

C.R.O.S.S. PDF Generation Suite Version 1.0

Copyright (c) 2003 C.R.O.S.S. Pty Limited. All rights reserved. Unauthorised reproduction or publication in any form is prohibited. Property of C.R.O.S.S. Pty Limited.

All trademarks are the property of their respective owners.

This publication and the information herein are furnished AS IS, are subject to change without notice, and should not be construed as a commitment by C.R.O.S.S. Pty Ltd. C.R.O.S.S. Pty Ltd assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third-party rights.

C.R.O.S.S. PDF GENERATION SUITE VERSION 1.0	1
INTRODUCTION	5
CHAPTER 1 INSTALLATION	6
Extracting the Installation Program	6
Installing the Software	7
Creating VOC Entries	7
Uninstallation	7
CHAPTER 2 CONFIGURATION	8
Making the Suite Available in an Account	8
CHAPTER 3 USING THE SUITE FROM TCL	10
Generating a PDF from a Spooler Entry or Text File using CROSS.SPOOL2PDF	10
Writing a Line Art XML file	13
Preparing Line Art XML for use in a PDF	19
Preparing a JPEG for use in a PDF	19
CHAPTER 4 USING THE SUITE IN BASIC PROGRAMS	20
The PDF file format	20
Reading, Writing and Copying PDF files	20
Generating a PDF in memory	21
Generating a PDF sequentially	26
Preparing Line Art XML for use in a PDF within a Program	29
Preparing a JPEG for use in a PDF within a Program	30
CHAPTER 5 ADDING ADDITIONAL FONTS	31

Table of Contents

APPENDIX A CONFIGURATION FILES REFERENCE	34
Convention Items	35
Format Items	39
Option Items	40
Paper Items	41
APPENDIX B LINE ART XML REFERENCE	42
LINEART	43
PAGE	44
APPLY	45
BEZIER	46
BEZTO	47
CIRCLE	48
FIELD	49
FILL	50
LABEL	51
LINE	52
LINETO	53
PDF	54
POLY	55
QUADRANT	56
QUADTO	57
RECT	58
RESOURCE	59
STROKE	60

APPENDIX C PROGRAM AND SUBROUTINE REFERENCE	62
CROSS.COMPILEFONT (Program)	63
CROSS.GENART (Program)	64
CROSS.GENARTSUB (Subroutine)	65
CROSS.GENFORMAT (Program)	66
CROSS.JPEG4PDF (Program)	67
CROSS.JPEG4PDFSUB (Subroutine)	68
CROSS.PDFADD (Subroutine)	69
CROSS.PDFADDLINE (Subroutine)	70
CROSS.PDFADDPAGE (Subroutine)	71
CROSS.PDFBREAKLINE (Subroutine)	72
CROSS.PDFBREAKPAGE (Subroutine)	73
CROSS.PDFEND (Subroutine)	74
CROSS.PDFSTART (Subroutine)	75
CROSS.SPOOL2PDF (Program)	76
CROSS.TEXT2PDF (Subroutine)	77
APPENDIX D ADOBE STANDARD PDF FONTS INF LICENSE	78
INDEX	79

Introduction

The C.R.O.S.S. PDF Generation Suite is a set of programs and subroutines written in BASIC for UniVerse. They allow Adobe Portable Document Format (PDF) files to be generated from text input.

The PDF generator allows line art, JPEG graphics and text to be placed on the pages of generated PDF files and allows form fields to be filled in at runtime.

Chapter 1 Installation

CROSS recommend that you install the PDF Generation Suite into a new account. The name of the account does not matter, but we suggest CROSSPDF as a standard. We recommend that this account be in REALITY flavour, although you may choose to create an account of a different flavour.

Once you have chosen or created the installation account, you need to perform the following steps to install the software:

Extracting the Installation Program

1. Create a Type 19 file called PDFINST
2. Create a Type 19 file called PDFINST.O
3. Extract the contents of the supplied archive containing the software into the PDFINST file.
4. If you are installing on a Unix system, follow the instructions below, otherwise skip to the Windows section which follows it.

Extracting the archive on UNIX

i. Copy the archive you received by email into this file. How you will do this depends on your system configuration. The archive is a binary file, so be careful when transferring it that you use a binary mode transfer.

- You may be able to FTP the file to your server.
- If you are running Samba, you could share the PDFINST directory and copy the archive using Windows Explorer.
- Your terminal emulator may allow you to upload files.

ii. **LOGTO** the installation account (see the start of the chapter).

iii. Shell out to Unix using the **SH** command.

iv. If you requested a gzipped archive then you need to run the following command:

```
gunzip crosspdfNNNNN.tar.gz
```

Where NNNNN is your UniVerse installation serial number.

v. Now extract the contents of the tar file:

```
tar -xvf crosspdfNNNNN.tar
```

You should see a list of the contents of the installation archive on your screen.

vii. Type **exit** to return to TCL.

Extracting the archive on Windows

Use your preferred archive extraction tool (eg: WinZIP, PKZip for Windows etc) to extract the contents of the zip file and place them in the type 19 files you created in the installation account (see the start of the chapter).

Installing the Software

Once you have extracted the installation program into the PDFINST and PDFINST.O files in the installation account, you can install the software. The CROSS PDF routines are installed by running the program PDF.INSTALLER from TCL. This program must be catalogued in the account in which the PDF routines are being installed. eg:

CATALOG PDFINST PDF.INSTALLER

PDF.INSTALLER will display the end user licence agreement, and then unpack, compile and catalogue the CROSS PDF software, as well as creating additional files required by the PDF programs.

The PDF.INSTALLER program perform the following actions:

Create files:

File	Type	Contents
PDFPGMS	19	The PDF programs and subroutines
PDFPGMS.O	19	Object Files
PDFFONTs	19	Font programs
PDFFONTs.O	19	Object Files
PDFFORMATs	18	Format Data
PDFSAMPLEs	19	Sample Programs and XML files
PDFSAMPLEs.O	19	Object Files
PDFTEST	19	Test Programs to confirm installation.
PDFTEST.O	19	Object Files
PDFDOCS	19	PDF Suite Documentation

Compiles the font programs in PDFFONTs.

Catalogues the compiled objects from PDFPGMS and PDFFONTs.

Creating VOC Entries

You can automatically create VOC/MD entries for the CROSS PDF software by running

PDF.INSTALLER -S

from TCL in the account in which the software was installed

Uninstallation

The CROSS PDF software may be uninstalled by running

PDF.INSTALLER -U

from TCL in the account in which the software was installed

Chapter 2 Configuration

Making the Suite Available in an Account

In order to use any of the generation programs in an account the program needs to appear in your VOC or MD (you can create entries for all the PDF programs using the S option to the PDF . INSTALLER). Each of the programs can be run without any other subroutines except fonts being present.

The PDFFORMATS file.

CROSS.SPOOL2PDF reads formatting and convention information at runtime. By default, the program looks for this information in a file called PDFFORMATS. If you wish to use a different file name, you must specify this with each call to CROSS.SPOOL2PDF. The program CANNOT be run without this information being available. None of the other programs in the suite have this requirement.

Required Fonts

In order to set plain text, the following font programs must be visible in the catalogue space of the running account:

```
CFNT . COURIER
CFNT . COURIER . BOLD
CFNT . COURIER . OBLIQUE
CFNT . COURIER . BOLDOBLIQUE
```

These fonts are part of the set of standard PDF fonts and are generated, compiled and catalogued locally during the install process.

Other Fonts

If you are generating a PDF which uses other fonts via artwork, then these will also need to be visible. The rest of the standard 14 font programs are:

```
CFNT . HELVETICA
CFNT . HELVETICA . BOLD
CFNT . HELVETICA . BOLDOBLIQUE
CFNT . HELVETICA . OBLIQUE
CFNT . TIMES . ROMAN
CFNT . TIMES . BOLD
CFNT . TIMES . BOLDITALIC
CFNT . TIMES . ITALIC
CFNT . SYMBOL
CFNT . ZAPFDINGBATS
```

You can generate programs for other Type 1 fonts using CROSS.COMPILEFONT. The INF files for these fonts from Adobe are supplied with the installation and their programs are generated, compiled and locally catalogued during the installation process. The license terms for these INF files may be found in Appendix D. To add additional fonts see Chapter 5.

Setting up Generation Defaults

When generating a PDF users need to supply Options, Format, Paper, and Convention information. Programs run from TCL will read this from the

Chapter 4 – Using the suite in BASIC Programs

PDFFORMATS file. Programs run as subroutines require these parameters to be supplied as dynamic arrays.

When running CROSS.SPOOL2PDF it is possible to create default values for each of these in the PDFFORMATS file. The default item IDs are:

Parameter	Default Item ID
Options	ODEFAULT
Format	FDEFAULT
Paper	PDEFAULT
Convention	CDEFAULT

You can create these files by running CROSS.GENFORMAT and selection option D. You can find the formats for each of these in Appendix A.

Chapter 3 Using the suite from TCL

The C.R.O.S.S. PDF Generation Suite allows you to generate a PDF from a plain text file at TCL. Generation is done using CROSS.SPOOL2PDF. Images can be prepared for use in PDF files using CROSS.JPEG4PDF. Art work can be processed for inclusion in a PDF using CROSS.GENART.

Generating a PDF from a Spooler Entry or Text File using CROSS.SPOOL2PDF

CROSS.SPOOL2PDF is designed to place fixed pitch report text in a PDF and, optionally to place images and art work (including proportional text) behind this.

