Documentation and Programmer’s Reference

ACI RTF Library™

A Programmer’s Tool for Generating Professional Rich Text Format (RTF) Documents and Reports for

ASP 5 (and up)
Coldfusion 5
ColdFusion MX

Version 20050408

Edited by Douglas M. Smith

Table of Contents

Copyright and License

License Agreement

This License Agreement ("Agreement") governs your access to and use of the ACI RTF Library™.

[By clicking the "Accept" button, you accept all the terms of this binding Agreement between you, and Automation Creations, Inc., owner of the ACI RTF Library™. If you do not agree with the terms of this Agreement, you may click your "back" button now and stop the registration process. Clicking "Accept" constitutes acceptance of the terms of this Agreement. If you do not accept the terms of this Agreement, you will not be permitted to access or use the ACI RTF Library™.]

Grant of License: You acknowledge that the ACI RTF Library™ is owned by Automation Creations, Inc. Subject to the terms and conditions of this Agreement, Automation Creations, Inc. hereby grants you a personal, limited, nonexclusive, nontransferable, nonsublicensable license to use the ACI RTF Library™ solely from one computer, running as a web server, under your control.

Limitations: The ACI RTF Library™ is licensed, not sold, to you. You agree to retain all copyright and related notices of Automation Creations, Inc. included with and on the ACI RTF Library™. You may use the ACI RTF Library™ solely for your own personal and business purposes. You may not: (a) modify, translate, or reproduce the ACI RTF Library™, unless it is for your own personal or internal business purposes; (b) sell or redistribute the ACI RTF Library™; (c) collect, organize or catalog the ACI RTF Library™ for the purpose of creating a database unless it is for your own personal or internal business purposes provided that such personal database shall never be resold or released for public use; (d) sublicense, rent, lend, transfer, post, transmit, or otherwise make the ACI RTF Library™ available to anyone else; (f) mass export the ACI RTF Library™; and (g) use the ACI RTF Library™ in any manner which is defamatory, pornographic or otherwise violates any applicable laws. This Agreement and your rights under this Agreement shall automatically terminate upon any failure by you to comply with its terms. This Agreement may only be modified in writing by an authorized officer of Automation Creations, Inc.

Trademarks: You acknowledge that Automation Creations™, ACI RTF Library™, and related logos, and designs are trademarks of Automation Creations, Inc. and that no rights in such trademarks are granted to you by this Agreement.

Modifications: Automation Creations, Inc. reserves the right to modify the content of the ACI RTF Library™ from time to time without obligation to notify you, or any other person or organization of such revision or change.

Limitation of Liability: IN NO EVENT WILL AUTOMATION CREATIONS, INC. BE LIABLE FOR ANY DAMAGES, INCLUDING LOSS OF DATA, LOST OPPORTUNITY OR PROFITS, COST OF COVER OR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, DIRECT OR INDIRECT DAMAGES ARISING FROM OR RELATING TO THE USE OF THE ACI RTF LIBRARY™, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. THIS LIMITATION WILL APPLY EVEN IF AUTOMATION CREATIONS, INC HAS BEEN ADVISED OR GIVEN NOTICE OF THE POSSIBILITY OF SUCH DAMAGE. THE ENTIRE RISK AS TO THE USE OF THE ACI RTF LIBRARY™ IS ASSUMED BY YOU THE USER. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CERTAIN INCIDENTAL, CONSEQUENTIAL OR OTHER DAMAGES, THIS LIMITATION MAY NOT APPLY TO YOU.

Disclaimer of Warranty: TO THE EXTENT PERMITTED BY APPLICABLE LAW THE ACI RTF LIBRARY™, INCLUDING BUT NOT LIMITED TO ALL DATA, TOOLS AND CALCULATIONS THEREIN ARE PROVIDED "AS IS" AND WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND BY EITHER AUTOMATION CREATIONS, INC. OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THE ACI RTF LIBRARY™, INCLUDING BUT NOT LIMITED TO ANY EXPRESS OR IMPLIED WARRANTY OF MERCHANTABILITY, ACCURACY, QUIET ENJOYMENT, NONINFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE. NO COVENANTS, WARRANTIES OR INDEMNITIES OF ANY KIND ARE GRANTED BY AUTOMATION CREATIONS, INC. TO YOU THE USER.

Termination: If you do not accept the terms of this Agreement, you agree to click your "back" button and stop the registration process. If you do not accept the terms of this Agreement, you will not be permitted to access or use the ACI RTF Library™.

Government Rights: If used or acquired by the Government, the Government acknowledges that (a) the ACI RTF Library™ constitute "commercial computer software" or "commercial computer software documentation" for purposes of 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-4, as applicable and (b) the Government's rights are limited to those specifically granted pursuant to this Agreement. The contractor/manufacturer is AUTOMATION CREATIONS, INC, 2000 Kraft Drive, Suite 1202, Blacksburg, VA 24060.

Governing Law and General Provisions: This Agreement will be governed by the laws of the State of Virginia, U.S.A., excluding the application of its conflicts of law rules. This Agreement will not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded. If any part of this Agreement is found void and unenforceable, it will not affect the validity of the balance of the Agreement, which shall remain valid and enforceable according to its terms.

Export Control Obligations: You will not export or re-export the ACI RTF Library™ and any associated software in violation of any law, regulation, order or other governmental requirement (including, without limitation, the U.S. Export Administration Act, regulations of the Department of Commerce and other export controls of the U.S.). You shall, at your own expense, promptly obtain and arrange for the maintenance of all non-U.S.A. government approvals, if any, and comply with all applicable local laws and regulations as may be necessary for performance under this Agreement.

Introduction

The ACI RTF Library is designed for use by programmers who want to generate Rich Text Format (RTF) documents using ColdFusion or ASP scripting language.

RTF is notoriously hard to generate manually, and there is no easy way to debug it. A single out of place character will cause the document to simply fail to open or render in your word processor.

This library allows the programmer to easily assemble sophisticated and elegant RTF content without having to know how RTF works.

How it works

The ACI RTF Library is setup to write formatted RTF content to a script language string variable. After the RFT content is assembled, the string can be written to a file, or streamed to the user's web browser directly, using MIME type encoding. Documents can be viewed an edited using any word processor or editor that can read RTF format.

Features

- Pure ASP and ColdFusion scripted code, for plug and play functionality. No DLL, CFX or compiled code to install.

- Unencrypted source code, so you can modify the library to suite your purposes (for internal uses only; resale/redistribution of the code library is not covered by the license).

- Generates RTF tables and graphics: Handles embedded RTF images: inline, absolute placement and document watermarks.

- Use measurements of choice (TWIPS, inches, or lines).

- Predefined constants for options, and using web colors.

