage page1 = new RCMPDFPage(doc);     // create a new pdf page
    page1.addBox(new RCMPDFBox(50,50,"Hello World"));
    return doc;
}
```

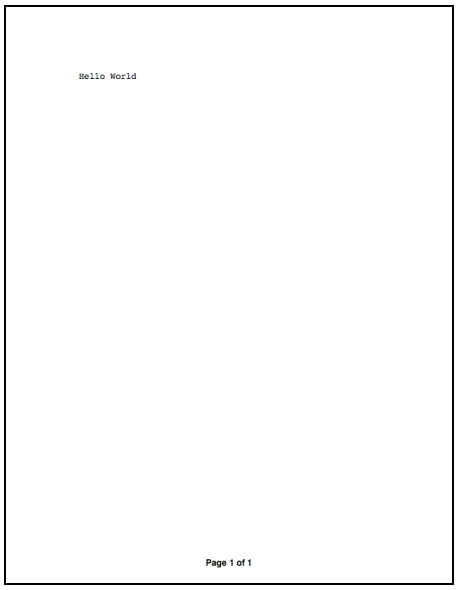
In Main.wod, add a descriptor:

```
HelloWorld: RCMPDFHyperlink { displayerMethod = "helloWorld"; }
```

In Main.html, add a link:

```
<WEBOBJECT NAME=HelloWorld>View Hello World</WEBOBJECT><BR>
```

Now build and run the project, click the link, and you should get this PDF page:



3.1 – RCMPDFHyperlink *extends WOComponent*

We have recently (since v1.4) improved how PDF is presented through the client browser. If the Acrobat plug-in is found, the PDF will appear within the loading window; otherwise, the PDF will be downloaded and opened by a client-side application – outside of the browser - and the downloading window will close itself once downloading has completed. You also have the option now to specify file names for client side downloading: e.g. “statement2003.pdf” or “timesheet20040202.pdf”.

As part of these improvements we have added two new classes **RCMPDFHyperlink** and **RCMPDFURL,** to ensure, in our opinion, the best client-side experience. Please note that **RCMPDFDisplayer** is invoked by these two classes and should not have to be invoked directly. The advantage of these classes is that before your method is invoked server-side, the client is already aware of that - the browser window shows “Generating PDF”. Using the **RCMPDFDisplayer** class, directly, would cause an initial client-side delay.

Attributes - passed in the .wod file of your WebObjects Component

Attributes must be declared in the WebObjects “.wod” descriptor files. Do not instantiate these classes directly. For example, if you have a WebObjects component called “FundsOverview.html”, with a hyperlink tag to LinkPrintPDF, your “FundsOverview.wod” file could appear as follows:

```
LinkPrintPDF: RCMPDFHyperlink {
        displayerMethod = "myFundsOverviewDocGenerator";
        displayerInstance = myReportingInstance;
        fileName = “Funds_Overview.pdf”;
}
```

String fileName (optional)

The PDF downloading to the client will have this file name. If not specified, a fileName from the RCMPDFDocument will be used. If that is also unavailable, the RCMPDFKit will generate an automatic name, based on the current date.

String displayerMethod
(recommended; required if pdfData and document are not used)

This is the recommended parameter, and in most cases, the **only** one you need to use. The RCMPDFKit will invoke your method specified here to generate an **RCMPDFDocument.** Please be sure your method returns this class. If the parameter “displayerInstance” is not used, your method should reside in the Java class belonging to the “.wod” file where you have placed the descriptor.

Object displayerInstance (optional)

Use this parameter if your method resides somewhere other than the class belonging to the ".wod" file; e.g. application or session

NSData pdfData
(required if displayerMethod and document are not used)

In special cases where you already have PDF binary data, which you wish to transport just to the client browser, you can use this parameter to specify the NSData instance containing your PDF data.

RCMPDFDocument document
(required if displayerMethod and pdfData are not used)

If you have already previously generated your RCMPDFDocument, you can use this parameter to transport it to the client browser.

3.2 – RCMPDFURL *extends RCMPDFHyperlink*

This class is intended for more sophisticated WebObjects/HTML developers to access the RCMPDFKit from JavaScript, Flash or other types of specially formed URLs. **RCMPDFURL** works exactly like a **WOActionURL** except that the interaction is generally with the RCMPDFKit and not WebObjects. The parameter passed arguments for this class match those of **RCMPDFHyperlink** as it extends this class.

As PDF generating windows might close themselves, it is recommended to always bring up this type of link in a new window, either by JavaScript's "window.open" method, or by using target = "_new".

Example:

In an Example.html file, you could have the following JavaScript function:

```
<SCRIPT Language="JavaScript">
var printUrl = "<WEBOBJECT NAME=GoPDFDisplayer></WEBOBJECT>";
function printThis() {
       printwindow = window.open(printUrl,'PrintWindow','');
       printwindow.focus();";
}
</SCRIPT>
```

with an Example.wod file:

```
GoPDFDisplayer: RCMPDFURL {
       displayerMethod   = "makePDF" ;
       filename="Example.pdf"
}
```

Then your Example.java file would have a method as follows:

```
public RCMPDFDocument makePDF() {
       code to fill up your document
       …
}
```

3.3 – RCMPDFDisplayer *extends WOComponent*

This class is part of the older RCMPDFKit Application Programmer's Interface. In some cases, where a developer wishes to create PDF as part of **WOActionResults**, this class can be invoked as a normal WebObjects component would. By default, this class is invoked by **RCMPDFHyperlink** and **RCMPDFURL**, and as such should not have to be invoked directly. Client-side delays may occur.

See below how an RCMPDFDisplayer could be used in your WebObjects application, such as within a ViewReports page:

In ViewReports.html:

```
<WEBOBJECT NAME=PDFTest>VIEW PDF</WEBOBJECT>
```

In ViewReports.wod:

```
PDFTest: WOHyperlink { action = goPrint; target = "_new" }
```

In ViewReports.Java

```
Public class ViewReports extends WOComponent {

        … other code, such as constructors, etc…

        public WOComponent goPrint() {

                RCMPDFDocument myDoc = new RCMPDFDocument();
                RCMPDFDisplayer result = pageWithName("RCMPDFDisplayer");
                result.setDocument(myDoc);
                result.setFileName("report.pdf");