CROSS.SPOOL2PDF takes the following parameters:

```
CROSS.SPOOL2PDF FILE, ITEM, OPTIONS, FORMAT, PAPER,  
CONVENTION, FORMATFILE, IMAGEFILE, IMAGELIST,  
PAGEARTFILE, PGARTLIST
```

A PDF is generated in the following manner. First the OPTIONS, FORMAT, PAPER and CONVENTION items are read, these tell the generator what size paper to use, and how to layout the text. If the name of a file containing these items is not specified, then CROSS.SPOOL2PDF will try to read them from the PDFFORMATS file. If the names of any of these are omitted, CROSS.SPOOL2PDF will try to use ODEFAULT, FDEFAULT, PDEFAULT and CDEFAULT respectively.

Next, if art work or image files are specified these are read and added to the PDF. After this, the fixed pitch input text is placed on the pages. Finally, the PDF is written to disk.

The OPTIONS item contains meta data for the PDF information header (this will usually be available for viewing in a PDF reader) and allows a pitch grid to be placed over the PDF for debugging text placement.

The FORMAT item sets up font parameters such as characters per line and lines per page.

The PAPER item defines paper sizes (items for the standard ISO and Imperial paper sizes are included in the installation).

The CONVENTION item defines input format, page and line numbering, pyjama (half-shade) paper, header and footer lines and line breaking conventions.

Full details of the all these items can be found in Appendix A.

To run CROSS.SPOOL2PDF, you need to supply one or more arguments on the command line. If only a single argument is supplied, it must be the id for an item in the &HOLD& file in the account in which you are working.

If two or more arguments are supplied, then they are used to replace the defaults in the following list of arguments:

	Name	Default	Description
1	FILE	&HOLD&	The file which contains the PDF input.
2	ITEM		The name of the item which contains the PDF input.
3	OPTIONS	ODEFAULT	The name of the OPTIONS for the PDF.
4	FORMAT	FDEFAULT	The name of the FORMAT for the PDF.
5	PAPER	PDEFAULT	The name of the PAPER for the PDF.
6	CONVENTION	CDEFAULT	The name of the CONVENTION for the PDF.
7	FORMATFILE	PDFFORMATS	The name of the file where items 3-6 are stored.
8	IMAGEFILE	&HOLD&	The name of the file where pre-processed images are stored.
9	IMAGELIST		A space separated list of images for the PDF.
10	PGARTFILE	&HOLD&	The name of the file where pre-processed artwork is stored.
11	PGARTLIST		A space separated list of artwork items for the PDF.

Examples:

```
CROSS.SPOOL2PDF MYREPORT
```

Will cause the following to happen:

- MYREPORT will be read from &HOLD&.
- ODEFAULT, FDEFAULT, PDEFAULT and CDEFAULT will be read from PDFFORMATS.
- A PDF will be generated and written to &HOLD& with the item id MYREPORT.PDF.

```
CROSS.SPOOL2PDF MYFILE, MYREPORT,, RFMT, PA4,,,,, ARTWK
```

Will cause the following to happen:

- MYREPORT will be read from MYFILE. ODEFAULT, RFMT, PA4 and CDEFAULT will be read from PDFFORMATS.
- ARTWRK will be read from &HOLD&.
- Artwork will be placed on the pages specified in the ARTWRK item.
- A PDF will be generated and written to MYFILE with the item id MYREPORT.PDF.

CROSS.SPOOL2PDF MYFILE, MYREPORT,, RFMT, PA4,,, IMFILE,
IM1 IM2 IM3, ARTFILE, ODDART EVENART P12

Will cause the following to happen:

- MYREPORT will be read from MYFILE.
- ODEFAULT, RFMT, PA4 and CDEFAULT will be read from PDFFORMATS.
- The pre-processed JPEG images IM1, IM2, IM3 will be read from IMFILE. And will be used by the artwork.
- ODDART, EVENART and P12 will be read from the ARTFILE file.
- Artwork specified in ODDART will be placed on all odd numbered pages. Artwork specified in EVENART will be placed on all even pages except page 12 which has its artwork specified in the P12 item.
- A PDF will be generated and written to MYFILE with the item id MYREPORT.PDF.

Writing a Line Art XML file

XML Basics

XML is a text based format for storing information. An XML file is composed of sets of tags which are text strings surrounded by angle brackets and optionally containing a list of attributes. XML is similar to HTML but is more stringent about the relationship between start and end tags and is not tied to any particular set of tag names. Tag and attribute names in XML are case sensitive.

A tag which is empty or which has a corresponding end tag is called an element. Consider a tag which has the name X.

This is a start tag for an X element: <X>
 This is an end tag: </X>

So a minimal X element would look like: <X></X>

Shorthand for this is: <X/>
 Which combines the / from the end tag into the start tag.

If an element is written like this: <COLOUR>blue</COLOUR>
 We would say that the string “blue” is the contents of this particular COLOUR element.

Elements can also have attributes. These are a set of space separated name/value pairs which come after the tag name but before the / or > characters at the end of a start tag. If the X element had attributes x, y and z with values 1, hello and " it would be written as:

```
<X x="1" y='hello' z=' ' />
```

You could also write the z attribute as z=""". This uses one of the inbuilt entity references which allow characters with special meaning to be embedded in an XML file. The standard set of entity references are:

Entity	Character	Description	ASCII Character No.
&	&	Ampersand,	38
"	"	Double Quotes,	34
'	'	Apostrophe,	39
<	<	Left Angle Bracket,	60
>	>	Right Angle Bracket,	62
&#n;	Any character where n is its decimal character number.		
&#xh;	Any character where h is its hexadecimal character number.		

An XML element can contain text, other XML elements or both. It is important to note that unlike HTML, XML requires end tags for all non-empty elements. It also requires that elements nest correctly – a child element must close before its parent. Lastly, it is a requirement that an XML file have a single root element which implies that every other element in the file will be a descendant of this element.

XML examples 1, a list of cars:

```
<CARLIST>
  <CAR id="1">
    <COLOUR>red</COLOUR>
    <DESC language="EN">
      A fast car.
    </DESC>
  </CAR>
  <CAR id="9">
    <COLOUR>blue</COLOUR>
    <DESC language="SP">
      Un coche lento.
    </DESC>
  </CAR>
  <CAR id="43">
    <COLOUR>red</COLOUR>
    <DESC language="FR">
      Une voiture sportive.
    </DESC>
  </CAR>
</CARLIST>
```

XML example 2, some XHTML:

```
<HTML>
<HEAD>
<TITLE>This XML document is an example of XHTML</TITLE>
</HEAD>
<BODY
  color="red"
  >
  XML tags can cross lines.
  <BR/>
  To write an ampersand, you need to use an
  entity reference.
  <BR/>
  Either of these will work:
  <BR/>
  & or &#38;
  <BR/>
  Note that the &lt;BR/&gt; tags contain a closing
  slash.
</BODY>
</HTML>
```

XML example 3, the same XHTML with non-significant whitespace stripped:

```
<HTML><HEAD><TITLE>This XML document is an example of
XHTML</TITLE></HEAD><BODY color="red">XML tags can cross
lines, as whitespace is not meaningful with markup.<BR/>
To write an ampersand, you need to use an entity
reference.<BR/>Either of these will work:<BR/>& or
&#38;<BR/>Note that the &lt;BR/&gt; tags contain a
closing slash.</BODY></HTML>
```

Line Art

C.R.O.S.S. Line Art XML files allow you to place text, lines, shapes, JPEG images and form fields on a page. This is done by adding XML tags to the file.

When specifying widths or coordinates in line art values may be specified using the suffixes list on the table below. Values without units specified will be treated as points (1 point is 1/72 inches or 1/(25.4 × 72) mm).

The origin for all coordinates (0, 0) is the bottom left hand corner of the page.

Units	Suffix
Points	pt
Metres	m
Centimetres	cm
Millimetres	mm
Feet	ft
Inches	in

Root Element

The root element of a line art file must be a LINEART element. A line art file with no pages looks like this:

```
<LINEART>
</LINEART>
```

You can add a “units” attribute to the LINEART element to set the default units for lengths in a page art file. For example to have units default to millimetres rather than points you would use:

```
<LINEART units="mm">
</LINEART>
```

Page Elements

Under this, the file can have one or more PAGE elements (the white space in the examples will be ignored by the parser, and can be removed from production files):

```
<LINEART>
  <PAGE>
  </PAGE>
</LINEART>
```

Each page element can have the following attributes:

Attribute	Values	Description
x	Decimal Number	Clip Box Offset from origin. Defaults to 0.
y	Decimal Number	Clip Box Offset from origin. Defaults to 0.
width	Decimal Number	Width of Clip Box. Defaults to page width.
height	Decimal Number	Height of Clip Box. Defaults to page height.
cspace	greyscale, rgb, cmyk	Sets the colour space for page contents.

The clip box defined by the x, y, width and height attributes allows you design a page which overflows the page borders while only displaying that part which fits on the page or limit art to a portion of the page (for example to limit the artwork to the printable area of your printer).

Colour Space

Setting the colour space for a page determines how colour values on the page for stroke and fill will be specified. The default colour space is greyscale. Depending on the colour space, colour values are specified as follows:

Colour Space	Attributes Names	Attribute Values
greyscale	grey	Decimal between 0 and 1. White being 0 and Black being 1.
rgb	red, green, blue	Decimal between 0 and 1. Minimum Intensity being 0 and Maximum being 1.
cmyk	cyan, magenta, yellow, black	Decimal between 0 and 1. Minimum Intensity being 0 and Maximum being 1.

Page Contents

Each page can contain the following child elements:

Name	Description
APPLY	Defines which pages this PAGE applies to.
LINE	Draws a line.
LINETO	Draws a line continuing the current line/curve.
RECT	Draws a rectangle.
BEZIER	Draws a Bezier curve.
BEZTO	Draws a Bezier curve continuing the current line/curve.
QUADRANT	Draws a quadrant (quarter arc of a circle).
QUADTO	Draws a quadrant continuing the current line/curve.
CIRCLE	Draws a circle.
LABEL	Places text.
FIELD	Places a field which text may be supplied at runtime.
PDF	Inserts raw PDF graphics operators.
POLY	Encloses a set of drawing elements stroked and filled as a single element.
RESOURCE	Draws a resource such as a JPEG image.