System Requirements

ColdFusion
- Version 5, or MX

ASP
- ASP Version 5 and above
- ADO Version 2.5 and above

Advantages of RTF Format

Non-Proprietary Document Format: RTF documents can be opened and/or viewed in a variety of word processor applications and text editors, including MS Word.

Predictable Layout and Format: Like MS Word and PDF documents, the programmer can generate professionally formatted and stylized documents, with specific and exacting measurement requirements.

Small and Efficient: RTF has a MUCH smaller file size than either native MS Word document, PDF documents, OR event RTF generated from MS Word. This means faster downloads and less load on the server.

Some common uses:

- Executive Reports
- Merge letters and labels
- Form Letters
- Standard Printable Forms
- Convert messy HTML content to a predicable and nicely printable document.

Technical Support

ACI does not offer live (via phone) technical support for the ACI RTF Library. We have code samples and information on our web sites:

- ColdFusion: http://rtflib.aciwebs.com/CF/

- ASP: http://rtflib.aciwebs.com/ASP/

We do offer email support, and will accept emails describing issues and problems you are having. Send to: rtflib@aciwebs.com

We can offer custom assistance at an hourly rate. We have designed sophisticated reports with this library, and we can create your reports also. Please contact us for details. http://www.aciwebs.com/contact/contact.htm

Microsoft Word and Limitations with other RTF Editors

Microsoft created the RTF Specifications independently of MS Word, but since most people will open their RTF documents in MS Word, MS Word is the de facto standard for rendering RTF content.

This RTF Library has been tested to work with MS Word 2003 and previous version. RTF documents generated by this library cannot be guaranteed to work in later versions of MS Word, and should be tested first.

While we expect general cross-RTF-editor functionality, other RTF editors may not be able to render all of the RTF objects generated by the library. Therefore, you will need to test and develop your RTF output within the limitations of whatever RTF editor or viewer your end-users will be using.

For your reference, the official Microsoft RTF Specifications document is contained in this distribution:

Word 2002 RTF Specification Final.doc

Documentation Standards

Example code will be formatted like this:

```
<% This is some sample code %>
```

Features that only apply to CF will be indicated like so: **CF ONLY**

Features that only apply to ASP will be indicated like so: **ASP ONLY**

Installation

The files in your software zip file should unzip to a directory called "RTFLIB_CF" or "RTFLIB_ASP". Unzip directly to your web site, or copy this directory to your web site. This directory should contain:

- Library → main RTF Library
- Examples → code examples

Feel fee to copy the main RTFLIB_20050128.CFM/.ASP file to a location that is more convenient for your web application.

You should be able to run the sample code files in the "/examples/" directory without any modifications to the files.

Rules to Follow/Things to Know

1. **There are generally two kinds of functions in this library.**

   - **`Register…()`** functions set up information or settings that will modify what RTF code will be written.

   - **`Draw…()`** (or `QuickDraw()`) functions actually return a string that can be written to the RTF file, or concatenated with other strings.

   Normally for complex operations you will use the `Register…()` functions to set up the attributes of the RTF to write and then the `Draw…()` function to produce output.

2. **Always include an RTF Header and Footer:**

   - **DrawRTFHeader()** or **QuickDrawRTFHeader()** must always be used. The RTF that it generates must be at the start of your generated content string.

   - **DrawRTFFooter()** or **QuickDrawRTFFooter()** must always be used. The RTF that it generates must be at the **end** of your generated content string.

3. **Use the RTF Constants**

   Most functions expect to be passed predefined RTF CONTSTANTS, when formatting or layout specifications are called for. These are all defined in the **RTF Constants** section of this doc.

4. **Using Optional Parameters**

   Optional function parameters are indicated with square brackets:

   Function MyFunction(requiredparam, [optionalparam])

   - In **ColdFusion**, optional parameters may be dropped from the function call. However, the parameters are place-sensitive. For example, if you want to use optional parameter 5, then you must include optional parameters 1 – 4 before you can use 5.

   - In **ASP/VB Script** optional parameters that you wish to leave out must be replaced with the reserved word "null".

5. Function parameters have these naming conventions:

Starts with	Means
CONST_	Provide one of the RTF constants defined in the RTF Constants section of this doc.
i	Provide an integer value
f	Provide a real number value
s	Provide a string value
b	Provide a Boolean (TRUE or FALSE) value

6. Register Functions

In **ASP/VB Script**, all functions starting with the word **Register** (for example: RegisterParagraphIndents()) are actually Subs, and should be called as Subs. In ASP, a sub does not return anything, but you will get an error if you use parens for the parameters, so call them like this:

RegisterParagraphIndents param1, param2, param3

You can also subs with parens, using the CALL statement:

Call RegisterParagraphIndents (param1, param2, param3)

Exception: RegisterImage() and RegisterImageCache() are functions, not subs.

Getting Started

For both ColdFusion and ASP, the RTF Library is intended to be used as a simple include. The entire function library is contained in the RTFLIB.CFM or RTFLIB.ASP file. The following examples assume you have put your RTF library file in a directory named "library" relative to the calling page.

ColdFusion Example

```
<cfinclude template="library/rtflib_20050128.cfm">
```

ASP Example

```
<!-- #INCLUDE FILE="library/rtflib_20050128.asp" -->
```

Generate Basic RTF Output

In this example, the RTF content is being built and appended to a string called RTF_OUTPUT. It does not generate an RTF document, but just shows how the RTF content is assembled.

ColdFusion Example

```
<cfinclude template="library/rtflib_20050128.cfm">

<cfscript>
RTF_OUTPUT = "";
RTF_OUTPUT = QuickDrawRTFHeader(RTF_LANDSCAPE,"1","1","1","1");
RTF_OUTPUT = RTF_OUTPUT & DrawChars("Hello World!");
RTF_OUTPUT = RTF_OUTPUT & QuickDrawRTFFooter();
</cfscript>

<CFOUTPUT>#RTF_OUTPUT#</CFOUTPUT>
```

ASP Example

```
<!-- #INCLUDE FILE="library/rtflib_20050128.asp" -->
<%

dim RTF_OUTPUT

RTF_OUTPUT = ""
RTF_OUTPUT = QuickDrawRTFHeader(RTF_LANDSCAPE, "1", "1", "1", "1", NULL, NULL,
       NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
RTF_OUTPUT = RTF_OUTPUT & DrawChars("Hello World!",NULL, NULL, NULL, NULL)
RTF_OUTPUT = RTF_OUTPUT & QuickDrawRTFFooter()

Response.Write RTF_OUTPUT

%>
```

Stream RTF Content directly to the browser

This code will send the RTF content directly to the user's web browser. If you have not configured your browser to handle RTF automatically, you should get a window that has options like [Open] [Save As…] [Cancel]

Important: When streaming content to the browser, do not include any HTML output (do not add <HTML><body> tags, for example.) HTML output will cause the browser to render the RTF content as HMTL, instead of as RTF content.