                … code to fill up your myDoc …

                return result;
        }
}
```

Attributes

The attributes are similar to **RCMPDFHyperlink**, but are not passed through ".wod" parameters. Instead, you must code them in your action method, as the example above shows.

4.1 – RCMPDFDocument

This class contains one or more pages (RCMPDFPages), and has a method that returns PDF data (NSData type) for all of its pages. You can set various static and instance variables with information about pages managed by this document class.

Constructors

public RCMPDFDocument()

public RCMPDFDocument(boolean landScape)

public RCMPDFDocument(int paperSize)

public RCMPDFDocument(int paperSize, boolean landScape)

public RCMPDFDocument(String paperSizeString)

public RCMPDFDocument(String paperSizeString, boolean landScape)

Please see "Layout Instance Attributes" below for an explanation of the attributes used in these constructors.

Layout Static Attributes

The following attributes apply application wide and will be re-used as defaults, every time a new document is created. Although these defaults are pre-set by the RCMPDFKit, the developer is welcome to change these values.

static int	**defaultPaperSize**	**= RCMPDFDocument.LETTER;**
static String	**defaultPaperSizeString**	**= "Letter";**
static double	**defaultHeight**	**= 792;**
static double	**defaultWidth**	**= 612;**
static boolean	**defaultLandScape**	**= false;**
static double	**defaultMarginTop**	**= 50;**
static double	**defaultMarginBottom**	**= 50;**
static double	**defaultMarginLeft**	**= 50;**
static double	**defaultMarginRight**	**= 50;**

E.g. 'RCMPDFDocument.setDefaultMarginTop(72);' sets the default top margin to 72 points or 1 inch.

Layout Instance Attributes

int paperSize (defaults from defaultPaperSize)
Paper size from Constant - see “Paper Sizes Defined As Constants” below.

String paperSizeString (defaults from defaultPaperSizeString)
Paper size from String - see “Paper Sizes Defined As Strings” below.

double width (defaults from defaultWidth)
Page width in pixels (1/72” points)

double height (defaults from defaultHeight)
Page height in pixels (1/72” points)

boolean landscape (defaults from defaultlLandScape)
If **true**, width and height will be flipped – landscape - **false** implies portrait

double marginTop (defaults from defaultMarginTop)
Top margin of the page in pixels (1/72” points)

double marginBottom (defaults from defaultMarginBottom)
Bottom margin of the page in pixels (1/72” points)

double marginLeft (defaults from defaultMarginLeft)
Left margin of the page in pixels (1/72” points)

double marginRight (defaults from defaultMarginRight)
Right margin of the page in pixels (1/72” points)

Non Layout Instance Attributes

NSMutableArray pages (At least one page is required)
This attribute contains all the RCMPDFPages for this document.

boolean showPageNumbers (default = true)
This attribute controls whether or not the page number is displayed on the page. If it is set to true, the page number appears at the bottom of the page; otherwise, page numbers are not shown.

RCMPDFFont pageNumberFont (default = `new RCMPDFFont("Helvetica", "Bold",12`**)**
This attribute defines the font used for displaying page numbers.

String fileName (default = a date based file name)

The PDF downloading to the client will have this file name. If not specified, the RCMPDFKit will generate an automatic name, based on the current date.

String creator (default = `WOApplication.application().name()`)

Creator of the PDF file – will be shown in Adobe Acrobat's Document Information along with a document creation date. If not specified, the name of your WebObjects application will be used. The producer will always be "RCMPDFKit".

String fontEncoding (default = "MacRomanEncoding" on non-Windows clients and "WinAnsiEncoding" on Windows)

This attribute is automatically determined by the kit, but may be over-ridden.

Static Methods

static void setLogPerformance(boolean)

If set to true, all invocations of the **pdfData()** method within your application will be timed and printed to the standard output console. See RCMPDFKitTutorials for an example. The default value is false.

static void listPaperSizes()

Prints out a list of international paper sizes supported by the PDFKit to the standard console output: in 1/72 inch points, in inches, and in millimeters. If you find any sizes missing in this extensive list, please contact WEBAPPZ, and we will add such sizes to the next version. Please see "Paper Sizes Defined as Strings" below for this method's output.

static String[] paperSizes()

Returns a list of international paper sizes supported by the PDFKit in String array form.

Instance Methods

public NSData pdfData()

Generates an NSData object - for displaying this document – holding the PDF content.

public int numberOfPages()

Returns the number of pages.

public void addPage(RCMPDFPage value)

Add an RCMPDFPage to this document. This method should not have to be invoked directly in most scenarios, as the constructors of RCMPDFPage will already call this method.

public void removePage(RCMPDFPage value)

Remove an RCMPDFPage from this document.

Paper Sizes Defined as Strings

```
##*## RCMPDFKit: LIST OF AVAILABLE PAPER SIZES

-------------------------------------------------------------
| PAPER SIZE   |POINTS(1/72")|      INCHES      |  MILIMETERS |
-------------------------------------------------------------
|     Business |  297 x  684 |  4.125 x  9.500 |  105 x  242 |
|   Commercial |  576 x  765 |  8.000 x 10.625 |  203 x  271 |
|    Executive |  522 x  756 |  7.250 x 10.500 |  184 x  268 |
|        Folio |  596 x  936 |  8.278 x 13.000 |  210 x  332 |
|       Folio2 |  612 x  936 |  8.500 x 13.000 |  216 x  332 |
|     Foolscap |  612 x  936 |  8.500 x 13.000 |  216 x  332 |
|   HalfLetter |  396 x  612 |  5.500 x  8.500 |  140 x  217 |
|       Ledger | 1224 x  792 | 17.000 x 11.000 |  432 x  281 |
|        Legal |  612 x 1008 |  8.500 x 14.000 |  216 x  357 |
|       Letter |  612 x  792 |  8.500 x 11.000 |  216 x  281 |
|         Note |  539 x  720 |  7.486 x 10.000 |  190 x  255 |
|       Quarto |  576 x  720 |  8.000 x 10.000 |  203 x  255 |
|      Quarto2 |  610 x  780 |  8.472 x 10.833 |  215 x  276 |
|   SmallLegal |  612 x  936 |  8.500 x 13.000 |  216 x  332 |
|    Statement |  396 x  612 |  5.500 x  8.500 |  140 x  217 |
|      Tabloid |  792 x 1224 | 11.000 x 17.000 |  279 x  434 |
|  Env_Monarch |  279 x  539 |  3.875 x  7.486 |   98 x  191 |
|       Env_Dl |  312 x  624 |  4.333 x  8.667 |  110 x  221 |
|       Env_C4 |  649 x  918 |  9.014 x 12.750 |  229 x  325 |
|       Env_C5 |  459 x  649 |  6.375 x  9.014 |  162 x  230 |
|       Env_C6 |  323 x  459 |  4.486 x  6.375 |  114 x  163 |
|      Env_C10 |  297 x  684 |  4.125 x  9.500 |  105 x  242 |
|      Env_C65 |  324 x  648 |  4.500 x  9.000 |  114 x  230 |
|           A0 | 2383 x 3370 | 33.097 x 46.806 |  841 x 1194 |
|           A1 | 1685 x 2383 | 23.403 x 33.097 |  594 x  844 |
|           A2 | 1191 x 1685 | 16.542 x 23.403 |  420 x  597 |
|           A3 |  842 x 1191 | 11.694 x 16.542 |  297 x  422 |
|           A4 |  595 x  842 |  8.264 x 11.694 |  210 x  298 |
|           A5 |  421 x  595 |  5.847 x  8.264 |  149 x  211 |
|           A6 |  297 x  421 |  4.125 x  5.847 |  105 x  149 |
|           A7 |  210 x  297 |  2.917 x  4.125 |   74 x  105 |
|           A8 |  148 x  210 |  2.056 x  2.917 |   52 x   74 |
|           A9 |  105 x  148 |  1.458 x  2.056 |   37 x   52 |
|          A10 |   74 x  105 |  1.028 x  1.458 |   26 x   37 |
|           B0 | 2920 x 4127 | 40.556 x 57.319 | 1030 x 1462 |
|           B1 | 2064 x 2920 | 28.667 x 40.556 |  728 x 1034 |
|           B2 | 1460 x 2064 | 20.278 x 28.667 |  515 x  731 |
|           B3 | 1032 x 1460 | 14.333 x 20.278 |  364 x  517 |
|           B4 |  729 x 1032 | 10.125 x 14.333 |  257 x  366 |
|           B5 |  516 x  729 |  7.167 x 10.125 |  182 x  258 |
|           B6 |  363 x  516 |  5.042 x  7.167 |  128 x  183 |
|           B7 |  258 x  363 |  3.583 x  5.042 |   91 x  129 |
|           B8 |  181 x  258 |  2.514 x  3.583 |   64 x   91 |
|           B9 |  127 x  181 |  1.764 x  2.514 |   45 x   64 |
|          B10 |   91 x  127 |  1.264 x  1.764 |   32 x   45 |
|           C0 | 2599 x 3676 | 36.097 x 51.056 |  917 x 1302 |
|           C1 | 1838 x 2599 | 25.528 x 36.097 |  648 x  920 |
|           C2 | 1299 x 1838 | 18.042 x 25.528 |  458 x  651 |
|           C3 |  919 x 1299 | 12.764 x 18.042 |  324 x  460 |
|           C4 |  649 x  919 |  9.014 x 12.764 |  229 x  325 |
|           C5 |  459 x  649 |  6.375 x  9.014 |  162 x  230 |
|           C6 |  324 x  459 |  4.500 x  6.375 |  114 x  163 |
|           C7 |  229 x  324 |  3.181 x  4.500 |   81 x  115 |
|           C8 |  162 x  229 |  2.250 x  3.181 |   57 x   81 |
|           C9 |  114 x  162 |  1.583 x  2.250 |   40 x   57 |
|          C10 |   81 x  114 |  1.125 x  1.583 |   29 x   40 |
-------------------------------------------------------------
```

Paper Sizes Defined as Constants

The above list of paper sizes is also defined in static int attributes by RCMPDFDocument. E.g. document.setPaperSize(RCMPDFDocument.EXECUTIVE);

North American Sizes:

```
BUSINESS, COMMERCIAL, EXECUTIVE, FOLIO, FOLIO2, FOOLSCAP,
HALFLETTER, LEDGER, LETTER, NOTE, QUARTO, SMALLLEGAL, STATEMENT,
TABLOID
```

Envelope Sizes:

```
ENV_MONARCH, ENV_DL, ENV_C4, ENV_C5, ENV_C6, ENV_C10, ENV_C65
```

International Sizes:

```
A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10

B0, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10

C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10
```

4.2 – RCMPDFPage

This class represents a PDF page, which contains tables (a number of boxes), boxes, lines, and images. Besides belonging to an RCMPDFDocument, it has several attributes associated with layout.

Constructors

public RCMPDFPage(RCMPDFDocument document)

public RCMPDFPage(RCMPDFDocument document,
int paperSize)

public RCMPDFPage(RCMPDFDocument document,
String paperSizeString)

public RCMPDFPage(RCMPDFDocument document,
double width,
double height)

public RCMPDFPage(RCMPDFDocument document,
double width,
double height,
double marginTop,
double marginBottom,
double marginLeft,
double marginRight)

Please see “Layout Attributes” below for an explanation of the attributes used in these constructors.

Layout Attributes

int paperSize (defaults from document.paperSize())
Paper size from Constant - see “Paper Sizes Defined As Constants” above

String paperSizeString (defaults from document.paperSizeString())
Paper size from String - see “Paper Sizes Defined As Strings” above

double width (defaults from document.width())
Page width in pixels (1/72” points)

double height (defaults from document.height())
Page height in pixels (1/72" points)

boolean landscape (defaults from document.landScape())
If **true**, width and height will be flipped – landscape - **false** implies portrait

double marginTop (defaults from document.marginTop())
Top margin of the page in pixels (1/72" points)

double marginBottom (defaults from document.marginBottom())
Bottom margin of the page in pixels (1/72" points)

double marginLeft (defaults from document.marginLeft())
Left margin of the page in pixels (1/72" points)

double marginRight (defaults from document.marginRight())
Right margin of the page in pixels (1/72" points)

Non Layout Attributes

RCMPDFDocument document (Set in Constuctor)
Document this page belongs to - required.

NSMutableArray boxes
Array of RCMPDFBoxes on the page

NSMutableArray lines
Array of RCMPDFLines on the page

NSMutableArray images
Array of RCMPDFImages on the page

NSMutableArray circles
Array of RCMPDFCircles on the page (planned for version 2.1)

NSMutableArray polygons
Array of RCMPDFPolygons on the page (planned for version 2.1)

Methods

public void addBox (RCMPDFBox value)
Add an RCMPDFBox to the page

public void removeBox(RCMPDFBox value)
Remove an RCMPDFBox from the page

public NSMutableArray boxes()
Returns all the boxes on the page

public void setBoxes(NSMutableArray value)
Set all the boxes on the page

public void addLine (RCMPDFLine value)
Add an RCMPDFLine to the page

public void removeLine(RCMPDFLine value)
Remove an RCMPDFLine from the page

public NSMutableArray lines()
Returns all the lines on the page

public void setLines(NSMutableArray value)
Set all the lines on the page

public void addImage(RCMPDFImage value)
Add an RCMPDFImage to the page. You must add the page to an RCMPDFDocument before adding any images to the page.

public void removeImage(RCMPDFImage value)
Remove an RCMPDFImage from the page

public NSMutableArray images()

Returns all the images on the page

public void setImages(NSMutableArray value)

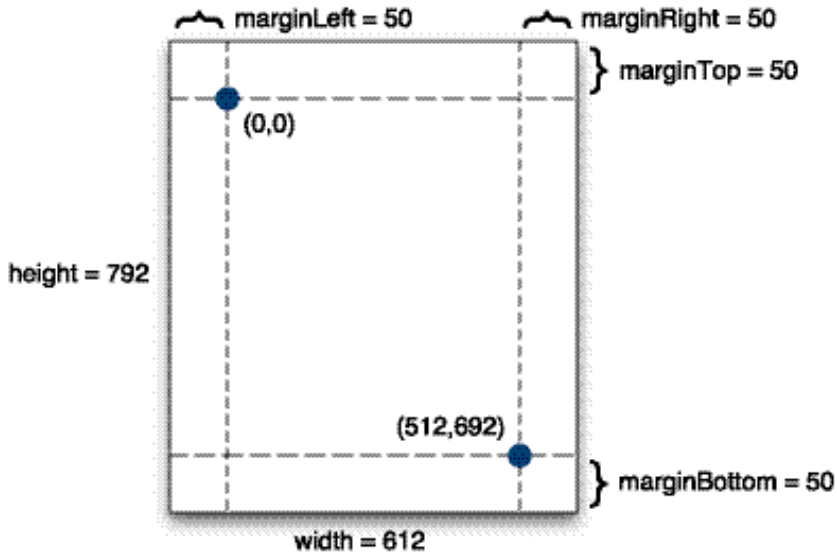
Set all the images on the page

Example: Add and remove content in this page