The LINE, LINETO, RECT, BEZIER, BEZTO, QUADRANT, QUADTO and CIRCLE elements are drawing elements. As such, they can be included in POLY elements to create complex shapes that can be stroked or filled together.

When used outside of a POLY element, each of these may include a STROKE and/or a FILL element which set colour and line characteristics for the shape. When included in a POLY element, only the POLY element may include STROKE and FILL.

LABEL and FIELD elements use a FILL element to set text colour.

Full details for these elements may be found in Appendix B Line Art XML Reference

Line Art Examples

To draw a line diagonally across an A4 page (595 by 842 points). The line is black (greyscale colour space), 1 point wide.

```
<LINE x="0" y="0" x2="595" y2="842">
  <STROKE grey="1" width="1"/>
</LINE>
```

To draw a box (100 mm square) with rounded corners (radius 10 mm) and fill it blue (RGB colour space). The bottom left corner of the box is at (100mm, 100mm).

```
<POLY>
  <FILL red="0" green="0" blue="1"/>
  <LINE x="110mm" y="200mm" x2="190mm" y2="200mm"/>
  <QUADTO radius="10mm" quadrant="2"/>
  <LINETO x2="200mm" y2="110mm"/>
  <QUADTO radius="10mm" quadrant="3"/>
  <LINETO x2="110mm" y2="100mm"/>
  <QUADTO radius="10mm" quadrant="4"/>
  <LINETO x2="100mm" y2="190mm"/>
  <QUADTO radius="10mm" quadrant="1"/>
</POLY>
```

To draw a JPEG image processed with name LOGO in the bottom left corner of a page, 1 inch by 2 inches. The resname attribute is set to the IMNAME given to an image when it is processed by CROSS.JPEG4PDF. Images are passed into the PDF generator separately from page art.

```
<RESOURCE x="0" y="0" height="1 in" width="2 in"
resname="LOGO"/>
```

To have a page apply to pages 1, 3, and all even pages:

```
<PAGE>
  <APPLY to="1"/>
  <APPLY to="3"/>
  <APPLY to="even"/>
</PAGE>
```

Possible values for the “to” attribute are odd, even, default or a page number.

Preparing Line Art XML for use in a PDF

To prepare a CROSS line art XML file for use in a PDF, you need to process it with CROSS.GENART. Doing this will check the syntax of your XML, convert the XML into PDF drawing operations and generate other information required for the art work to be embedded into a PDF. CROSS.GENART takes four arguments as follows:

	Name	Default	Description
1	SFILE	&HOLD&	The file which contains the XML input.
2	SITEM	GENART.INPUT	The name of the item to process.
3	OFILE	SFILE	The file where the processed art will be placed.
4	OITEM	SITEM.CPGART	The item name which the processed art will be given. This defaults to the value of SITEM concatenated with “.CPGART”.

Once an XML file has been processed by CROSS.GENART, you can include it in the PGARTLIST argument to CROSS.SPOOL2PDF.

Preparing a JPEG for use in a PDF

To prepare a JPEG for use in a PDF, you need to process it with CROSS.JPEG4PDF. Doing this will confirm that the file is a JPEG and that it is suitable for embedding in a PDF. It will also extract meta data about the image (such as its width and height) and generate other information required to embed the file in a PDF. CROSS.JPEG4PDF takes five arguments.

	Name	Default	Description
1	IMNAME	SITEM	The name by which the image can be referenced in a Page Art file. The names Pyjamas, FFGRID and Pageart <i>n</i> are reserved and should not be used.
2	SFILE	&HOLD&	The file which contains the JPEG.
3	SITEM	GENART.INPUT	The name of the JPEG item to process.
4	OFILE	SFILE	The file where the processed JPEG will be placed.
5	OITEM	SITEM.CPJPG	The item name which the processed JPEG will be given. This defaults to the value of SITEM concatenated with “.CPJPG”.

Once a JPEG has been processed, it can be included in the IMAGELIST argument to CROSS.SPOOL2PDF and used in a page art XML file in a RESOURCE element which rename attribute is set to IMNAME.

Chapter 4 Using the suite in BASIC Programs

The PDF file format

The Portable Document Format is written and maintained by Adobe. It provides a platform independent rich document interchange format. Document readers are available for most client platforms. Readers are available from Adobe but other commercial and open source readers are available.

Reading, Writing and Copying PDF files

PDF files are a binary format. This means that although they are generally human readable in an editor, character conversion or stripping whitespace characters may corrupt the file. In particular, since it is most common to store PDF documents in a type 1 or type 19 file (for later transfer by FTP etc) you need to be careful to disable attribute mark to line feed conversion when writing a PDF to a type 1 or type 19 file. Similarly, you need to disable line feed to attribute mark conversion when reading from a type 1 or type 19 file.

To enable this conversion to occur, in BASIC use the statement:

```
ASSIGN 0 TO SYSTEM(1017)
```

To disable the conversion:

```
ASSIGN 1 TO SYSTEM(1017)
```

For more information, see UniVerse GTAR25455.

Generating a PDF in memory

The simplest way to generate a PDF in your program is to use the CROSS.TEXT2PDF subroutine. This subroutine works in a similar manner to CROSS.SPOOL2PDF except that you supply parameters as dynamic arrays rather than as ids for items in a file.

The subroutine format for CROSS.SPOOL2PDF is

```
CROSS.TEXT2PDF(ERRORS, TEXT, PDF, OPTIONS, FORMAT, PAPER,
CONV, IMAGES, PGART, FIELDS)
```

The subroutine parameters are defined as below.

ERRORS Parameter (Output)

If the ERRORS Parameter is empty, then no errors occurred while generating the PDF. Otherwise, ERRORS contains one or more attributes describing errors which occurred. ERRORS is the first parameter to CROSS.TEXT2PDF so that the subroutine may be used as a function via an appropriate DEFFUN statement.

TEXT Parameter (Input)

This parameter is the text input for the PDF. The SEPS field (1) of the CONV parameter controls how the text is interpreted. The text is treated as having pages, lines and line parts. Line sections are used to add bold and underline/overtyping functionality to plain text reports. It is implemented this way to make the conversion of line printer reports which use carriage-return and overtyping to achieve bold and underline effects.

The values for the SEPS field are in the table below:

Break Characters				
Value	Page	Line	Line Part	Usage
<empty>	@AM	@VM	@SVM	This is simplest for generation in memory as it allows you to create your input as a dynamic array.
U	FF	LF	CR	Unix Text File
W	FF	CRLF	CR	Windows Text File
N	FF	@AM	@VM	Text item read from Type1 or Type19 file with character conversion applied.

Currently, only the meaning of the first three line parts is defined. Each line part overtypes the previous line parts so that if you include text in all three parts, the text from part one will be printed, then the text from part two will be printed over this and finally the text from part three will be printed over the top. The three parts are defined as:

Part	Usage
1	Normal Text
2	Bold Text
3	Underline, Strike Out or other overtyping effects.

So, for example to print something like this:

Normal **Bold** Underline ~~Strikeout~~

You would create the following line parts:

Part 1	Normal Underline Strikeout
Part 2	Bold
Part 3	<u> </u>

Notice that you need to use white space in each section to achieve horizontal positioning.

PDF Parameter (Output)

Once the PDF generation process is finished, the PDF parameter contains the PDF as a string which you can write to file, pass to a print processor or encode and attach to an email. The PDF generator prevents any special marks (Chars 0, 128, 252-255) from appearing in this string however the PDF specification itself does not prohibit the use of these characters. When writing this file to disk, you need to be aware of the character conversions which UniVerse may perform when reading/writing to type1 or type19 files (see the section at the start of this chapter).

OPTIONS Parameter (Input)

The OPTIONS parameter is a dynamic array which allows you to control PDF document metadata (Author, Keywords etc) and to enable the debugging grid.

Field	Name	Format	Description
1	TITLE	Text	Document Title
2	AUTHOR	Text	Document Author
3	SUBJECT	Text	Document Subject
4	KEYWORDS	Text	Document Keywords
5	CREATOR	Text	Name of the program which created the text input for the PDF.
6	PITCHGRID	P	Causes the debugging pitch grid to be drawn.

The debugging grid can be enabled by placing the character “P” in the PITCHGRID field (6) of the OPTIONS parameter. Enabling the pitch grid, prints a grid over the PDF which allows you to adjust art work and text placement. The grid includes also includes column and row numbers. Enabling this option will also cause the PDF generator to print out the following text formatting information at run time (all values are in points):

Field	Meaning
TPRT	Top margin (from PAPER parameter)
BPRT	Bottom margin (from PAPER parameter)
LPRT	Left margin (from PAPER parameter)
RPRT	Right margin (from PAPER parameter)
CHAR WIDTH	Horizontal displacement between characters (column width)
CHAR HEIGHT	Maximum character height (above the baseline)
LINE LEAD	Distance between consecutive text base lines (row height)
CHAR SPACING	Factor by which character spacing is increased to fit margins

For example:

```
TPRT: 10 BPRT: 10
LPRT: 15 RPRT: 15
CHAR WIDTH: 6.1343
CHAR HEIGHT: 10.2239
LINE LEAD: 12.2687
CHAR SPACING: 118.0834 %
```

FORMAT Parameter (Input)

The FORMAT parameter controls how text is placed on the page. It controls the number of lines per page, columns per line and the paper orientation. Its format is below:

Field	Name	Format	Description
1	<unused>		
2	ORI	P or L	Portrait or Landscape paper orientation
3	LPP	1 – 14400	Lines per page (Rows)
4	CPL	1 – 14400	Characters per line (Columns)
5	LEAD	Decimal Number	Ratio of line height to character height

PAPER Parameter (Input)

The paper parameter controls the paper size and margins for a PDF. Its format is below (all values are in points):

Field	Name	Format	Description
1	PNAME	Text	Paper Display Name eg: ISO A4
2	WIDTH	3 – 14400	Paper Width
3	HEIGHT	3 – 14400	Paper Height
4	LPRT	0 – 14400	Left Margin. Must be less than WIDTH.
5	RPRT	0 – 14400	Right Margin. Must be less than WIDTH.
6	TPRT	0 – 14400	Top Margin. Must be less than HEIGHT.
7	BPRT	0 – 14400	Bottom Margin. Must be less than HEIGHT.