ColdFusion Example

```
{…START OF DOC - NO HTML ABOVE HERE }
<cfsilent>

<cfinclude template="library/rtflib_20050128.cfm">

<cfscript>
{…code to generate RTF content goes here…}
</cfscript>

<cfcontent type="application/rtf">
<cfheader name="content-disposition" value="filename=hello_world.rtf" >
<cfoutput>#RTF_OUTPUT#</cfoutput>
{…END OF DOC - NO HTML BELOW HERE }
```

ASP Example

```
{…START OF DOC - NO HTML ABOVE HERE }
<!-- #INCLUDE FILE="library/rtflib_20050128.asp" -->
<%

dim RTF_OUTPUT

{…code to generate RTF content goes here…}

Response.ContentType="application/rtf"
Response.AddHeader "content-disposition", "filename=sample.rtf"
Response.Write RTF_OUTPUT

%>
{…END OF DOC - NO HTML BELOW HERE }
```

Save RTF Content to a file

When saving RTF content to a file, the page can contain normal HTML output to the browser, if you wish. This can include a link to the newly created file, or JavaScript to redirect the user to the newly created file, for example.

ColdFusion Example

```
<cfinclude template="library/rtflib.cfm">

<cfscript>

{…code to generate RTF content goes here…}

filename = "sample_writetofile.rtf";
filespec = ExpandPath(filename);
</cfscript>

<cffile action = "write" file = "#filespec#" output = "#RTF_OUTPUT#">
```

ASP Example

```
dim RTF_OUTPUT, filename, filespec
dim objFSO, objTextStream

{…code to generate RTF content goes here…}

filename = "sample_writetofile.rtf"
filespec = Server.MapPath(filename)

Set objFSO = CreateObject("Scripting.FileSystemObject")
set objTextStream = objFSO.OpenTextFile (filespec , 2, TRUE)
objTextStream.Write(RTF_OUTPUT)
objTextStream.Close
SET objFSO = nothing
```

More Detailed Examples

More complete examples and demonstrations can be found at:

ColdFusion:

http://rtflib.aciwebs.com/CF/

ASP:

http://rtflib.aciwebs.com/ASP/

Best Practices

1. Read the "Rules To Follow/Things to Know" section of this document.

2. Keep your RTF output clean, and rely on the RTF Library functions to do the work. Don't try appending your own RTF code or special characters, unless you really know what you are doing. RTF is notoriously easy to break, and there are no error messages returned to explain why MS Word won't open or render your RTF document.

3. Never mix HTML and RTF content, for the same reason as #1.

4. Assemble the RTF content string first, and then output it, or save it to a file. The RTF assembly code could be isolated in an include or custom module.

5. For convenience, set default measurements to Inches right at the start of your RTF processing. Otherwise the default measurement is Twips, for some function attributes, which might produce very small or nearly invisible output, since there are 1440 twips to the inch.

   SetVerticalMeasure("inches");
   SetHorizontalMeasure("inches");

6. For convenience, use a simple concatenate function to simplify building the RTF content string.

CF Example:

```
<cfscript>
RTF_Content = ""
function Append(iString) {
      RTF_Content = RTF_Content & iString;
}
</cfscript>
```

Then, you can write:

```
Append(DrawChars("This is some text."));
```

Instead of:

```
RTF_Content = RTF_Content & DrawChars("This is some text.");
```

ASP Example:

```
Dim RTF_OUTPUT
sub Append(strContent)
      RTF_OUTPUT = RTF_OUTPUT & strContent
end sub
```

Then, you can write:

```
Append DrawChars("This is some text.", null, null, null, null)
```

Instead of:

```
RTF_OUTPUT = RTF_OUTPUT & DrawChars("This is some text.", null, null, null,
      null)
```

RTF Constants

Most of these constants are used by various functions in the library. Some may be used when assembling text to be passed into a function. Please refer to this section when any functions indicate an RTF constant or parameter.

Page or Document Layout

The following have been defined for convenience:

RTF_PORTRAIT................	Display/print the page in portrait mode
RTF_LANDSCAPE............	Display/print the page in landscape mode
RTF_PROTECTED	The page is read-only (locked, not editable)
RTF_EDITABLE................	The page is editable
RTF_CENTER_ALIGN	center what follows
RTF_PAGE_BREAK	page break
RTF_SECTION_BREAK...	section break, for when a page break won't work. *
RTF_PAGE_NUMBER......	will render the current page number.
RTF_PAGE_TOTAL..........	will render the total number of pages in the RTF document.**

* There are some occasions, such as after some tables, where a page break won't break the page. In that case, try a section break.

** The document must be EDITABLE for RTF_PAGE_TOTAL to work. You must use the RTF_EDITABLE constant with either DrawRTFHeader() or QuickDrawRTFHeader(). Documents are locked by default.

Text Layout

The following have been defined for convenience:

RTF_LINE_BREAK...........	break a line within a paragraph, or add an extra line.
RTF_PARAGRAPH	another way to add an extra line.
RTF_TAB	An RTF tab code. Text will be tabbed to the tabs set in RegisterParagraphTabs() function.

Fonts

There are three fonts defined in the RTF library:

RTF_FONT_TIMES...........	Times New Roman
RTF_FONT_ARIAL...........	Arial
RTF_FONT_COURIER	Courier New

Font Attributes

These fonts can have attributes to modify their appearance. These attributes should be added together (RTF_BOLD + RTF_ITALIC) when being passed to a function.

```
RTF_BOLD
RTF_ITALIC
RTF_UNDERLINE
```

Font Colors

Web style colors (HEX numbers) cannot be used for fonts, and they must be pre-defined. There are two types of colors used in RTF files. Text and cell shading colors come from a color table. The default color table in our RTF library has 17 entries:

```
RTF_AUTO
RTF_BLACK
RTF_BLUE
RTF_CYAN
RTF_GREEN
RTF_PURPLE
RTF_RED
RTF_YELLOW
RTF_WHITE
RTF_DARK_BLUE
RTF_LIGHT_CYAN
RTF_LIGHT_GREEN
RTF_LIGHT_PURPLE
RTF_LIGHT_RED
RTF_LIGHT_YELLOW
RTF_DARK_GRAY
RTF_LIGHT_GRAY
```

You may add your own colors to the color table using the `DrawRTFHeader()` function. Each color will be numbered sequentially in order starting with RTF_USER_COLOR (17).