```
yourImage = new RCMPDFImage("filename");
aPage.addImage(yourImage);
aPage.removeImage(yourImage)
```

Page Layout used in the PDFkit framework

All pages have x-coordinates starting with zero at the left margin line (marginLeft), and the maximum value on the right margin line (rightXPos). Y-coordinates start with zero at the top margin line, and the maximum value on the bottom margin line (bottomYPos).



public double rightXPos (calculated as width - marginLeft - marginRight)

Convenience method - returns the maximum x position of the right of the page. A page with default settings would yield 612 – 50 – 50 = 512

public double bottomYPos (calculated as height - marginTop - marginBottom)

Convenience method - returns the maximum y position of the bottom of the page. A page with default settings would yield 792 – 50 – 50 = 692

4.3 – RCMPDFBox

The primary usage for **RCMPDFBox** is to place text anywhere on a page. This class has a number of features: x position, y position, width, height, stroke color, fill color, and text color, and padding space around text within the box. It also features horizontal and vertical text alignment including text wrapping. The box can be expanded and/or shortened according to the text's length. An RCMPDFTable consists of several of these boxes arranged together.

Constructors

1. Using default line width and without setting text string
public RCMPDFBox(double xPos, double yPos, double width, double height)

2. Using default line width and with setting text string
public RCMPDFBox(double xPos, double yPos,
double width, double height,
String textString)

3. Setting line width, but no text string
public RCMPDFBox(double strokeWidth, double xPos, double yPos,
double width, double height)

4. Constructor setting line width and text string
public RCMPDFBox(double strokeWidth,
double xPos, double yPos,
double width, double height,
String textString)

5. Constructor for text only, without box lines (defaults width and height to zero)
public RCMPDFBox(double xPos, double yPos, String textString)

Remember, you can set the text font, adjust the box size, and control box expansion and shortening through the accessor methods at any given point.

Attributes

double width (default = 10)
Box width in pixels

double height (default = 10)
Box height in pixels

double xPos (default = 0)
Box horizontal starting position

double yPos (default = 0)
Box vertical starting position

double padTop (default = 2)
Padding space between the top of the text and the box

double padBottom (default = 2)
Padding space between the bottom of the text and the box

double padLeft (default = 2)
Padding space between the left of the text and the box

double padRight (default = 2)
Padding space between the right of the text and the box

double strokeWidth (default = 1.0)
Stroke (= border line) width

RCMPDFColor strokeColor (default = black)
Box stroke (= border line) color

RCMPDFColor fillColor (default = white)
Box fill color – inside or background color

RCMPDFColor textColor (default = black)
Color of the box text

String textString (default = "")
This attribute is the box text. When the text is long, it is wrapped to a number of lines according to the box width. If a word in the text is longer than the box's width, this long word is wrapped also.

RCMPDFFont textFont

Font of the text in the box

String vAlign (default = "top")

Vertical alignment of the box text. Its value can be "top", "middle" or "bottom"

String hAlign (default = "left")

Horizontal alignment of the box text. Its value can be "left", "center" or "right"

double spacingFactor (default = 1.25)

Spacing in between multiple text lines. Increase this number to increase leading.

boolean allowHeightReduction (default = false)

When its value is true, the box height can be reduced according to box text length; when its value is false, the box height may be fixed or expanded according to **allowHeightExpansion's** value. We can control box height by setting and resetting this attribute.

boolean allowHeightExpansion (default = true)

When its value is true, the box height can be expanded according to text length; when its value is false, the box height may be fixed or reduced according to **allowHeightReduction's** value. We can control box height by setting and resetting this attribute.

boolean allowWidthExpansion (default = false)

If true, the width of the box will be expanded - if necessary - with a maximum of **allowWidthExpansionMaxPerc**. This only applies to fitting one line of text.

double allowWidthExpansionMaxPerc (default = 200.0)

Used in combination with above attribute. Default is the width may be expanded two-fold.

double allowOneLineFitting (default = false)

If true and the text does not fit on the first line, the RCMPDFKit framework will attempt to fit all text on one line. This is a multiple step process, where the fontSize is reduced by **allowOneLineFittingReductionSize** each step, until the text fits or the font is reduced to **allowOneLineFittingMinSize** without success.

double allowOneLineFittingMinSize (default = 4.0)

Minimum acceptable font size for one line fitting.

double allowOneLineFittingReductionSize (default = 1.0)

Step size for fitting process; lower numbers – e.g. 0.01 - will provide more detail but require more calculations.

boolean isOverflow (default = false)

This attribute is a boolean variable set by **RCMPDFBox** itself after **wrapText**() has been invoked. When the bottom of the page is reached or **allowHeightExpansion** is set to false, the box height may be fixed. In this case, if the text is longer than the box, **isOverflow** is set to true.

String overflowText (default = "")

Holds content of overflow text if the box cannot be expanded enough to hold all requested text (see above). The program captures this automatically.

Methods

public void wrapText()

Invoke this method if you are concerned that the text may not fit in the box or wish to see how high the box will get. If **isOverflow()** returns true, you may either adjust the box to make it larger, remove the box to put it on another page, or keep the box, but create another one on the next page with the text **from overflowText()**. If you don't invoke this method, the **RCMPDFPage** will invoke it at time of PDF generation.

NOTE: You may only invoke wrapText() after the box has been added to an RCMPDFPage.

4.4 – RCMPDFTable

RCMPDFTable has general features of a normal table. It allows you to specify column width explicitly or by percentage of the row width using **setColumnWidths()** or **setColumnPercentWidths()**, respectively, once the table is created. It also allows you to set column horizontal alignment to left, center or right by setting **colHorizontalAlignments**. You can control the table position and size by setting the attributes **xLeft**, **xRight**, **yTop**, **yBottom** in the first page, and **xLeftRepeat**, **xRightRepeat**, **yTopRepeat**, **yBottomRepeat** in the remaining pages, using the accessor methods. This class uses **RCMPDFBoxes** internally as table cells. If the first page is full, repeating pages are automatically generated. At the end, the last/final page is returned, and you can subsequently add another table to the last page.

To create a table, you must pass in the first page (add to a document before used to create the table), on which the table starts, an array of string for heading, and an array of array of string for rows' content.

Tables are ideal to present database driven reports.

Constructors

public RCMPDFTable(RCMPDFPage firstPage,
NSArray columnHeadings,
NSArray values)

You must call the **draw()** method, finally, to complete the **RCMPDFTable** operation.

Example: How to create and use RCMPDFTable

1) Create a String array for heading

```
String[] headings = {"Report File Name",
                     "Description", "Audiences",
                     "User Specified Selection Criteria",
                     "Key Questions Answered "};
```

2) Create the heading

```
NSArray heading = new NSArray(headings);
```

3) Create some rows

```
String[] preRow1 = {"FunnyReport.imr", "Interesting",
                     "* Marketing Buyers",
                     "* Secure Internet Connections ",
                     "* What is my name?"  };
String[] preRow2 = …….
       ……..
```

```
NSArray row1 = new NSArray(preRow1);
NSArray row2 = …….
        ……..