Fixed text cannot be printed outside these margins, however art work is only affected by paper dimensions. If you are printing PDF files, you may wish to create paper entries which margins map to the printable area of your printer.

CONV Parameter (Input)

The CONV parameter controls how the PDF generator processes a job. It specifies how text input should be interpreted, how lines and pages should be numbered and whether to print “pyjamas” behind text (emulates half shade paper).

Field	Name	Format	Description
1	SEPS	U, W or N	Determines how text input is parsed for page, line and line part separators. If this field is empty, separators default to @AM, @VM, @SVM. U sets FF, LF, CR. W sets FF, CRLF, CR. N sets FF, @AM, @VM.
2	<unused>		
3	<unused>		
4	PJ	P	If this field contains P then pyjama lines will be printed behind text.
5	PJC	Y, R or B	Determines Pyjama line colour. By default lines are 10% Grey. Y sets 50% Grey, R sets Green, B sets Blue.
6	LN	Integer	Line numbers are printed. The value of LN will be the number of the first line.
7	LNCHAR	Integer	By default line number are right justified in a field of 4 spaces. This value can be changed here.
8	LNRST	R	If this field contains an R, line numbers reset for each page.
9	HLINES	Integer	Number of lines at the top of the page to skip numbering and Pyjamas.
10	FLINES	Integer	Number of lines at the bottom of the page to skip numbering and pyjamas.
11	PNUM	Integer	Page numbers are printed. The value of PNUM will be the number of the first page.
12	PNUMPOS	[T/B] [L/R/C]	Sets the position of page numbers. The first character [T or B] determines whether page are numbered at the top or bottom. The second character [L, R or C] determines whether they are left, right or centre aligned. By default page numbers are placed at the bottom right of the page.
13	LLINES	T or S	Determines how data lines longer than page width are treated. By default, long lines are wrapped and page numbering is incremented. Setting T causes long lines to truncate. Setting S causes lone lines to wrap but maintain line numbering.
14	FPITCH	Decimal	By default, font pitch is automatically calculated from the values in the FORMAT and PAPER items. FPITCH allow an explicit font pitch to be specified.
15	TABW	Integer 0 – 32	By default, tab characters convert to eight spaces. TABW allows tab width to be set between 0 and 32 characters.

IMAGES Parameter (Input)

The IMAGES parameter is a dynamic array of images preprocessed with CROSS.JPEG4PDF. Items processed by CROSS.JPEG4PDF can be read and inserted straight into a dynamic array, as they contain no attribute marks. If you wish to change an image name (for reference in a RESOURCE element) at run time, you can change the value in value 2. For example, to change the name of image 6 to be MUGSHOT you would code:

```
IMAGES<6, 2> = "MUGSHOT"
```

PGART Parameter (Input)

The PGART parameter is a dynamic array of CROSS page art XML items preprocessed with CROSS.GENART. Items processed by CROSS.GENART will contain one attribute per XML PAGE element. If you wish to generate a PDF which contains only page art input and you are not using fields, you must explicitly specify page numbers in the APPLY element of your page art XML. The PDF generator will generate pages up to the maximum numeric value in any APPLY element.

FIELDS Parameter (Input)

If you design your line art to include text fields that are to be filled at run time (see Appendix B Line art XML Reference, FIELD element) the data for these fields is supplied in this parameter. FIELDS is a dynamic array separated into pages (@AM) and fields (@VM).

To create a PDF which has no TEXT input and consists entirely of forms with filled fields, create your XML page art with FIELD tags as required and place the values you wish to fill in the FIELDS parameter. The PDF generator will generate as many pages as are required in order to use all the fields supplied.

If you do not supply a value for a field, the value specified in the Line art XML will be used. Otherwise the field will be empty.

Generating a PDF sequentially

If you wish to generate a PDF and the input or the output will be too large to fit in memory, you can use the sequential generation routines. These routines are also useful if you are integrating PDF production into an existing report generation process and you do not wish to write the report out to file before generating a PDF in memory or you wish to replace PRINT statements which add data to a PDF.

When generating a PDF sequentially, the calling program is responsible for opening a sequential file for writing (using OPENSEQ) and passing this handle into the PDF generator routines in the PDFHANDLE parameter. It is also responsible for passing the PDF.STATE and PDF.OBJ variables between routines.

It is important to note, that the settings supplied to CROSS.PDFSTART in the CONV parameter determine how future calls to other routines will handle input. In particular, the SEPS field (1) will determine which characters are treated as embedded line or page breaks.

Similarly, data added will observe the line overflow conventions specified in the LINES field (11) of the CONV parameter. This is of particular importance if you choose to truncate long lines (T) as repeated calls to CROSS.PDFADD will discard data after line length characters have been added to a line.

Any of the routines which add data to a PDF will observe these conventions, so that including a page break in the input for CROSS.PDFADDLINE will cause the line to break across two page at the point where the page break occurs.

Input uses the same line part conventions for bold and overtyping as the input to CROSS.TEXT2PDF.

In each of the sequential generation routines, the ERRORS parameter is the first argument so that you can use the DEFFUN statement to call the routines as FUNCTIONS. Consider the definition of the CROSS.PDFSTART routine:

```
CROSS.PDFSTART(ERRORS, PDFHANDLE, OPTIONS, FORMAT, PAPER,
CONV, IMAGES, PGART, PDF.STATE, PDF.OBJ)
```

To allow your program to call the routines as functions, you might include statements like the following:

```
DEFFUN CROSS.PDFSTART(PDFHANDLE, OPTIONS, FORMAT, PAPER,
CONV, IMAGES, PGART, PDF.STATE, PDF.OBJ) CALLING
CROSS.PDFSTART
```

You can then use CROSS.PDFSTART in your program as a function which return value is the ERRORS parameter. For example:

```
PERR = CROSS.PDFSTART(A, B, C, D, E, F, H, I, J)
IF PERR NE "" THEN
    * Do some error handling
END
```

Initialising a PDF

To begin a PDF, your program should call the CROSS.PDFSTART subroutine. This routine will write PDF header information to the sequential file handle and process the OPTIONS, FORMAT, PAPER, CONV, IMAGES and PGART parameters.

You must supply all IMAGES and PGART to the CROSS.PDFSTART routine as you cannot add them to the file after this routine has been called. If you generate line art which contains FIELD elements, only the default values for the fields will be used as you cannot supply field data at run time to the sequential generator.

The format of the subroutine is:

```
CROSS.PDFSTART(ERRORS, PDFHANDLE, OPTIONS, FORMAT, PAPER,  
CONV, IMAGES, PGART, PDF.STATE, PDF.OBJ)
```

All the parameters to CROSS.PDFSTART have the same meanings as their counterparts in CROSS.TEXT2PDF. The parameters unique to the routine are defined below:

PDFHANDLE Parameter (Input/Output)

This is a file handle to a sequential file opened by the calling program using OPENSEQ. This parameter must be passed to subsequent calls to the other sequential generation routines. The file is closed by CROSS.PDFEND.

PDF.STATE Parameter (Input/Output)

This parameter contains PDF state information. It must be passed to subsequent calls to the other sequential generation routines.

PDF.OBJ Parameter (Input/Output)

This parameter contains PDF object information. It must be passed to subsequent calls to the other sequential generation routines.

Finishing a PDF

The CROSS.PDFEND routine finishes a sequentially written PDF file. It flushes any output to the file and completes the files structure. It also writes an EOF mark to the sequential file and closes the file handle.

```
CROSS.PDFEND(ERRORS, PDFHANDLE, PDF.STATE, PDF.OBJ)
```

Adding arbitrary Data

The CROSS.PDFADD subroutine adds one or more characters, lines or pages to a PDF. Its input parameters have the same meaning as those described for CROSS.PDFSTART and CROSS.TEXT2PDF.

```
CROSS.PDFADD(ERRORS, PDFHANDLE, TEXT, PDF.STATE, PDF.OBJ)
```

Adding a line of Data

The CROSS.PDFADDLINE subroutine adds text to a PDF and follows this by a line break. If the input data contains embedded line or page breaks then these will be obeyed. Its input parameters have the same meaning as those described for CROSS.PDFSTART and CROSS.TEXT2PDF.

```
CROSS.PDFADDLINE(ERRORS, PDFHANDLE, TEXT, PDF.STATE,  
PDF.OBJ)
```

Adding a page of Data

The CROSS.PDFADDLINE subroutine adds text to a PDF and follows this by a page break. If the input data contains embedded page breaks then these will be obeyed. Its input parameters have the same meaning as those described for CROSS.PDFSTART and CROSS.TEXT2PDF.

```
CROSS.PDFADDPAGE(ERRORS, PDFHANDLE, TEXT, PDF.STATE,  
PDF.OBJ)
```

Breaking a Line

You can break the current line of input by calling CROSS.PDFBREAKLINE. Its input parameters have the same meaning as those described for CROSS.PDFSTART and CROSS.TEXT2PDF.

```
CROSS.PDFBREAKLINE(ERRORS, PDFHANDLE, PDF.STATE, PDF.OBJ)
```

Breaking a Page

You can break the current page of input by calling CROSS.PDFBREAKPAGE. Its input parameters have the same meaning as those described for CROSS.PDFSTART and CROSS.TEXT2PDF.

```
CROSS.PDFBREAKPAGE(ERRORS, PDFHANDLE, PDF.STATE, PDF.OBJ)
```

Preparing Line Art XML for use in a PDF within a Program

To prepare a CROSS line art XML file for use in a PDF, you need to process it into a compiled format. If you wish to do this within a program, you can use CROSS.GENARTSUB. This subroutine performs the same job as CROSS.GENART (See Chapter 3) but does not require the programmer to use a PERFORM or EXECUTE statement. CROSS.GENARTSUB will check the syntax of your XML, convert the XML into PDF drawing operations and generate other information required for the art work to be embedded into a PDF. The subroutine is defined as:

```
CROSS .GENARTSUB ( ERRORS , PGARTXML , PGART )
```

The parameters are described below.