The other colors, for shapes, use an extremely confusing RTF color scheme that we've chosen to ignore. Instead, all our functions use web colors (FFD0B0, etc.) and convert them for you. Unfortunately, you cannot use web colors with text.

Line and Border Formats

The following line format variables have been defined:

```
RTF_SOLID
RTF_DASH_WIN
RTF_DOT_WIN
RTF_DASH_TO_WIN
RTF_DASH_DOT_DOT_WIN
RTF_DOT, RTF_DASH
RTF_LONG_DASH
RTF_DASH_DOT
RTF_LONG_DASH_DOT
RTF_LONG_DASH_DOT_DOT
```

Wrapping

The following wrapping constants have been defined fox text boxes and images:

```
RTF_WRAP_TOP_BOTTOM – wrap top and bottom of image but not beside
RTF_WRAP_AROUND_BOTH – wrap around image on both sides
RTF_WRAP_AROUND_LEFT – wrap around image on left
RTF_WRAP_AROUND_RIGHT – wrap around image on right
RTF_WRAP_AROUND_LARGEST – wrap around image on largest side
RTF_WRAP_NONE – no wrapping, text ignores image
RTF_WRAP_TIGHT_BOTH – wrap tight on both sides
RTF_WRAP_TIGHT_LEFT – wrap tight on left side
RTF_WRAP_TIGHT_RIGHT – wrap tight on right side
RTF_WRAP_TIGHT_LARGEST – wrap tight on largest side
RTF_WRAP_THROUGH – text goes through image
```

Shapes

The following constants have been defined for use by the DrawFigure() and DrawTextFigure() functions:

```
RTF_BOX
RTF_BOX_ROUNDED
RTF_CIRCLE
RTF_DIAMOND
RTF_TRIANGLE
```

```
RTF_TRIANGLE_SQUARE
RTF_PARALLELOGRAM
RTF_TRAPEZOID
RTF_HEXAGON
RTF_OCTOGON
RTF_CROSS
RTF_STAR
RTF_ARROW
RTF_ARROW_FAT
RTF_BOX_3D
RTF_BOX_CALLOUT
RTF_STARBURST
RTF_BOX_INDENTED_CORNERS
RTF_CYLINDER
RTF_WASHER
```

RTF Functions (by Type)

These functions are used to create the RTF file structures and set up the default measurement units. Make sure you have read the "Rules To Follow/Things to Know" section of this document. There is important information there.

Document Functions

DrawRTFHeader(sTitle, sSubject, sAuthor, sCompany, sComments, CONST_FONT_FACE, iDefaultFontSize, CONST_PAGE_ORIENTATION, fPageWidth, fPageHeight, fMargLeft, fMargRight, fMargTop, fMargBottom, CONST_PROTECTION,[aColors])

sTitleTitle	that is displayed in the Properties of the document
sSubject	Subject that is displayed in the Properties of the document
sAuthor	Author that is displayed in the Properties of the document
sCompany	Company that is displayed in the Properties of the document
sComments	Comments that are displayed in the Properties of the document
CONST_FONT_FACE	default font for document (see RTF Constants)
fDefaultFontSize	default font size for the document in points
CONST_PAGE_ORIENTATION	landscape or portrait mode (see RTF Constants). **Note: This only affects the print mode.** You must also swap the fPageWidth and fPageHeight variables to get a true landscape printout.
fPageWidth	page width in inches
fPageHeight	page height in inches
fMargLeft	left margin in inches
fMargRight	right margin in inches
fMargTop	top margin in inches
fMargBottom	bottom margin in inches
CONST_PROTECTION	Use RTF_PROTECTED or RTF_EDITABLE constant. (see RTF Constants).
[aColors]	array of structures that define additional colors to add to the color table

aColors[index].val = color value (starting with RTF_USER_COLOR, for your documentation only)
aColors[index].red = red value
aColors[index].green = green value
aColors[index].blue = blue value

This function is required to be called, and is the first Draw…() function that must be called for the RTF document. This function returns an RTF header for a report.

Note: this is not a page header, but the header of the RTF file itself.

QuickDrawRTFHeader(CONST_PAGE_ORIENTATION, fMargLeft, fMargRight, fMargTop, fMargBottom)

CONST_PAGE_ORIENTATION	landscape or portrait mode (RTF_LANDSCAPE or RTF_PORTRAIT) (see RTF Constants)
fMargLeft	left margin in inches
fMargRight	right margin in inches
fMargTop	top margin in inches
fMargBottom	bottom margin in inches
[CONST_PROTECTION]	Use RTF_PROTECTED or RTF_EDITABLE constant. (see RTF Constants)
[CONST_FONT_FACE]	default font for page, an RTF Font constant (see RTF Constants)
[fDefaultFontSize]	default font size for page
[sTitle]	Title that is displayed in the Properties of the document
[sSubject]	Subject that is displayed in the Properties of the document
[sAuthor]	Author that is displayed in the Properties of the document
[sCompany]	Company that is displayed in the Properties of the document
[sComments]	Comments that are displayed in the Properties of the document
[fPageWidth]	page width in inches
[fPageHeight]	page height in inches
[aColors]	array of structures that define additional colors to add to the color table

aColors[index].val	USER defined color index
aColors[index].red	red value
aColors[index].green	green value
aColors[index].blue	blue value

This function is a wrapper for DrawRTFHeader(). It simplifies the call, by only requiring 5 parameters, instead of the 14 parameters required by DrawRTFHeader().

It returns an RTF document or file header. This is the first function that must be called for the RTF report. Note that this is not an HTML page header, but the header of the RTF file itself.

* Note that this function will lock your RTF document by default (make it read only), if you only provide the required parameters.

DrawRTFFooter()
This function returns an RTF footer for a report. It closes an RTF document and must be the last function called. Note that this is not a page footer but the RTF file footer.

QuickDrawRTFFooter()
This is a simple wrapper function for DrawRTFFooter(). It exists only to provide symmetric coding language when QuickDrawRTFHeader is used. It is a synonym for DrawRTFFooter(), and does not add or remove any functionality.

DrawPageHeader(sText)
And
DrawPageFooter(sText)

sText....................................	text to place in the header

These functions return the code to draw either a page header or a page footer inside the document. They should be included at the beginning of the document, right after the DrawRTFHeader() call that creates the RTF header.

This is an unstructured header and footer, unlike the `DrawPageHeader_formatted()` and `DrawPageFooter_formatted()` functions which create a 3 cell formatted header/footer for you.

You can create a more customized format by using the table or graphics functions to put tables and graphics into the header or footer.

DrawPageHeader_formatted(sLeftContent, sMidContent, sRightContent)
And
DrawPageFooter_formatted(sLeftContent, sMidContent, sRightContent)

sLeftContent........................	content for left cell (or "" for no cell)
sMidContent........................	content for left cell (or "" for no cell)
sRightContent	content for left cell (or "" for no cell)

These functions are wrappers around DrawPageHeader() and DrawPageFooter(), and draw a 3 column table for you.