NSArray[] preRows = { row1, row2, row3, row4, row5,
                        row6,  row7, row8, row9 };
NSArray rows = new NSArray(preRows);
```

4) Create an RCMPDFPage as the firstPage of RCMPDFTable

```
RCMPDFPage firstPage = new RCMPDFPage(myPage);
```

5) Create an RCMPDFTable, select some settings and draw the table

```
// create a table
RCMPDFTable testTable = new RCMPDFTable(firstPage,
                              headings, rows);

//setting column width by percentage
testTable.setColumnPercentWidths(new
                        NSArray(new Double[]
                        {new Double(20.0),new Double(30),
                        new Double(15), new Double(25),
                        new Double(10)}));

// or by absolute values
testTable.setColumnWidths(new NSArray(new Double[] {
      new Double(130),new Double(110), new Double(70),
      new Double(100), new Double(100)}));

// setting fonts....
testTable.setRowFont(new RCMPDFFont("Times","Bold", 16));
testTable.setRowColor(new RCMPDFColor(0.9,0.9,1));

//draw the table
testTable.draw();  //MUST DO THIS !!!
```

6) If you want to add another table after this table,

```
RCMPDFPage tempPage = testTable.lastPage();
double tempY = testTable.yBottomLast();
tempPage.setIsRepeating(false);//IMPORTANT !!!
                              //if you have repeating image
testTable = new RCMPDFTable(tempPage,
                anotherHeading, anotherRows);

testTable.setColumnWidths(new NSArray(new Double[] {
    new Double(130),new Double(110),
    new Double(70), new Double(100),
    new Double(100)}));

testTable.setYTop(tempY+50);  //make some space from last
```

```
                          //table to this table by "+ 50"
testTable.setHeadingColor(new RCMPDFColor(1,1,0));
testTable.setRowFont(new RCMPDFFont("Times","Plain", 16));
testTable.setRowColor(new RCMPDFColor(0.9,1,.9));
testTable.draw(); // DON'T FORGET TO CALL draw()!
```

Note: You must provide an equal number of values for the heading and for a row; otherwise, an error message will be generated.

Attributes - Basic

RCMPDFDocument document (Optional. Set automatically)

The constructor determines the document this table belongs to by its firstPage argument. So is not necessary to set this attribute directly.

RCMPDFPage firstPage (Required)

This is the first page where you start a table. This page must have been added to an **RCMPDFDocument** before a table can be drawn. As you pass this attribute in the constructor, you are not required to set it individually.

NSArray columnHeadings (Optional)

String array holding the column titles – can be omitted, in which case, no headings will be shown.

NSArray[2] values (Required)

NSArray of rows holding each an NSArray of Strings for each row with text values for the table's cell text content.

String title (Optional)

Title to be displayed above the table. You can specify its font and color with **titleFont** and **titleColor**.

NSArray columnWidths (Optional)

NSArray of Double holding the column width numbers. If not provided, and **columnPercentWidths'** values are not set, the column widths will be calculated intelligently based on all the cells' contents.

NSArray columnPercentWidths (Optional)

NSArray of Double holding the column width percentage numbers. If not provided, and **columnWidths'** values are not set, the column width will be calculated intelligently based on all the cells' contents.

NSArray columnsForWidthExpansion (Optional)

NSArray of Boolean (class java.lang.Boolean) indicating for each column, if its width may be expanded according to the table's intelligent calculations. This attribute has no effect if **column(Percent)Widths** are set.

NSArray colHorizontalAlignments (Optional)

NSArray of String values - either "left", "right", or "center" - of column horizontal setting. The number of values in this attribute must be equal to that in columnHeadings (if not null), or to that in the first object in the values array.

double xLeft (default = RCMPDFPage.defaultLeft)

Left position of table in pixels. Defaults to the most left position within the margins of the page.

double xRight (default = RCMPDFPage.defaultRight)

Right position of table in pixels. Defaults to the most right position within the margins of the page.

double yTop (default = RCMPDFPage.defaultTop)

Top position of table in pixels. Defaults to the most upper position within the margins of the page.

double yBottom (default = RCMPDFPage.defaultBottom)

Bottom position of table in pixels. Defaults to the lowest position within the margins of the page. Depending on the size of your table, this position may not be reached as it is only a maximum allowed bottom position.

double xLeftRepeat (default = RCMPDFPage.defaultLeft)

Same as xLeft – will be applied to newly created pages by the table, after the first page. You may want this to be different from the first page.

double xRightRepeat (default = RCMPDFPage.defaultRight)

Same as xRight – will be applied to newly created pages by the table, after the first page. You may want this to be different from the first page.

double yTopRepeat (default = RCMPDFPage.defaultTop)

Same as yTop – will be applied to newly created pages by the table, after the first page. You may want this to be different from the first page.

double yBottomRepeat (default = RCMPDFPage.defaultBottom)

Same as xBottom – will be applied to newly created pages by the table, after the first page. You may want this to be different from the first page.

double textPadding (default=0)

To increase the padding around the text in all boxes for the table.

RCMPDFColor titleColor (default = new RCMPDFColor(0.0, 0.0, 0.0))

Color of the table's title **text.** Default is black.

RCMPDFColor titleBGColor (default = new RCMPDFColor(1.0, 1.0, 1.0))

Color of the table's title **background.** Default is white.

RCMPDFColor headingColor (default = new RCMPDFColor(0.5, 0.75,1.0))

Color of the headings text **background**. Default is light blue.

RCMPDFColor headingTextColor (default = new RCMPDFColor(0.0, 0.0,0.0))

Color of the headings **text**. Default is black.

RCMPDFColor rowColor (default = new RCMPDFColor(1.0, 1.0, 1.0))

Color of the row cells' content text **background**. Default is white.

RCMPDFColor altRowColor (Optional, default = null))

Alternative row cells' color content text **background**. Used for every other row in the table. Default is null, indicating that no alternative color is being used.

RCMPDFColor rowTextColor (default = new RCMPDFColor(0.0, 0.0, 0.0))

Color of the row cells' content **text**. Default is black.

RCMPDFColor strokeColor (default = new RCMPDFColor(0.0, 0.0, 0.0))

All **lines** for the table will be drawn in this color. Default is black.

RCMPDFColor titleStrokeColor
(Optional, default = new RCMPDFColor(1.0, 1.0, 1.0))

Optional **outline** color around the title at the top. Default is white.

RCMPDFFont titleFont (Optional)

Font for table title, shown above the table.

RCMPDFFont headingFont (default = new
RCMPDFFont("Courier","BoldOblique", 8))

Font for table headings – top row for each column.

RCMPDFFont rowFont (default = new RCMPDFFont("Courier","Plain", 8))

All cells in the table will be drawn in this font. This can be overridden with **cellFonts**.

NSMutableArray repeatingPageImages (Optional, default = null)
Set of **RCMPDFImages**, which appear in every repeating page created by the table. E.g. an array of logos you want to depict around the table on every page.

boolean repeatingTitle (default = false)
Instruct whether the title repeats or not in the table, for every newly created page by the table.

boolean repeatingHeadings (default = true)
Instruct whether the headings repeat or not in the table, for every newly created page by the table.

double titleSpace (default = 5.0)
Empty space between title and where the actual table starts – default 5 points.

double overFlowMargin (default 0.20 – min = 0.0 and max = 1.0)
Within the bottom 20%, or whatever you specify here, overflow rows will go to the next newly created page, without breaking up the boxes in a table.

Attributes^2 – at cell (box) level detail

The following attributes allow you to set certain values for every value cell in your table, similarly to **values**. The values are stored in two-dimensional arrays: an NSMutableArray of rows holding each an NSMutableArray of such attribute for each cell.

NSArray^2 cellColors (Optional)
Holds a nested array for each cell's **background** color.

NSArray^2 cellTextColors (Optional)
Holds a nested array for each cell's **text** color.

NSArray^2 cellFonts (Optional)
Holds a nested array for each cell's **font**.

NSArray^2 cellHorizontalAlignments (Optional)
Holds a nested array for each cell's **horizontal alignment**.

NSArray^2 cellVerticalAlignments (Optional)
Holds a nested array for each cell's **vertical alignment**.

Methods

public void draw()

> **REQUIRED: MUST BE CALLED AFTER TABLE HAS BEEN DEFINED; OTHERWISE TABLE WILL NOT SHOW UP IN YOUR DOCUMENT! ALTERNATIVELY, YOU MAY CALL drawSelf()**
>
> This method constructs the table consisting of various RCMPDFBoxes over several RCMPDFPages, starting with firstPage. A loop is used through the rows. If a new page is needed, a repeating page is automatically created and added to the document. At the end, the **lastPage**() and **yBottomLast**() variables are set.

public RCMPDFPage lastPage()

Returns the last page, on which the current table ends. The returned page can be used as the first page of the next table or can be used to add other contents in your RCMPDFDocument in combination with **yBottomLast().**

public double yBottomLast()

Returns the y position, where table ends on **lastPage(),** after **draw()** has been invoked. This is where you can pick up control from the table and continue your RCMPDFDocument.

public RCMPDFLocation drawSelf()

This method can be used instead of **draw()** and returns an **RCMPDFLocation** instance. Then you would use that instance instead of **lastPage()** and **yBottomlast()** for location values. For example:

```
protected RCMPDFLocation loc = new table.drawSelf();
double verticalPosition = loc.yPos();
RCMPDFPage currentPage = loc.page():
```

4.5 – RCMPDFImage

This class represents an image (in the types of **jpeg**, **gif** and **png**, and may include more types in later versions) created by given NSData, or a filename of an image, or a url of an image reachable over the internet. For loading efficiency, we recommend limiting the number of images to be used – especially large images.

Constructors

1. Setting position, size, filename, and framework

```
public RCMPDFImage( double xPos,
                    double yPos,
                    double width,
                    double height,
                    String fileName,
                    String framework )
```

2. Setting position, filename, and framework
> image will default to its own size

```
public RCMPDFImage( double xPos,
                    double yPos,
                    String fileName,
                    String framework )
```

3. Setting position, size, and fileName
> framework will default to "app" – your current application project.

```
public RCMPDFImage( double xPos,
                    double yPos,
                    double width,
                    double height,
                    String filename )
```

4. Setting position and fileName
> image will default to its own size
> framework will default to "app" – your current application project.

```
public RCMPDFImage( double xPos,
                    double yPos,
                    String filename )
```

5. Setting position, size, and imageData (from NSData instance)

```
public RCMPDFImage( double xPos,
                    double yPos,
                    double width,
                    double height,
                    NSData imageData )
```

6. Setting position and imageData (from NSData instance)
> image will default to its own size

```
public RCMPDFImage( double xPos,
                    double yPos,
                    NSData imageData )
```

7. Setting imageSourceURL (from web), position, and size

```
public RCMPDFImage( String imageSourceURL,
                    double xPos,
                    double yPos,
                    double width,
                    double height )
```

8. Setting imageSourceURL (from web) and position
> image will default to its own size

```
public RCMPDFImage( String imageSourceURL,
                    double xPos,
                    double yPos )
```

Example: How to create and use an RCMPDFImage

```
myPage.addImage(new RCMPDFImage(10, 50, "rcmlogo.gif" )).
Where: myPage is an instance of RCMPDFPage.
```

Attributes

String fileName (Required if imageSourceURL and imageData not specified)
This attribute specifies the image file name in your project's Web Server Resources or Resources section.

NSData imageData (Required if src and fileName not specified)
This attribute requires image data of the type of **NSData**.
This is useful if your image data is stored in a database.

String imageSourceURL (Required if fileName and imageData not specified)
This attribute specifies the URL (Universal Resource Locator) of an image.
E.g. "http://www.webappz.com/images/webappz_logo.gif"

String framework (default = application.name() or "app")
This attribute specifies the framework to locate file, if the image is located in a framework rather than your application. E.g. "RCMListEdit"

double xPos (default = 0.0)
This attribute gives X, or horizontal, start position of an image in pixels.

double yPos (default = 0.0)
This attribute gives Y, or vertical, start position of an image in pixels.

double width (default = width of the real image if successfully found)
This attribute specifies the width of an image in pixels.

double height (default = height of the real image if successfully found)
This attribute specifies the height of an image in pixels.

4.6 – RCMPDFLine

This class is used to draw a line shape in a PDF file. Future RCMPDFGraph will need lines to draw itself.

Constructors

1. Setting the position of two ends of a line.

```
RCMPDFLine(     double x1,   double y1,
                double x2,   double y2      )
```

2. Setting the position of two ends of a line, and the width of a line.

```
RCMPDFLine(     double width,
                double x1,   double y1,
                double x2,   double y2      )
```

3. Setting the position of two ends of a line, and the color of a line.

```
RCMPDFLine(     RCMPDFColor color,
                double x1,   double y1,
                double x2,   double y2      )
```

4. Setting the position of two ends of a line, width and color of a line.

```
RCMPDFLine(     double width,
                RCMPDFColor color,
                double x1,   double y1,
                double x2,   double y2      )
```

Example: How to create and use RCMPDFLine

```
myPage.addLine(new RCMPDFLine(100, 100, 200, 100 ))
Where: myPage is an instance of RCMPDFPage
```

Attributes

double x1 (required)
X value (the value in x axial) of the starting point of a line in pixels

double y1 (required)
Y value (the value in y axial) of the starting point of a line in pixels

double x2 (required)
X value (the value in x axial) of the ending point of a line in pixels

double y2 (required)
Y value (the value in y axial) of the ending point of a line in pixels

double width (default = 1)
Width of the line in pixels.

RCMPDFColor color (default = black)
Color of the line.

4.7 – RCMPDFPolygon - to be included in future version 2.1

4.8 – RCMPDFCircle - to be included in future version 2.1

4.9 – RCMPDFGraph - to be included in future version 2.2

4.10 – RCMPDFPieChart - to be included in future version 2.2

4.11 – RCMPDFFont

This class defines fonts that may be used when generating PDF. The RCMPDFKit framework currently supports 300 server side fonts with so called Adobe Font Metrics (afm) files. To keep the size of the PDF files compact and for legal reasons – WEBAPPZ may not resell or package fonts it does not own – the actual fonts are not bundled in the PDFKit and are also not embedded in the generated PDF file. As such, the client-side system on which the PDF files are rendered and printed from must have these fonts installed: On Windows, select Control Panel, then Fonts. This provides an options view with currently installed fonts and additional fonts you can install. On Mac OS X, open the Font Book application – this allows you to view and install fonts. If a font is not present on the client-side, it will be substituted with a standard font and the letters may not display correctly. Please run Tutorial 10 from RCMPDFkitTutorials to determine which fonts display correctly and which do not. Every client system, regardless of the operating system, supports at least the four standard PDF Fonts: “Courier”, “Helvetica”, “Times”, and Symbol”.

RCMPDFFont is typically used in **RCMPDFTable** for setting title, heading, and row fonts, or in an **RCMPDFBox** for its text.

Constructors

1. Setting type, style and size

public RCMPDFFont(String type, String style, double size)

2. Setting a PDF base font name and size

public RCMPDFFont(String fontName, double size)

3. Choosing defaults

public RCMPDFFont()

Please note that ‘new RCMPDFFont()’ creates the same font as ‘new RCMPDFFont(“Courier”,12)’ or ‘new RCMPDFFont(“Courier”, “”, 12)’

Examples: How to create and use RCMPDFFonts