ERRORS (Output)

If an error has occurred during line art XML processing, ERRORS will contain a description of the error. If no errors occur, ERRORS will be empty.

PGARTXML (Input)

PGARTXML is the input CROSS Line Art XML for processing. If this input is read from a type 1 or type 19 file, be sure to disable LF to @AM conversion, as the XML parser expects the input XML to be in correct XML format.

PGART (Output)

After successful processing, PGART is a dynamic array which contains an attribute for each of the XML PAGE elements in the input XML.

Once the XML has been processed by CROSS.GENARTSUB, you can include it in the PGART argument to CROSS.TEXT2PDF or CROSS.PDFSTART.

Preparing a JPEG for use in a PDF within a Program

To prepare a JPEG for use in a PDF, you need to process it into a compiled format. To do this within a BASIC program, you should use CROSS.JPEG4PDFSUB. This routine performs the same processing as CROSS.JPEG4PDF (See Chapter 3) but does not require the program to use a PERFORM or EXECUTE statement nor to read the processed data from disk. CROSS.JPEG4PDFSUB will confirm that the file is a JPEG and that it is suitable for embedding in a PDF. It will also extract meta data about the image (such as its width and height) and generate other information required to embed the file in a PDF. The subroutine call format for CROSS.JPEG4PDFSUB is:

```
CROSS.JPEG4PDFSUB(ERRORS, JPEG, IMNAME, PROCIMAGE)
```

The parameters are described below:

ERRORS (Output)

If an error has occurred during image processing, ERRORS will contain a description of the error. If no errors occur, ERRORS will be empty.

JPEG (Input)

This parameter contains the raw JPEG image read from disk. If you are reading the image data from a type 1 or type 19 file using a READ statement, be sure to disable LF to @AM conversion, as JPEG images are binary files which will be damaged by such conversions.

IMNAME (Input)

This a text string name for the image. This name is used when referring to the processed image in a line art XML file via a RESOURCE tag. If you supply two images with the same IMNAME as input to the PDF generator, only the first image will be used. The names Pyjamas, FFGRID and Pageart*n* are reserved and should not be used.

PROCIMAGE (Output)

This parameter contains the processed image and associated metadata. Once a JPEG has been processed, it can be included in the IMAGES parameter to CROSS.TEXT2PDF or CROSS.PDFSTART and be referenced in a page art XML file in a RESOURCE element which rename attribute is set to IMNAME.

Chapter 5 Adding Additional Fonts

The C.R.O.S.S. PDF Generation Suite supports Adobe Type 1 Format fonts. The standard installation includes support for the following fonts.

Courier, Courier Bold, Courier Bold Oblique, Courier Oblique
Helvetica, Helvetica Bold, Helvetica Bold Oblique, Helvetica Oblique
Times Roman, Times Bold, Times Bold Italic, Times Italic
Symbol
ZapfDingbats

The PDF generator requires that fonts be converted into BASIC subroutines. This conversion is done using CROSS.COMPILEFONT. The program reads one or more font files (.afm, .inf or .pfb files) and generates a BASIC subroutine. This is done to minimise the time to parse and load a font that is used repeatedly. It also allows you to load a font which into global shared memory. Once a font subroutine has been generated, it needs to be compiled and made available in the VOC or MD of any account in which it will be used. CROSS.COMPILEFONT is called using this format:

```
CROSS.COMPILEFONT FILE, AFM, INF, PFB, FONTFILE,  
FONTNAME, DEFENC, DEFCUST
```

Only the first two arguments are required. The arguments have the following meanings:

FILE

The name of the type1 or type19 file containing the input files for font program generation.

AFM

The name of the item in FILE which contains Adobe Font Metric (AFM) data.

INF

If your font supplier provides an “inf” file for your font, you can use it to add additional information to the font program. INF contains the name of the item in FILE which contains this data.

PFB

If you are using a font which will not be available on your target reader system you can embed data from the Printer Font Binary (PFB) file into your PDF. This will allow you to use specific fonts and have them attached to a PDF without the requirement that these fonts be installed on your reader’s computer. If you choose to embed fonts in a PDF, you need to ensure that the license under which you purchased your fonts allows this use.

FONTFILE

By default, generated font subroutines are written into FILE. You can specify the name of an alternate type 1 or type 19 file in the FONTFILE parameter.

FONTNAME

By default, generated font programs are named based on the font name in the fonts AFM file. If you need to specify an alternate name, you can supply it in the FONTNAME parameter. This is only recommended if you are testing different versions of a font with the same name or if you require multiple custom encodings of a single font.

DEFENC

Adobe Type 1 fonts can contain many thousands of different glyphs and these glyphs need not correspond to the standard ASCII character set – for example symbol and musical fonts may contain no alphanumeric characters at all. When a font is used in a PDF file, it is used with a specific encoding which maps the byte value of text strings to glyph names. Adobe have defined several standard encodings for Latin fonts as well as custom encodings for the Symbol and ZapfDingbats fonts. You can also specify a custom encoding which allows you to determine the mapping between byte values and glyph names. The values for DEFENC are in the table below:

Value	Encoding	Description
0	Custom Encoding	Uses the fonts custom encoding or the encoding supplied in DEFCUST.
1	Adobe Standard Encoding	This is the standard built in encoding for Latin Type 1 fonts.
2	MacRoman Encoding	This is the Apple Macintosh standard encoding for Latin text.
3	WinANSI Encoding	This corresponds to code page 1252, the default text encoding on Microsoft Windows Systems.
4	PDF Doc Encoding	This is the encoding used by PDF itself for text outside of content streams (not normally used).
5	MacExpert Encoding	Some fonts are supplied as “expert” fonts containing additional characters. Not all fonts have the required additional glyphs to use this encoding.
6	Symbol Encoding	The custom encoding for the Symbol font supplied by Adobe.
7	ZapfDingbats Encoding	The custom encoding for the ZapfDingbats font supplied by Adobe.

For a complete description of these encodings and their value/glyph name tables see *PDF Reference, third edition, Adobe Portable Document Format, Version 1.4*, Appendix D, “Character Sets and Encodings”, Adobe Systems, Addison-Wesley, 2001 available from www.adobe.com.

DEFCUST

If you wish to define your own glyph subset of a font – to take advantage of special characters or when using a symbolic font – you can supply a custom encoding in an item in FILE. DEFCUST contains the name of this item. The custom encoding file should contain a list of glyph names on separate lines in a text file. The line number on which the glyph name occurs determines the byte value which the glyph will map to. The file should not contain more than 255 entries.

Most fonts will include a “space” glyph and it is recommended that this glyph be placed on the first line of the file.

Appendix A Configuration Files Reference

By default, CROSS.SPOOL2PDF expects configuration data to be stored in a file called PDFFORMATS. It is suggested that this be a hashed file.

Convention Items

When created using CROSS.GENFORMAT, convention items have an id prefix of “C” followed by the convention name. The default convention id is CDEFAULT. The values in this item will be used by CROSS.SPOOL2PDF if no other convention is specified. The attributes of a convention item are described below.

Attribute 1 : **SEPS**
 Format : Empty or a single character
 Description : The SEPS field determines how text input is parsed for page, line and line part breaks.

Values :

<default>	Pages separated by Char 254 (Attribute Marks). Lines separated by Char 253 (Value Marks) Line Parts separated by Char 252 (Sub Value Marks) This is the recommended format for in memory in put generation as input can be treated as a dynamic array.
U	Pages separated by Char 12 (Form Feed/FF). Lines separated by Char 10 (Line Feed/LF) Line Parts separated by Char 13 (Carriage Return/CR) This is the output format that would be expected in a text file formatted to print on a UNIX print queue and which uses overtyping to apply bold, underline etc.
W	Pages separated by Char 12 (Form Feed/FF). Lines separated by Chars 13,10 (Carriage Return:Line Feed/CRLF) Line Parts separated by Char 13 (Carriage Return/CR) This is the output format that would be expected in a text file formatted to print on a windows print queue and which uses overtyping to apply bold, underline etc.
N	Pages are separated by Char 12 (Form Feed/FF). Lines are separated by Char 254 (Attribute Marks) Line Parts are separated by Char 253 (Value Marks)

Attribute 2 : **Unused**

Attribute 3 : **Unused**

Attribute 4 : **PJ**
 Format : Empty or a single character
 Description : The PJ field determines whether pyjama lines are printed behind text. Pyjama line colour is set by the PJC attribute. Pyjama lines can be applied only to a portion of the page using the HLINEs and FLINES attributes. The height of the pyjama stripes is determined by the font pitch used for text output. Pyjamas are printed underneath page art and the debugging grid.

Values :

<default>	Pyjama lines are not printed.
P	Pyjama lines are printed.

Attribute 5 : **PJC**
 Format : Empty or a single character
 Description : The PJC field determines the colour of pyjama lines.
 Values :

<default>	10% Grey
Y	50% Grey
R	Green
B	Blue

Attribute 6 : **LN**
 Format : Empty or an integer
 Description : If not empty, line numbers will be printed. The first line will be numbered with the value of LN. Line numbers begin HLINEs down the page and end FLINES from the end of the page.