They return the code to draw either a page header or a page footer inside the document. They should be included at the beginning of the document, right after the DrawRTFHeader() call that creates the RTF header. The FORMATED header or footer is divided into three segments: a left cell aligned left, a middle cell aligned center, and a right cell aligned right.

If a cell is not displayed, by passing in an empty string, the other cells get the width so that the entire header or footer still takes up the entire free width of the page. The alignment does not change, so that if you want a single, centered cell, you should set the left and right content to “”and use the center cell.

ColdFusion Example:

```
RTF_CONTENT = RTF_CONTENT & DrawPageHeader_formatted("Date: #DateFormat(Now(),
        'mm/dd/yyyy')#", "Report Title", "Page " & RTF_PAGE_NUMBER & " of " &
        RTF_PAGE_TOTAL);
```

Produces:

04/10/2003	Report Title	Page 1 of 12

SetVerticalMeasure(sMeasure)

sMeasure	measure to use. Valid values are: "inches", "lines", "twips"

This function sets the vertical measurement units. This should be called before functions that use the measurements are called. The default is TWIPS.

SetHorizontalMeasure(sMeasure)

sMeasure	measure to use. Valid values are: "inches", "lines", "twips"

This function sets the horizontal measurement units. This should be called before functions that use the measurements are called. The default is TWIPS.

GetAvailableWidth()
This function returns the amount of horizontal space on the page available after the page size and margins are taken into consideration. It will return the value in the current horizontal units. This function requires `DrawRTFHeader()` to have been called in order to set up the page size and margins.

GetAvailableHeight()
This function returns the amount of vertical space on the page available after the page size and margins are taken into consideration. It will return the value in the current vertical units. This function requires `DrawRTFHeader()` to have been called in order to set up the page size and margins.

Text Formatting Functions

These RTF functions operate on text, either in text boxes or on the page. For paragraph and character text, there are a number of `Register…()` functions that are used to set up the attributes of the text once (or to change it later). Then `Draw…()` functions actually create the RTF code to display the text.

RegisterFont(CONST_FONT, fFontSize, iAttrib, CONST_FONT_COLOR)

CONST_FONT	font to use for this text (see RTF Constants)
fFontSize	font size of text in points
CONST_FONT_ATTRIB...	font attributes added together (see RTF Constants) or 0 for none
CONST_FONT_COLOR ...	color of text from color table (see RTF Constants)

This function sets up the font characteristics until they are changed. It writes no code itself and returns nothing.

Note: instead of using RegisterFont(), you can provide attributes to over-ride font attributes on the fly with DrawChars();

ResetFont()
This function resets the font to the default set up in `DrawRTFHeader()`.

RegisterParagraphIndent(cAlign, fLeftIndent, fRightIndent, fFirstIndent)

cAlign.................................	paragraph alignment ("l" – left, "c" – center, "r" – right)
fLeftIndent	left indent in current horizontal units
fRightIndent	right indent in current horizontal units
fFirstIndent..........................	first indent in current horizontal units

This function sets up the paragraph spacing for all following paragraphs until it is changed. It writes no code itself and returns nothing.

See SetHorizonalMeasure()

ResetParagraphIndent()
This function resets paragraph indenting to 0 and alignment to left.

RegisterParagraphSpacing(fSpaceBefore, fSpaceAfter, iSpacing)

fSpaceBefore.......................	amount of space before paragraph in current vertical units
fSpaceAfter	amount of space after paragraph in current vertical units
iSpacing..............................	spacing between lines in current vertical units

This function sets up the paragraph spacing for all following paragraphs until it is changed. It writes no code itself and returns nothing.

See SetVerticalMeasure()

ResetParagraphSpacing()
This function resets paragraph spacing to defaults (single spaced, no space before or after).

RegisterParagraphTabs(TabStopList, TabTypeList)

TabStopList.........................	A comma delimited list of tab stops, in current horizontal units. For example: 0.5,1.5,2.5,3.5
TabTypeList........................	A comma delimited list of tab types. The number of elements in this list MUST match the number of elements in TabStopList. Allowed tab types are: ("l" – left, "c" – centered, "r" – right, "d" – decimal) For example: l,l,c,c

This function takes a variable number of tab type and stop pairs. It sets up the tabs for all the following paragraphs until they are changed. It writes no code itself and returns nothing. In order to use the tabs in your paragraph text, use "`\tab `" (make sure it has a space after it).or the RTF_TAB constant You can have several lines within the same paragraph use the tabs if you separate each line with the `\line` code (again, it must have a space behind it).

ResetParagraphTabs()
This function resets the paragraph tabs, enabling the default 0.5" tabs.

DrawParagraph(sText)

sText....................................	text of paragraph

This function returns the code to draw the paragraph text using the current paragraph and font settings that have been previously registered. All the text in the paragraph will have the same font attributes applied.

DrawRawParagraph(sText)

sText....................................	raw text of paragraph

This function returns the code to draw a raw paragraph. This means that unlike `DrawParagraph()` this function will not apply `DrawChars()` to the text string passed in. If font formatting is required, you must apply `DrawChars()` to the text yourself. This allows you to draw a paragraph with mixed character formatting within the paragraph itself.

DrawChars(sText, [CONST_FONT_FACE , fFontSize, CONST_FONT_ATTRIB, CONST_FONT_COLOR])

sText....................................	text to apply font formatting to
[CONST_FONT_FACE]....	font to use for this text (see Fonts section above)
[fFontSize]	font size of text in points
[CONST_FONT_ATTRIB]	attributes added together (see Fonts section above) or 0 for none
[CONST_FONT_COLOR].	color of text from color table (see Color section above)

This function returns the code to apply the currently registered font formatting to the text. You may also override the current formatting by passing in explicit formatting.

QuickDrawTextBox (sText, fLeft, fRight, fTop, fBottom, cAlign)

sText....................................	Formatted text to display. To change the text formatting from the default, use DrawText() on the text string first.
fLeft....................................	position of the box's LEFT border from the LEFT PAGE MARGIN, in current horizontal units
fRight	position of the box's RIGHT border from the LEFT PAGE MARGIN, in current horizontal units
fTop....................................	position of the box's TOP border from the TOP PAGE MARGIN, in current vertical units
fBottom	position of the box's BOTTOM border from the TOP PAGE MARIGIN, in current vertical units
cAlign.................................	alignment ("l" – left, "r" – right, "c" – center)

This function is a wrapper for DrawTextBox(), but provides the user with simpler options. This function returns code to place a standard text box on the page. The box is absolutely positioned, and text will flow around it.