```
     protected RCMPDFFont helveticaBold28 = new RCMPDFFont("Helvetica","Bold", 28);
     protected RCMPDFFont timesItalic18 = new RCMPDFFont("Times","Italic", 18);
or
     protected RCMPDFFont helveticaBold28 = new RCMPDFFont("Helvetica-Bold", 28);
     protected RCMPDFFont timesItalic18 = new RCMPDFFont("Times-Italic", 18);
```

Attributes

double size (default = 12)

This attribute is the size of this font

String type (default = "Courier")

This attribute is the font type of this font. Other values could be “Helvetica”, “Times_Roman” or “Symbol”. The values are not “case sensitive”.

String style (default = “”)

This attribute specifies the font style. Other values could be “Bold”, “Oblique”, “Italic”, or “BoldOblique”. The values are not “case sensitive”.

Instance Methods

double getStringWidth(String)

This method calculates the width in points for a specific String using an instance of RCMPDFFont by using font metrics data found in an associated afm file. Please use this method instead of similar methods from the standard Java Font class as it provides much better significance. The kit uses this method for its very precise text wrapping capabilities.

double getCharWidth(char)

Same as above, but for individual characters.

Static Methods

static void listAvailableFonts()

This method lists all type and style combinations to the standard console output. This list is based on the afm files present in the the directory:
$/Library/Frameworks/RCMPDFKit.framework/Resources/afm/

static String[] afmFilesList()

Same as above, but results will be returned in an array of String.

Standard Supported PDF Fonts:

The following type and style combinations are guaranteed to work on any PDF Client:

Type	Style	Sample	**Bold**	**Italic**
"Courier"	""	RCMPDFKit		
"Courier"	"Bold"	**RCMPDFKit**	✔	
"Courier"	"Oblique"	*RCMPDFKit*		✔
"Courier"	"BoldOblique"	***RCMPDFKit***	✔	✔
"Helvetica"	""	RCMPDFKit		
"Helvetica"	"Bold"	**RCMPDFKit**	✔	
"Helvetica"	"Oblique"	*RCMPDFKit*		✔
"Helvetica"	"BoldOblique"	***RCMPDFKit***	✔	✔
"Times"	"Roman"	RCMPDFKit		
"Times"	"Bold"	**RCMPDFKit**	✔	
"Times"	"Italic"	*RCMPDFKit*		✔
"Times"	"BoldItalic"	***RCMPDFKit***	✔	✔
"Symbol"	""	ΡΧΜΠΔΦΚιτ		

Current Supported Non-Standard Fonts:

The following list shows all the type and style combinations for which afm files are bundled with the RCMPDFKit framework (currently 300). In order to support more combinations, the developer is advised to add new afm files to the directory: $Library/Frameworks/RCMPDFKit.framework/Resources/afm/ WEBAPPZ can also provide afm files at developer's requests. Newly inserted files will immediately be recognized by the kit.

Calling **RCMPDFFont.listAvailableFonts()**, without any font additions, displays:

```
###*## RCMPDFKit: LIST OF AVAILABLE FONTS

-------------------------------------------------------------
|       FONT TYPE        | STYLES AVAILABLE FOR THIS FONT TYPE
-------------------------------------------------------------
|               AbadiMT | "CondensedExtraBold" , "CondensedLight"
|             Abduction | ""
|       ACMESecretAgent | ""
|    ACMESecretAgentBold | ""
|  ACMESecretAgentItalic | ""
|             AdamsHand | "Plain"
|       AdventureNormal | ""
|              AgencyFB | "Bold" , "Reg"
|     AmericanTypewriter | "Bold", "Condensed", "CondensedBold", "CondensedLight", "Light",""
|            AndaleMono | ""
|                 Apple | "Chancery"
|                 Arial | "Black" , "BlackItalic" , "BoldItalicMT" , "BoldMT" , "ItalicMT"
|               ArialMT | ""
|           ArialNarrow | "Bold" , "BoldItalic" , "Italic" , ""
|     ArialRoundedMTBold | ""
|           Baskerville | "Bold" , "BoldItalic" , "Italic", "SemiBold", "SemiBoldItalic", ""
|           BaskOldFace | ""
|             Bauhaus93 | ""
|        BeatnikHayseed | ""
|               BellMT | ""
|            BellMTBold | ""
|          BellMTItalic | ""
|             BernardMT | "Condensed"
|            BerrysHand | "Plain"
|             BigCaslon | "Medium"
|             BlackWolf | ""
|           BookAntiqua | "Bold" , "BoldItalic" , "Italic" , ""
|       BookmanOldStyle | "Bold" , "BoldItalic" , "Italic" , ""
|           Braggadocio | ""
|           BrandysHand | "Plain"
|         BritannicBold | ""
|         BrushScriptMT | ""
|            CalisMTBol | ""
|             CalistoMT | "BoldItalic" , "Italic" , ""
|           CartersHand | "Plain"
|               Centaur | ""
|               Century | ""
|         CenturyGothic | "Bold" , "BoldItalic" , "Italic" , ""
|      CenturySchoolbook | "Bold" , "BoldItalic" , "Italic" , ""
|             Chalkboard | ""
|                 Cochin | "Bold" , "BoldItalic" , "Italic" , ""
|              ColonnaMT | ""
|           ComicSansMS | "Bold" , ""
|           CooperBlack | ""
|           Copperplate | "Bold" , "Light" , ""
|      CopperplateGothic | "Bold" , "Light"
```