Values :

<default>	No line numbers
Integer n	Lines numbered starting with n

Attribute 7 : **LNCHAR**
 Format : Empty or an integer
 Description : Sets the number of columns which will be used to display line numbers.

Values :

<default>	Line numbers are right justified in a field of four spaces.
Integer n	Line numbers are right justified in a field of n spaces.

Attribute 8 : **LNRST**
 Format : Empty or a single character
 Description : Resets line numbers for each page.
 Values :

<default>	Line numbers increment across pages.
Integer n	Line numbers reset to the value of LN at the start of each page.

Appendix C – Program and Subroutine Reference

Attribute 9 : **HLINES**
 Format : Empty or an integer
 Description : Sets the number of header lines to skip before line numbering and pyjamas begin.

Values :

<default>	No header lines.
Integer n	Header lines begin at line n.

Attribute 10 : **FLINES**
 Format : Empty or an integer
 Description : Sets the number of footer lines above the bottom of the page to end line numbering and pyjamas.

Values :

<default>	No Footer lines.
Integer n	Footer lines begin at line n.

Attribute 11 : **PNUM**
 Format : Empty or an integer
 Description : If not empty, page numbers will be printed. The first page will be numbered with the value of PNUM. Page number position is determined by PNUMPOS.

Values :

<default>	No Page numbers.
Integer n	Page numbers begin at n.

Attribute 12 : **PNUMPOS**
 Format : Empty or two characters
 Description : The first character determines whether page numbers appear at the top or bottom of the page. The second character determines whether they are left, right or centre aligned.

Values :

<default>	As for BR page numbers are at the bottom right of the page.
Char 1 B	Page numbers are at the bottom of the page.
Char 1 T	Page numbers are at the top of the page.
Char 2 L	Page numbers are left aligned.
Char 2 R	Page numbers are right aligned.
Char 2 C	Page numbers are centre aligned.

For example TC would place page numbers at the top of the page, centre aligned, while BL would place them at the bottom of the page, left aligned.

Attribute 13 : **LLINES**
 Format : Empty or a single character
 Description : Determines how input lines containing more characters than the line length set by the FORMAT attribute CPL are treated.

Values :

<default>	Long lines wrap onto multiple lines. Line numbering increments for wrapped lines (A line three times CPL in length would appear on lines numbered 1, 2 and 3. The next line of data would be numbered 4).
T	Long lines truncate. (A line three times CPL would be truncated and the last two thirds discarded).
S	Long lines wrap but maintain line numbers. This is useful for numbering data such as program code (A line three times CPL would appear on a line numbered 1 followed by two lines without line numbers. The next line of data would be numbered 2).

Attribute 14 : **FPITCH**
 Format : Empty or a decimal number
 Description : Overrides automatic text pitch calculation based on PAPER and FORMAT data. Allows an explicit font pitch to be set in points.

Values :

<default>	Font pitch is adjusted to fit PAPER and FORMAT parameters.
Decimal d	Font pitch is set explicitly to d points. Text may overflow page and margins and be lost.

Attribute 15 : **TABW**
 Format : Empty or an integer between 0 and 32
 Description : Determines the width in spaces for horizontal tab (TAB/Char 9) characters.

Values :

<default>	Tabs are converted to 8 space characters.
Integer n	Tabs are converted to n space characters.

Format Items

When created using CROSS.GENFORMAT, format items have an id prefix of “F” followed by the format name. The default format id is FDEFAULT. The values in this item will be used by CROSS.SPOOL2PDF if no other format is specified. The attributes of a format item are described below.

Attribute 1 : Unused

Attribute 2 : ORI

Format : A single character.

Description : Determines whether paper is used in landscape or portrait orientation.

Values :

P	Portrait
L	Landscape

Attribute 3 : LPP

Format : An integer from 1 to 14400.

Description : Sets the number of lines of text the page should contain between the paper margins. The text font pitch will be adjusted to satisfy this setting unless it is overridden by the FPITCH attribute of the CONVENTION item.

Values :

Integer n	The generator will format n lines of text per page.
-----------	---

Attribute 4 : CPL

Format : An integer from 1 to 14400.

Description : Sets the number of columns of text the page should contain between the paper margins. The text font pitch will be adjusted to satisfy this setting unless it is overridden by the FPITCH attribute of the CONVENTION item.

The kerning of text is adjusted so that the columns completely fill the space between the margins specified for PAPER.

Values :

Integer n	The generator will format n columns of text per page.
-----------	---

Attribute 5 : LEAD

Format : Empty or a Decimal Number.

Description : Adjusts line spacing. The distance between the base lines of consecutive lines of text will be LEAD multiplied by the text font pitch (whether calculated or set explicitly).

Values :

<default>	The generator will use a LEAD value of 1.2
Decimal d	The generator will use a LEAD value of d.

Option Items

When created using CROSS.GENFORMAT, option items have an id prefix of “O” followed by the option name. The default option id is ODEFAULT. The values in this item will be used by CROSS.SPOOL2PDF if no other options are specified. The attributes of an option item are described below.

- Attribute 1** : **TITLE**
 Format : A text string. May be empty.
 Description : This string will be used as the PDF document title in the Title field of the document metadata. Your PDF reader may also display this in window titles etc.
- Attribute 2** : **AUTHOR**
 Format : A text string. May be empty.
 Description : This string will be shown as the PDF document author in the Author field of the document metadata.
- Attribute 3** : **SUBJECT**
 Format : A text string. May be empty.
 Description : This string will be shown as the PDF document subject in the Subject field of the document metadata.
- Attribute 4** : **KEYWORDS**
 Format : A text string. May be empty.
 Description : This string will be used in the Keywords field of the document metadata. Since this field is never encoded, it is an appropriate place for keywords that you wish to be indexed for text searches such as web robots.
- Attribute 5** : **CREATOR**
 Format : A text string. May be empty.
 Description : This attribute should contain the name of the program or system which produced the input for the PDF generator.
- Attribute 6** : **PITCHGRID**
 Format : Empty or a single character.
 Description : Determines whether the font pitch debugging grid is printed in the document. The grid allows you to position text in relation to other art work in the document or to design reports. The grid is printed over the top of all other document contents.

Values :

<default>	Grid is not printed
P	Grid is printed.

Paper Items

When created using CROSS.GENFORMAT, paper items have an id prefix of “P” followed by the paper name. The default paper id is PDEFAULT. The values in this item will be used by CROSS.SPOOL2PDF if no other paper is specified. The attributes of a paper item are described below.

When text is automatically sized, it is adjusted to fit inside the margins of the paper rather than the edges.

Attribute 1	:	PNAME
Format	:	Text
Description	:	The paper display name. This attribute allows you to give paper a meaningful name such as “ISO A4” or “Line Printer Special” to display to users.
Attribute 2	:	WIDTH
Format	:	Decimal between 3 and 14400
Description	:	The paper width in points.
Attribute 3	:	HEIGHT
Format	:	Decimal between 3 and 14400
Description	:	The paper height in points.
Attribute 4	:	LPRT
Format	:	Decimal between 0 and 14400
Description	:	The left margin at which text will begin. LPRT + RPRT may not be greater than WIDTH.
Attribute 5	:	RPRT
Format	:	Decimal between 0 and 14400
Description	:	The right margin at which text will begin. LPRT + RPRT may not be greater than WIDTH.
Attribute 6	:	TPRT
Format	:	Decimal between 0 and 14400
Description	:	The top margin at which text will begin. TPRT + BPRT may not be greater than HEIGHT.
Attribute 7	:	BPRT
Format	:	Decimal between 0 and 14400
Description	:	The bottom margin at which text will end. TPRT + BPRT may not be greater than HEIGHT.

Appendix B Line art XML Reference

C.R.O.S.S. XML line art files allow you to create pages on which you can draw shapes, text and form fields. These graphic elements can then be stroked and filled.

Lengths and distances may be specified with a unit suffix. Available unit suffixes are:

Suffix	Meaning
pt	Points (Default)
m	Metres
cm	Centimetres
mm	Millimetres
ft	Feet
in	Inches

Colours for stroke and fill are specified with distinct colour spaces containing 1, 3 or 4 components. The colour space and component names are are:

Colour Space	Component attributes			
greyscale	grey			
rgb	red	green	blue	
cmyk	cyan	magenta	yellow	black

All colour components can have values from 0 to 1.

LINEART

The LINEART element is the root element of line art XML file. It is a required element and only one LINEART element may be present in a file.

Attributes

Name	Format	Description	
units	Text String	This attribute allows you to reset the default measurement units for a line art file. Available values are:	
		pt	Points (Default)
		m	Metres
		cm	Centimetres
		mm	Millimetres
		ft	Feet
		in	Inches

Parents

Elements	None
-----------------	------

Children

Elements	PAGE	1 or more required
Text	None	

PAGE

PAGE elements contain the description of a single page of line art. They can be set to apply to particular pages or ranges of pages using APPLY elements.

Attributes

Name	Format	Description	
x	Decimal Units	The horizontal displacement from the left page edge to the lower left corner of the clipping box for this page of artwork. Graphics outside the clipping box will not display. The default value for this attribute is 0.	
y	Decimal Units	The vertical displacement from the bottom page edge to the lower left corner of the clipping box for this page of artwork. Graphics outside the clipping box will not display. The default value for this attribute is 0.	
width	Decimal Units	The width of the clipping box for this page of artwork. Graphics outside the clipping box will not display. The default value for this attribute is the page width used at runtime.	
height	Decimal Units	The height of the clipping box for this page of artwork. Graphics outside the clipping box will not display. The default value for this attribute is the page width used at runtime.	
cspace	Text	Sets the default colour space for the page. Values are:	
		grey	Greyscale
		rgb	RGB
		Cmyk	CMYK

Parents

Elements	LINEART
-----------------	---------

Children

Elements	APPLY	0 or more
	BEZIER	0 or more
	BEZTO	0 or more
	CIRCLE	0 or more
	FIELD	0 or more
	LABEL	0 or more
	LINE	0 or more
	LINETO	0 or more
	PDF	0 or more
	POLY	0 or more
	QUADRANT	0 or more
	QUADTO	0 or more
	RECT	0 or more
	RESOURCE	0 or more
Text	None	

APPLY

APPLY elements are used to set the page or range of pages to which a PAGE element will be applied. A PAGE may contain multiple APPLY elements.