Note about box sizing: the width of the shape is determined by the difference between fLeft and fRight and the height of the shape is determined by the difference between fTop and fBottom. In other words, fRight should always be greater than fLeft, and fBottom should always be greater than fTop.

DrawTextBox(sFormatedText, fLeft, fRight, fTop, fBottom, cAlign, fLineWidth, sLineColor, CONST_LINE_FORMAT, sFillColor, fLeftMargin, fTopMargin, fRightMargin, fBottomMargin)

sFormatedText	Formatted text to display. To change the text formatting from the default, use DrawText() to apply font formatting.
fLeft....................................	position of the box's LEFT border from the LEFT PAGE MARGIN, in current horizontal units
fRight	position of the box's RIGHT border from the LEFT PAGE MARGIN, in current horizontal units
fTop....................................	position of the box's TOP border from the TOP PAGE MARGIN, in current vertical units

fBottom	position of the box's BOTTOM border from the TOP PAGE MARIGIN, in current vertical units
cAlign................................	alignment ("l" – left, "r" – right, "c" – center)
fLineWidth.........................	width of border line in points
sLineColor..........................	color of border line using web colors
CONST_LINE_FORMAT..	line dashing (see RTF line/border constants above)
sFillColor	color of shape using web colors
fLeftMargin........................	left internal text margin in current horizontal units, 0 is none
fRightMargin......................	right internal text margin in current horizontal units, 0 is none
fTopMargin	top internal text margin in current horizontal units, 0 is none
fBottomMargin	bottom internal text margin in current horizontal units, 0 is none

This function returns code for a more detailed text box. The box is absolutely positioned, and text will flow around it.

Note about box sizing: the width of the shape is determined by the difference between fLeft and fRight and the height of the shape is determined by the difference between fTop and fBottom. In other words, fRight should always be greater than fLeft, and fBottom should always be greater than fTop.

DrawField(sFieldName, sResult, sAttribs)

sFieldName	name of word field to draw
sResult................................	field result from last update (usually a default)
sAttribs...............................	RTF field attributes (\flddirty, etc.)

This function returns the code to draw an MS Word field variable value. If you don't know what a field is in MS Word, and what it's attributes are, then you probably should not be using this function.

Table Functions

The following functions are used to produce tables. In general a table is made up of several rows, each of which contains one or more cells. The table must be started with `DrawTableStart()` and ended with `DrawTableEnd()`. Between these each row is created by first registering all of its cells using `RegisterTableCell()`. Once all the cells for a row have been registered, then the row is drawn using `DrawTableRow()`. After each call to `DrawTableRow()`, more cells can be registered and more rows drawn. When the whole table is finished, `DrawTableEnd()` finishes the table.

RegisterTableSettings (TableAlignment, [CellContentsAlign, iCellShadePercent, RTF_CellShadeColor])

cTableAlignment................	alignment of table on page, 'l' - left, 'c' - center, 'r' - right
[cCellAlignment]	horizonal alignment of text in cell, 'l' - left, 'c' - center, 'r' - right
[cCellVerticalAlignment] ...	vertical alignment of text in cell, 't' - top, 'c' - center, 'b' - bottom
[RTF_CellShadeColor]	if cell is shaded, what RTF color?
[iCellShadePercent]	percent of shading 0 is no color, 100 is full color

This function creates default table settings, so that they do not need to be set each time in RegisterTableCell() and DrawTableRow().

RegisterTableCellPadding (fCellPadLeft, fCellPadRight, fCellPadTop,fCellPadBottom)

fCellPadLeft........................	left cell padding in current HORIZONAL units
fCellPadRight......................	right cell padding in current HORIZONAL units
fCellPadTop	top cell padding in current VERTICAL units
fCellPadBottom..................	bottom cell padding in current VERTICAL units

This function sets default table cell padding values, so that they do not need to be set each time in DrawTableRow(). Note that padding is set for all cells in a given row, and cannot be set for individual cells in a given row.

RegisterTableCellBorders (bCellBorderLeft, bCellBorderRight, bCellBorderTop, bCellBorderBottom, RTF_BorderColor)

bCellBorderLeft	is there a left cell border?
bCellBorderRight................	is there a right cell border?
bCellBorderTop	is there a top cell border?
bCellBorderBottom	is there a bottom cell border
RTF_BorderColor	what is the border color?

This function creates default table cell border settings, so that they do not need to be set each time in RegisterTableCell()

RegisterTableCell(sText, fWidth, [cAlign, cValign, RTF_SHADE_COLOR, iPercentLightDark, bBorderLeft, bBorderRight, bBorderTop, bBorderBottom, RTF_BorderColor])

sText	text to fill in cell (can include RTF or other RTF text function)
fWidth	width of cell in current HORIZONAL units
[cAlign]	horizontal alignment of contents of cell ("l" – left, "c" – center, "r" – right)
[cValign]	VERTICAL alignment of contents of cell ("t" – top, "c" – center, "b" – bottom)
[RTF_SHADE_COLOR] ...	color of shaded cell (see RTF Constants section above). **Note:** This is not a hex formatted web color, but an RTF color constant!
[iPercentLightDark]	if shade color is specified, what percent light or dark Must be a number between -100 and 100. Positive numbers lighten the color. Negative numbers darken the color.
[bBorderLeft]	is there a left border?
[bBorderRight]	is there a right border?
[bBorderTop]	is there a top border?
[bBorderBottom]	is there a bottom border
RTF_BorderColor	The color of the cell borders (see RTF Constants section)

This function registers a cell to be drawn by the `DrawTableRow()` function. Once a number of cells have been registered, call `DrawTableRow()` to draw them. Each call to `DrawTableRow()` draws a different row. The register function itself draws nothing nor does it return anything. Cells default to having borders, so you can either turn them off one by one here or turn off borders for the entire row in `DrawTableRow()`.

Default values can be set with RegisterTableSettings(), RegisterTableCellBorders() and RegisterTableCellPadding() so they do not need to be set each time this function is called.

DrawTableRow(bRepeateHeaderRow, bBorders, [cPadLeft, cPadRight, cPadTop, cPadBottom])

bRepeateHeaderRow...........	is this row one that should be repeated on each page?
bBorders..............................	should there be borders?
[cPadLeft]	left cell padding in current HORIZONAL units
[cPadRight]	right cell padding in current HORIZONAL units
[cPadTop]............................	top cell padding in current VERTICAL units
[cPadBottom]	bottom cell padding in current VERTICAL units

This function actually returns the code to draw a table row consisting of all the cells defined so far. After this row is drawn more cells may be registered and the process begun again.

Note: only one header row can be assigned in a given table.