```
|               Courier | "Bold" , "BoldOblique" , "Oblique" , ""
|          CourierNewPS | "BoldItalicMT" , "BoldMT" , "ItalicMT"
|         CourierNewPSMT | ""
|              Cuomotype | ""
|                CurlzMT | ""
|              Desdemona | ""
|                Destroy | ""
|                  Didot | "Bold" , "Italic" , ""
|               DirtyEgo | ""
|     EdwardianScriptITC | ""
|           EngraversMT | "Bold" , ""
|          EurostileBold | ""
|       EurostileRegular | ""
|         FelixTitlingMT | ""
|       FootlightMTLight | ""
|      ForgottenFuturist | "Bold" , "BoldItalic" , "Italic" , ""
|                ForteMT | ""
|         FranklinGothic | "Medium" , "MediumItalic"
|                 Futura | "CondensedExtraBold" , "CondensedMedium" , "Medium" ,
"MediumItalic"
|               Garamond | "Bold" , "Italic" , ""
|                Geneva | ""
|               Georgia | "Bold" , "BoldItalic" , "Italic" , ""
|                  Gigi | "Regular"
|               GillSans | "Bold" , "BoldItalic" , "Italic" , "Light" , "LightItalic" ,
"UltraBold" , ""
|           GloucesterMT | "ExtraCondensed"
|         GoudyOldStyleT | "Bold" , "Italic" , "Regular"
|       Haettenschweiler | ""
|            HarmonsHand | "Plain"
|             Harrington | ""
|              Helvetica | "Bold" , "BoldOblique" , "Oblique" , ""
|          HelveticaNeue | "Bold" , "BoldItalic" , "CondensedBlack" , "CondensedBold" ,
"Italic" , "Light" , "LightItalic" , "UltraLight" , "UltraLightItalic" , ""
|             Herculanum | ""
|            HoeflerText | "Black" , "BlackItalic" , "Italic" , "Regular"
|            Hulkbusters | ""
|                Impact | ""
|              ImprintMT | "Shadow"
|              Inkburrow | ""
|            JessiesHand | "Plain"
|               Jokerman | "Regular"
|               JuiceITC | "Regular"
|             KeithsHand | "Plain"
|                KinoMT | ""
|         KunstlerScript | ""
|              LatinWide | ""
|      LucidaBlackletter | ""
|           LucidaBright | "Demi" , "DemiItalic" , "Italic" , ""
|      LucidaCalligraphy | "Italic"
|          LucidaConsole | ""
|              LucidaFax | "Demi" , "DemiItalic" , "Italic" , ""
|           LucidaGrande | "Bold" , ""
|      LucidaHandwriting | "Italic"
|             LucidaSans | "Demi" , "DemiItalic" , "Italic" , "Typewriter" , "TypewriterBold"
, "TypewriterBoldOblique" , "TypewriterOblique" , ""
|      LucidaSansUnicode | ""
|             MarkerFelt | "Thin" , "Wide"
| MaturaMTScriptCapitals | ""
|           MichaelsHand | "Plain"
|     MicrosoftSansSerif | ""
|              Millennia | ""
|                Mistral | ""
|                 Modern | "Regular"
|                 Monaco | ""
|        MonotypeCorsiva | ""
|                     MT | "Extra"
|            MyriadApple | "Bold" , "BoldItalic" , "Medium" , "MediumItalic" , "Semibold" ,
"SemiboldItalic" , "Text" , "TextItalic"
```

```
|           NewsGothicMT | "Bold" , "Italic" , ""
|                  Onyx | ""
|                Optima | "Bold" , "BoldItalic" , "ExtraBlack" , "Italic" , "Regular"
|             OrangeFizz | ""
|       OrangeFizzItalic | ""
|       PalatinoLinotype | "Bold" , "BoldItalic" , "Italic" , "Roman"
|                Papyrus | ""
|      PerpetuaTitlingMT | "Bold" , "Light"
|              Playbill | ""
|             RageItalic | ""
|                RedFive | ""
|              Rockwell | "Bold" , "BoldItalic" , "ExtraBold" , "Italic" , ""
|             RustysHand | "Plain"
|                  Shag | "Exotica" , "ExoticaExtras" , "ExoticaFractions" , "ExoticaSwash"
, "Lounge" , "LoungeExtras" , "LoungeFractions" , "Mystery" , "MysteryExtras" ,
"MysteryFractions" , "MysterySwash" , "Shagbats"
|             ShagExpert | "Exotica" , "Lounge" , "Mystery" , "Shagbats"
|                  Skia | "Regular"
|               Stencil | ""
|             StixnStonz | ""
|                Symbol | ""
|              SymbolMT | ""
|                Tahoma | "Bold" , ""
|                  Teen | "Bold" , "BoldItalic" , "Italic" , ""
|              TeenLight | "Italic" , ""
|                 Times | "Bold" , "BoldItalic" , "Italic" , "Roman"
|        TimesNewRomanPS | "BoldItalicMT" , "BoldMT" , "ItalicMT"
|      TimesNewRomanPSMT | ""
|              TrajanPro | "Regular"
|              Trebuchet | "BoldItalic"
|            TrebuchetMS | "Bold" , "Italic" , ""
|                Verdana | "Bold" , "BoldItalic" , "Italic" , ""
|               Vivaldii | ""
|         VladimirScript | ""
|            WalkersHand | "Plain"
|             WaynesHand | "Plain"
|               Webdings | ""
|              Wingdings | "Regular" , ""
|             Wingdings2 | ""
|             Wingdings3 | ""
|              YanksHand | "Plain"
|           YearBookMess | ""
|            ZonkersHand | "Plain"
-------------------------------------------------------------
```

4.12 - RCMPDFColor

RCMPDFColor can be used as font color, stroke color and fill color for RCMPDFBox, and several combinations in RCMPDFTable.

Constructors

RCMPDFColor(double redValue, double greenValue, double blueValue)

public RCMPDFColor(String html)

This constructor allows you to create a new color based on a standard HTML color code. Note: all codes **must** start with **#**.

Examples: How to create RCMPDFColors

```
protected RCMPDFColor red = new RCMPDFColor(1.0,0.12,0.02); // double RGB
protected RCMPDFColor white = new RCMPDFColor(1.0,1.0,1.0); // double RGB
protected RCMPDFColor veryLightGrey = new RCMPDFColor("#f0f2f4"); // HTML
protected RCMPDFColor lightGrey = new RCMPDFColor("#e0e2f4"); // HTML
protected RCMPDFColor grey = new RCMPDFColor("#a2a5a6"); // HTML
protected RCMPDFColor darkBlue = new RCMPDFColor("#0c4989"); // HTML
```

Attributes

double **redValue** (Required)
Red component of a color - RGB Model - from 0.000 to 1.000.
double **greenValue** (Required)
Green component of a color - RGB Model - from 0.000 to 1.000.
double **blueValue** (Required)
Blue component of a color - RGB Model - from 0.000 to 1.000.

4.13 – RCMPDFLocation

RCMPDFLocation can be used to identify the location within an **RCMPDFDocument**.

Constructors

public RCMPDFLocation(RCMPDFPage page, double yPos)

public RCMPDFLocation(RCMPDFPage page, double xPos, double yPos)

The first constructor assumes a left most position on the page: **xPos** = 0

Example: How to create an RCMPDFLocation

```
protected RCMPDFLocation aLocation =
         new RCMPDFLocation(currentPage,yPos); // assume drawing finished
                                               // on currentPage at yPos
```

Example: How to use an RCMPDFLocation

```
protected location = statement.draw(); // assume our method 'statement'
                                       // returns its last bottom position
RCMPDFPage currentPage = location.page();
double y =  location.yPos();
```

Attributes

RCMPDFPage page (Required)
Identifies a specified page within an **RCMPDFDocument**.

double xPos (Optional; default = 0.0)
Horizontal position on the **page**.

double yPos (Required)
Vertical position on the **page**.

5.1 – Support

Need help:

visit us at http://www.webappz.com/products/pdfkit/index.html, or
email us at support@webappz.com

Although rights to the source code are not included, WEBAPPZ is open to any suggestions for continuous framework improvements. Updated versions will be released regularly.

- www.webappz.com -

WEBAPPZ Systems, Inc.
726 – 1489 Marine Drive
West Vancouver, BC
V7T 1B8 CANADA
604.921.1333