Attributes

Name	Format	Description	
to	Variable	Specifies a page number or range of pages to which this page of artwork should be applied. Values are below:	
		even	Parent page applies to all even numbered pages.
		odd	Parent page applies to all odd numbered pages.
		default	Parent page applies as default where no other artwork is specified.
		<integer>	Parent page applies to page number specified.

Parents

Elements	PAGE
-----------------	------

Children

Elements	None
Text	None

BEZIER

BEZIER elements draw a cubic Bezier curve on a page. Cubic Bezier curves can be used to closely approximate most other curves whilst being efficient to render.

Attributes

Name	Format	Description
x	Decimal Units	X coordinate for the start of the curve
y	Decimal Units	Y coordinate for the start of the curve
x1	Decimal Units	X coordinate for the first curve control point
y1	Decimal Units	Y coordinate for the first curve control point
x2	Decimal Units	X coordinate for the second curve control point
y2	Decimal Units	Y coordinate for the second curve control point
x3	Decimal Units	X coordinate for the end of the curve
y3	Decimal Units	Y coordinate for the end of the curve

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
	FILL	0 or 1
Text	None	

BEZTO

BEZTO elements draw a cubic Bezier curve on a page. They work as BEZIER elements except that they start from the end of the last drawing operation. You can use BEZTO elements to construct a path containing a Bezier curve as a subpath.

You cannot use a BEZTO element without a preceding path drawing element.

Attributes

Name	Format	Description
x1	Decimal Units	X coordinate for the first curve control point
y1	Decimal Units	Y coordinate for the first curve control point
x2	Decimal Units	X coordinate for the second curve control point
y2	Decimal Units	Y coordinate for the second curve control point
x3	Decimal Units	X coordinate for the end of the curve
y3	Decimal Units	Y coordinate for the end of the curve

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
	FILL	0 or 1
Text	None	

CIRCLE

CIRCLE elements draw a circle on a page. You cannot include a CIRCLE as a subpath of another path.

Attributes

Name	Format	Description
x	Decimal Units	X coordinate for the centre of the circle.
y	Decimal Units	Y coordinate for the centre of the circle.
radius	Decimal Units	Length of the radius of the circle.

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
	FILL	0 or 1
Text	None	

FIELD

FIELD elements allow you to provide text to be filled into a PDF at run time.

Attributes

Name	Format	Description	
x	Decimal Units	X coordinate for the field anchor	
y	Decimal Units	Y coordinate for the field anchor	
width	Decimal Units	Width of the field.	
height	Decimal Units	Height of the field.	
font	Text	Name of the font program to use for the field eg: CFNT.HELVETICA.	
pitch	Decimal Units	The font pitch.	
align	Text	The horizontal alignment of the field contents: left, right or centre.	
valign	Text	The vertical alignment of the field contents: top, middle or bottom.	
alignbaseline	Text	Determines whether vertical alignment applies to the absolute text height or to the text base line. Values are:	
		true	Text aligns on baselines.
		false	Text aligns based on text height.
xpos	Text	The horizontal position of the field in relation to the field anchor. Values are:	
		left	The left hand side of the field is placed on the anchor point.
		centre	The centre of the field is placed on the anchor point.
		right	The right hand side of the field is placed on the anchor point.
ypos	Text	The vertical position of the field in relation to the field anchor. Values are:	
		top	The top of the field is placed on the anchor point.
		middle	The centre of the field is placed on the anchor point.
		bottom	The bottom of the field is placed on the anchor point.

Parents

Elements	PAGE
-----------------	------

Children

Elements	FILL	0 or 1. Determines text colour.
Text	The text contents of the element will be the default value for the field.	

FILL

The FILL element determines the fill colour for a closed path. Applying a fill to a non closed path will have indeterminate results. Its attributes are determined by the current colour space. Attributes specified outside the current colour space will be ignored.

Attributes

Name	Format	Description
Greyscale		
The greyscale colour space constructs colour using levels of grey. It is appropriate for display or print use.		
grey	Decimal 0 to 1	The grey level for the fill. Black being 0 and White being 1.
RGB		
The rgb colour space constructs colours using additive intensities of light. It will give best results for documents to be viewed on computer screens.		
red	Decimal 0 to 1	The value for the red component of the fill. No red being 0 and maximum intensity red being 1.
green	Decimal 0 to 1	The value for the green component of the fill. No green being 0 and maximum intensity green being 1.
blue	Decimal 0 to 1	The value for the blue component of the fill. No blue being 0 and maximum intensity blue being 1.
CMYK		
The cmyk space constructs colours by combining concentrations of ink components. It will give best results for documents which are to be printed.		
cyan	Decimal 0 to 1	The value for the cyan component of the fill. No cyan being 0 and maximum concentration cyan being 1.
magenta	Decimal 0 to 1	The value for the magenta component of the fill. No magenta being 0 and maximum concentration magenta being 1.
yellow	Decimal 0 to 1	The value for the yellow component of the fill. No yellow being 0 and maximum concentration yellow being 1.
black	Decimal 0 to 1	The value for the black component of the fill. No black being 0 and maximum concentration black being 1.

Parents

Elements	CIRCLE, POLY, RECT, LABEL, FIELD
-----------------	----------------------------------

Children

Elements	None
Text	None

LABEL

LABEL elements allow you place text on a page to be filled in at design time.

Attributes

Name	Format	Description	
x	Decimal Units	X coordinate for the field anchor	
y	Decimal Units	Y coordinate for the field anchor	
width	Decimal Units	Width of the field.	
height	Decimal Units	Height of the field.	
font	Text	Name of the font program to use for the field eg: CFNT.HELVETICA.	
pitch	Decimal Units	The font pitch.	
align	Text	The horizontal alignment of the field contents: left, right or centre.	
valign	Text	The vertical alignment of the field contents: top, middle or bottom.	
alignbaseline	Text	Determines whether vertical alignment applies to the absolute text height or to the text base line. Values are:	
		true	Text aligns on baselines.
		false	Text aligns based on text height.
xpos	Text	The horizontal position of the field in relation to the field anchor. Values are:	
		left	The left hand side of the field is placed on the anchor point.
		centre	The centre of the field is placed on the anchor point.
		right	The right hand side of the field is placed on the anchor point.
ypos	Text	The vertical position of the field in relation to the field anchor. Values are:	
		top	The top of the field is placed on the anchor point.
		middle	The centre of the field is placed on the anchor point.
		bottom	The bottom of the field is placed on the anchor point.

Parents

Elements	PAGE
-----------------	------

Children

Elements	FILL	0 or 1. Determines text colour.
Text	The text contents of the element will be the text of the label.	

LINE

LINE elements draw a straight line on a page.

Attributes

Name	Format	Description
x	Decimal Units	X coordinate for the start of the line
y	Decimal Units	Y coordinate for the start of the line
x2	Decimal Units	X coordinate for the end of the line
y2	Decimal Units	Y coordinate for the end of the line

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
Text	None	

LINETO

LINETO elements draw a straight line on a page starting from the end of the last path drawn. You can use LINETO elements to construct a path containing a straight line as a subpath.

You cannot use a LINETO element without a preceding path drawing element.

Attributes

Name	Format	Description
x2	Decimal Units	X coordinate for the end of the line
y2	Decimal Units	Y coordinate for the end of the line

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
Text	None	

PDF

PDF elements allow you to embed PDF graphics operators which are similar to PostScript directly into a page.

You should not do this unless you have a good understanding of PDF or Postscript.

When using PDF elements, it is recommended that they be placed as the very last elements of a PAGE.

Parents

Elements	PAGE
-----------------	------

Children

Elements	None
Text	None

POLY

The POLY element allows you to combine multiple paths into a single path. All the child elements of a POLY element are combined into a single path and then stroked and filled in a single operation. Since the subpaths in a POLY are stroked/filled as one path, any STROKE or FILL elements in the POLY child elements are ignored.

You may not nest POLY elements.

Attributes

Name	Format	Description
None		

Parents

Elements	PAGE

Children

Elements	STROKE	0 or 1
	FILL	0 or 1
	BEZIER	0 or more
	BEZTO	0 or more
	CIRCLE	0 or more
	LINE	0 or more
	LINETO	0 or more
	QUADRANT	0 or more
	QUADTO	0 or more
	RECT	0 or more
Text	None	

QUADRANT

QUADRANT elements draw a quarter of a circle on the page. The quadrant attribute of a QUADRANT element determines which quadrant will be drawn. The sign of the number in the quadrant attribute determines whether the quadrant will be drawn clockwise or counter clockwise.

Attributes

Name	Format	Description
x	Decimal Units	X coordinate for the start point of the quadrant (not the centre of the circle).
y	Decimal Units	Y coordinate for the start point of the quadrant (not the centre of the circle).
radius	Decimal Units	Radius of the quadrant.
quadrant	+/- 1 to 4	Sets the quadrant and drawing direction. Values are:
		1 Top Left Hand Quadrant Clockwise
		2 Top Right Hand Quadrant Clockwise
		3 Bottom Right Hand Quadrant Clockwise
		4 Bottom Left Hand Quadrant Clockwise
		-1 Top Left Hand Quadrant Counter Clockwise
		-2 Top Right Hand Quadrant Counter Clockwise
		-3 Bottom Right Hand Quadrant Counter Clockwise
-4 Bottom Left Hand Quadrant Counter Clockwise		

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
Text	None	

QUADTO

QUADTO elements draw a quarter of a circle on the page. The quadrant starts at the end of the last path drawn. The quadrant attribute of a QUADTO element determines which quadrant will be drawn. The sign of the number in the quadrant attribute determines whether the quadrant will be drawn clockwise or counter clockwise.