Default padding values can be set with RegisterTableCellPadding() so they do not need to be set each time this function is called.

Graphical Functions

These functions produce images of one sort or another.

Note: In cases where the same image is used repeatedly in your output, you can save processing time by calling RegisterImageCache() or RegisterImage once, and save the Hexadecimal image data to a local variable.

RegisterImageCache(ImageFileSpec) **ASP ONLY**

ImageFileSpec.....................	Fully qualified path to the image file. This may be a UNC path.

This function returns the image data as a hexadecimal string, for use in using `DrawImage()`, or one of the other image functions. It is limited to image files of type: GIF, JPG or PNG.

ImageFileSpec should be a fully qualified path to an image. It can be something like:

- c:\inetpub\wwwroot\MyImage\picture.gif
- \\SERVER\SHARENAME\images\image.jpg

It is easy to get the fully qualified path from the URL for the image. The image URL is what you would put in the <IMG SRC=""> tag.

- ASP: filespec = Server.MapPath(fileURL)
- CF: filespec = Expandpath(fileURL)

The first time you reference an image, it saves (caches) the hexadecimal string to a file on disk named filename.HEX. The .HEX file will be created in the same directory as your source image file. If the source image file is updated, it will re-create the related .HEX file also.

Note: You may notice a delay the first time you reference a large image file, since the function is doing the work of converting and storing the HEX data to disk. You may also notice a delay when you edit or modify a large image file.

Note: Since this function writes a .HEX file to disk, it requires IIS to have write permissions to the directory where your images are stored.

<CF_RegisterImageCache ImageFileSpec Variable> **CF CUSTOM TAG ONLY**

ImageFileSpec.....................	Fully qualified path to the image file
variable................................	The hexademical image content is returned to the calling page in a variable with this name.

This custom tag returns the image data as a hexadecimal string, for use in using `DrawImage()`, or one of the other image functions.

It is limited to image files of type: GIF, JPG or PNG.

The first time you reference an image, it saves (caches) the hexadecimal string to a file on disk named filename.HEX. The .HEX file will be created in the same directory as your source image file. If the source image file is updated, it will re-create the related .HEX file also.

Note: You may notice a delay the first time you reference a large image file, since the function is doing the work of converting and storing the HEX data to disk. You may also notice a delay when you edit or modify a large image file.

If your page times out when attempting to convert/cache the image file, you may need to increase the timeout for your page. Again, this is only a problem **the 1st time** you use or reference a graphic. After the image is cached, it will run very fast.

In CF5:

```
<cfset TimeoutSeconds = 120 * 60> <!---120 minutes --->
<cfparam name="URL.RequestTimeout" default=TimeoutSeconds>
```

In CFMX:

```
<cfsetting RequestTimeout = TimeoutSeconds>
```

Note: Since this function writes a .HEX file to disk, it requires ColdFusion to have write permissions to the directory where your images are stored.

Note: in CF, you would typically call this custom tag using the CFMODULE tag:

```
<CFMODULE template="library/RegisterImageCache.cfm"
       ImageFileSpec="#filespec#"
       variable="strImageData">
```

Note: This functionality is provided as a custom tag, so it works in both CF5 and CFMX. Feel free to re-write it as a function in CFC in CFMX.

RegisterImage(sBinaryImageData) **Deprecated**

sBinaryImageData...............	binary image data

This function translates binary image data to a hexadecimal string for display using `DrawImage()`.

Note: RegisterImageCache() is the preferred function to use. However, you may need to use this function if you do not have write permissions to the directory where your images are located.

This function is slow, and will have problems with large image files. If you use the same image in multiple locations in your output, convert the binary image data once, and save it to a variable.

Idea: You could also use application variables to cache the HEX output of this function, making them quicker.

In ASP, use OpenBinaryFile() to provide this function with binary image file contents.

In ColdFusion, use <CFFILE action="READBINARY" to provide this function with binary image file contents.

OpenBinaryFile(filespec) **ASP ONLY**

filespec	fully qualified file path (drive, path, filename). Can be a UNC path.

This function is used to open an image file, and return the binary contents. ColdFusion already has a built in tag <CFFILE> to open binary files.

Note: In ColdFusion, you can use the <CFFILE action="READBINARY"> tag to open a binary file.

DrawImage(sImageData, CONST_IMAGE_TYPE, iWidth, iHeight, fScale)

sImageData	Hexadecimal image data that has been registered with `RegisterImageCache() or RegisterImage()`
CONST_IMAGE_TYPE	type of image (RTF_JPG, RTF_PNG, RTF_GIF)
iWidth	width of image in pixels
iHeight................................	height of image in pixels
fScale.................................	ratio to scale image (1.0 = no scale, 0.5 = half size, etc.)

This function returns the code to include an inline image.

DrawImageShape(sImageData, CONST_IMAGE_TYPE, CONST_WRAP, fWidth, fHeight, fLeft, fRight, fTop, fBottom)

sImageData	Hexadecimal image data that has been registered with `RegisterImageCache() or RegisterImage()`
CONST_IMAGE_TYPE	type of image (RTF_JPG, RTF_PNG, RTF_GIF)
CONST_WRAP..................	type of wrapping (see Wrapping section above)
fWidth	width of image in pixels, change to scale
fHeight	height of image in pixels, change to scale
fLeft....................................	position of the box's LEFT border from the LEFT PAGE MARGIN, in current horizontal units

fRight	position of the box's RIGHT border from the LEFT PAGE MARGIN, in current horizontal units **Note:** provide a value of -1 to calculate at 72 pixels/inch)
fTop....................................	position of the box's TOP border from the TOP PAGE MARGIN, in current vertical units
fBottom	position of the box's BOTTOM border from the TOP PAGE MARIGIN, in current vertical units. **Note:** provide a value of -1 to calculate at 72 pixels/inch)

This function returns the RTF code to place an image in a shape. It may then be sized and wrapped accordingly. Setting the right or bottom dimension to -1 will cause the function to calculate the appropriate position based on 72 pixels/inch. Setting all four dimensions (left, right, top, bottom) to -1 will center the image in the margins.

RegisterWatermark(sImageData, CONST_IMAGE_TYPE, iWidth, iHeight, fScale)

sImageData	Hexadecimal image data that has been registered with `RegisterImageCache() or RegisterImage()`
CONST_IMAGE_TYPE	type of image (RTF_JPG, RTF_PNG, RTF_GIF)
iWidth	width in pixels
iHeight................................	height in pixels
fScale..................................	scale of picture (1.0 = no scale, 0.5 = half size, etc., -1 = whole area within margins)

This function registers a watermark to be drawn when `DrawPageHeader_formatted()` or `DrawPageHeader()` is called. You must call one of the `DrawPageHeader()` functions in order to have a watermark. To change the size of the image, set the scale to keep the aspect ratio, or change the width and height manually. We scale at 72 pixels/inch. If you want the image to be the size of the page within the margins, set the scale to -1. This function should be called between the `DrawRTFHeader()` and `DrawPageHeader()` calls. This will ensure that the margins will be setup and that the watermark will make it into the page header.

DrawLine(fLeft, fRight, fTop, fBottom, fLineWidth, sLineColor, CONST_LINE_TYPE)

fLeft....................................	position of the line's STARTING POINT from the LEFT PAGE MARGIN, in current horizontal units
fRight	position of the line's ENDING POINT from the LEFT PAGE MARGIN, in current horizontal units
fTop....................................	position of the line's STARTING POINT from the TOP PAGE MARGIN, in current vertical units
fBottom	position of the line's ENDING POINT from the TOP PAGE MARIGIN, in current vertical units.
fLineWidth..........................	width of border line in points

sLineColor..........................	color of border line using a hexadecimal web color. (AABBCC) for example.
CONST_LINE_TYPE	line dashing (see Line Formatting section above)

This function draws a line.

For a **horizontal** line, give fTop and fBottom the same value. fLeft and fRight will determine the length of the line.

To draw a line at an **angle**, give fTop and fBottom different values. Give fLeft and fRight different values.

To draw a **vertical** line, give fLeft and fRight the same value. fTop and fBottom will determine the height of the line.

DrawBox(fLeft, fRight, fTop, fBottom, fLineWidth, sLineColor, CONST_LINE_TYPE, sFillColor)

fLeft....................................	position of the box's LEFT border from the page LEFT margin, in current horizontal units
fRight	position of the box's RIGHT border from the page LEFT margin, in current horizontal units
fTop....................................	position of the box's TOP border from the page TOP margin, in current horizontal units
fBottom	position of the box's BOTTOM border from the page TOP margin, in current horizontal units
fLineWidth.........................	width of border line in points
sLineColor..........................	color of border line using a hexadecimal web color. (AABBCC) for example.
CONST_LINE_TYPE	line dashing (see Line Formatting constants above)
sFillColor	color of shape fill using web colors

This function draws a box.

DrawFigure(CONST_SHAPE,fLeft, fRight, fTop, fBottom, fLineWidth, sLineColor, CONST_LINE_FORMAT, sFillColor)

CONST_SHAPE................	one of the shape constants
fLeft....................................	position of the figure's LEFT border from the page LEFT margin, in current horizontal units
fRight	position of the figure's RIGHT border from the page LEFT margin, in current horizontal units
fTop....................................	position of the figure's TOP border from the page TOP margin, in current horizontal units
fBottom	position of the figure's BOTTOM border from the page TOP margin, in current horizontal units
fLineWidth..........................	border width in points
sLineColor..........................	color of border line using a hexadecimal web color. (AABBCC) for example.
CONST_LINE_FORMAT..	one of the line format constants (See the RTF Constants section of this doc.)
sFillColor	Color of shape using web colors

To reverse, or flip a figure, swap the fLeft and fRight values, and/or swap the fTop and fBottom values.

DrawTextFigure(CONST_SHAPE,sText, fLeft, fRight, fTop, fBottom, fLineWidth, sLineColor,
CONST_LINE_FORMAT, sFillColor, cAlign, sToolTip)

CONST_SHAPE................	an RTF_SHAPE constant (see RTF Constants)
sFormatedText	Formated text to display. Alternate text formatting should be already applied to the text being passed in with DrawChars(), otherwise the default text formatting is applied.
fLeft....................................	position of the figure's LEFT border from the page LEFT margin, in current horizontal units
fRight	position of the figure's RIGHT border from the page LEFT margin, in current horizontal units
fTop....................................	position of the figure's TOP border from the page TOP margin, in current horizontal units
fBottom	position of the figure's BOTTOM border from the page TOP margin, in current horizontal units
fTextSize	font size of text in points
cAlign	'l' - left, 'r' - right, 'c' - center
fLineWidth	width of border line in points
sLineColor	Color of border line using web colors
CONST_LINE_FORMAT..	a line/border format constants
sFillColor	color of border line using a hexadecimal web color. (AABBCC) for example.

fLeftMargin	Left internal text margin in inches, 0 is none
fTopMargin	Top internal text margin in inches, 0 is none
fRightMargin	Right internal text margin in inches, 0 is none
fBottomMargin	Bottom internal text margin in inches, 0 is none

To reverse, or flip a figure, swap the fLeft and fRight values, and/or swap the fTop and fBottom values.

Alphabetical Index of Functions

Function	Type
DrawBox()	Graphical
DrawChars()	Text Formatting
DrawField()	Text Formatting
DrawFigure()	Graphical
DrawImage()	Graphical
DrawImageShape()	Graphical
DrawLine()	Graphical
DrawPageFooter()	Text Formatting
DrawPageHeader()	Text Formatting
DrawPageHeader_formatted()	Text Formatting
DrawParagraph()	Text Formatting
DrawRawParagraph()	Text Formatting
DrawRTFFooter()	Document
DrawRTFHeader()	Document
DrawTableRow()	Table
DrawTextBox()	Text Formatting
DrawTextFigure()	Graphical
GetAvailableHeight()	Document
GetAvailableWidth()	Document
OpenBinaryFile() ASP	Graphical
QuickDrawRTFFooter()	Document
QuickDrawRTFHeader()	Document
QuickDrawTextBox()	Text Formatting
RegisterFont()	Text Formatting
RegisterImage()	Graphical
RegisterImageCache() ASP	Graphical
<CF_RegisterImageCache> CF	Graphical
RegisterParagraphIndent()	Text Formatting
RegisterParagraphSpacing()	Text Formatting
RegisterParagraphTabs()	Text Formatting
RegisterTableCell()	Table
RegisterTableCellBorders()	Table

RegisterTableCellPadding()	Table
RegisterTableSettings()	Table
RegisterWatermark()	Graphical
ResetFont()	Text Formatting
ResetParagraphIndent()	Text Formatting
ResetParagraphSpacing()	Text Formatting
ResetParagraphTabs()	Text Formatting
SetHorizontalMeasure()	Document
SetVerticalMeasure()	Document

Known Issues

Various Box and Shape functions

In MS Word 2002: These functions do not work very well if used inside page headers and footers, when using DrawPageHeader() or DrawPageFooter().

DrawTableStart([cAlign])

Providing the alignment attribute may cause problems with long tables that cross pages; the table may be cut off and the remainder of the table does not show up on the next page. Leave out the alignment attribute if this is the case.

Tables inside tables (nested tables) are not supported

Due to the way in which tables are generated, you cannot put a table inside the cell of another table.