Attributes

Name	Format	Description
radius	Decimal Units	Radius of the quadrant.
quadrant	+/- 1 to 4	Sets the quadrant and drawing direction. Values are:
		1 Top Left Hand Quadrant Clockwise
		2 Top Right Hand Quadrant Clockwise
		3 Bottom Right Hand Quadrant Clockwise
		4 Bottom Left Hand Quadrant Clockwise
		-1 Top Left Hand Quadrant Counter Clockwise
		-2 Top Right Hand Quadrant Counter Clockwise
		-3 Bottom Right Hand Quadrant Counter Clockwise
		-4 Bottom Left Hand Quadrant Counter Clockwise

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
Text	None	

RECT

The RECT element draws a rectangle.

Attributes

Name	Format	Description
x	Decimal Units	X coordinate for the lower left corner of the rectangle.
y	Decimal Units	Y coordinate for the lower left corner of the rectangle.
width	Decimal Units	Width of the rectangle.
height	Decimal Units	Height of the rectangle.

Parents

Elements	PAGE, POLY
-----------------	------------

Children

Elements	STROKE	0 or 1
	FILL	0 or 1
Text	None	

RESOURCE

The RESOURCE element draws an element from elsewhere in the PDF . Currently, the RESOURCE element allows you to draw preprocessed JPEG images which are passed in to the PDF generator separately to the processed page art. Resources are always rectangular (or treated as having a rectangular bounding box).

Attributes

Name	Format	Description
x	Decimal Units	X coordinate for the lower left corner of the resource.
y	Decimal Units	Y coordinate for the lower left corner of the resource.
width	Decimal Units	Width of the resource.
height	Decimal Units	Height of the resource.
resname	Text	The name of the resource you wish to draw. This corresponds to the IMNAME parameter to CROSS.JPEG4PDF and CROSS.JPEG4PDFSUB. The names Pyjamas, FFGRID and Pageart <i>n</i> are reserved and should not be used.
rotate	Degrees	The resource will be rotated by this value counter clockwise.
skewx	Degrees	The resource will be skewed to the right by this value.
skewy	Degrees	The resource will be skewed upwards by this value.

Parents

Elements	PAGE
-----------------	------

Children

Elements	None
Text	None

STROKE

The STROKE element determines how a path will be drawn. You can set colour, width and stroke pattern. Its attributes are determined by the current colour space. Attributes specified outside the current colour space will be ignored.

Attributes

Name	Format	Description	
width	Decimal Units		
cap	Integer	Determines how line ends are capped. Values are:	
		0	Butt Cap. Stroke is square and does not project over the end of the path. (Default).
		1	Round Cap. The stroke ends are rounded.
		2	Projecting square cap. The stroke is square and projects half of stroke width beyond the end of the stroke.
join	Integer	Determines how subpath joins are drawn.	
		0	Miter Join. Outer edges of the strokes extend until they intersect. If the angle is greater than the miter limit, a bevel join is used.
		1	Round Join. A circle of diameter equal to line width is drawn over the line join.
		2	The lines are ended with but caps and the triangle between their ends filled.
miter	Decimal	Sets the miter limit for joins. See the page 154 of the Adobe PDF Reference for more information.	
dash	Text	Sets the line dash pattern. The attribute value is of the form:	
		<i>[pattern] phase</i>	
		Where pattern may be empty or a series of space separated number which determine dash size. The numbers represent alternating numbers of points to be drawn and not drawn. The phase determines how many points into pattern stroke drawing should begin. Examples are:	
		The default dash pattern is which is a solid line.	
		[] 0	Solid line (default)
		[3] 0	3 on, 3 off, 3 on ...
[2] 1	1 on, 2 off, 2 on, 2 off, 2 on ...		
[2 3] 11	1 on, 3 off, 2 on, 3 off, 2 on ...		

Name	Format	Description
Greyscale		
The greyscale colour space constructs colour using levels of grey. It is appropriate for display or print use.		
grey	Decimal 0 to 1	The grey level for the stroke. Black being 0 and White being 1.
RGB		
The rgb colour space constructs colours using additive intensities of light. It will give best results for documents to be viewed on computer screens.		
red	Decimal 0 to 1	The value for the red component of the stroke. No red being 0 and maximum intensity red being 1.
green	Decimal 0 to 1	The value for the green component of the stroke. No green being 0 and maximum intensity green being 1.
blue	Decimal 0 to 1	The value for the blue component of the stroke. No blue being 0 and maximum intensity blue being 1.
CMYK		
The cmyk space constructs colours by combining concentrations of ink components. It will give best results for documents which are to be printed.		
cyan	Decimal 0 to 1	The value for the cyan component of the stroke. No cyan being 0 and maximum concentration cyan being 1.
magenta	Decimal 0 to 1	The value for the magenta component of the stroke. No magenta being 0 and maximum concentration magenta being 1.
yellow	Decimal 0 to 1	The value for the yellow component of the stroke. No yellow being 0 and maximum concentration yellow being 1.
black	Decimal 0 to 1	The value for the black component of the stroke. No black being 0 and maximum concentration black being 1.

Parents

Elements	BEZIER, BEZTO, CIRCLE, LINE, LINETO, QUADRANT, QUADTO, RECT
-----------------	---

Children

Elements	None
Text	None

Appendix C Program and Subroutine Reference

This appendix contains brief descriptions and usage formats for the programs and subroutines in the C.R.O.S.S. PDF Generation Suite. The documented programs and subroutines are:

CROSS.COMPILEFONT	Program
CROSS.GENART	Program
CROSS.GENARTSUB	Subroutine
CROSS.GENFORMAT	Program
CROSS.JPEG4PDF	Program
CROSS.JPEG4PDFSUB	Subroutine
CROSS.PDFADD	Subroutine
CROSS.PDFADDLINE	Subroutine
CROSS.PDFADDPAGE	Subroutine
CROSS.PDFBREAKLINE	Subroutine
CROSS.PDFBREAKPAGE	Subroutine
CROSS.PDFEND	Subroutine
CROSS.PDFSTART	Subroutine
CROSS.SPOOL2PDF	Program
CROSS.TEXT2PDF	Subroutine

CROSS.COMPILEFONT (Program)**Purpose:**

Generates a C.R.O.S.S. font program from a type 1 font. This generated program must be compiled and catalogued before it can be run.

Format:

CROSS.COMPILEFONT FILE, AFM, INF, PFB, FONTFILE,
FONTNAME, DEFENC, DEFCUST

Arguments:

FILE	Input file name for font data	Required
AFM	AFM file name	Required
INF	INF file name	
PFB	PFB file name (if fonts are to be embedded).	
FONTFILE	Output file name for font program source defaults to value of FILE	
FONTNAME	Font program name suffix. Defaults to font distinct name (Program name will be CFNT.FONTNAME).	
DEFENC	Default Encoding	
DEFCUST	Custom Default Encoding File Name	

CROSS.GENART (Program)

Purpose:

Processes a file in C.R.O.S.S. XML line art format for use by the PDF generator.

Format:

CROSS.GENART SFILE, SITEM, OFILE, OITEM

Arguments:

SFILE	File containing XML item (defaults to &HOLD&)
SITEM	XML Item name (defaults to GENART.INPUT)
OFIELD	Output file for processed line art (defaults to value of SFILE)
OITEM	Output Item name for processed line art (defaults to value of SITEM with “.CPGART” appended).

CROSS.GENARTSUB (Subroutine)

Purpose:

Processes XML in XML line art format for use by the PDF generator.

Format:

CROSS.GENARTSUB(ERRORS, PGARTXML, PGART)

Arguments:

ERRORS	Dynamic array of errors which occurred during processing.
PGARTXML	A string containing the input XML to be processed.
PGART	The processed line art for use by the PDF generator.

CROSS.GENFORMAT (Program)

Purpose:

Allows users to interactively create format items for use by CROSS.SPOOL2PDF.

Format:

CROSS . GENFORMAT

Arguments:

None

CROSS.JPEG4PDF (Program)

Purpose:

Processes a JPEG file for use by the PDF generator.

Format:

CROSS.JPEG4PDF IMNAME, SFILE, SITEM, OFILE, OITEM

Arguments:

IMNAME	The name by which the image will be referenced in a RESOURCE element in line art XML. (defaults to the value of SITEM).
SFILE	The source file containing the JPEG to be converted (defaults to &HOLD&)
SITEM	The name of the item containing the JPEG data to be processed (defaults to JPEG4PDF.INPUT).
OFILE	The destination file for processed output (defaults to SFILE).
OITEM	The destination item name for processed output (defaults to the value of SITEM with “.CPJPG” appended).

CROSS.JPEG4PDFSUB (Subroutine)

Purpose:

Processes JPEG data for use by the PDF generator.

Format:

CROSS.JPEG4PDFSUB(ERRORS, JPEG, IMNAME, PROCIMAGE)

Arguments:

ERRORS	Dynamic array of errors which occurred during processing.
JPEG	A string containing the input JPEG data to be processed.
IMNAME	A string containing the image name by which the JPEG will be referenced in line art XML.
PROCIMAGE	The processed image data.

CROSS.PDFADD (Subroutine)**Purpose:**

Adds data to a sequentially generated PDF. Embedded page, line and line part breaks will be observed. If data is added without line or page breaks, it will be flowed appropriately.

Format:

`CROSS.PDFADD(ERRORS, PDFHANDLE, TEXT, PDF.STATE, PDF.OBJ)`

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF.
TEXT	Text to add to the PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.PDFADDLINE (Subroutine)**Purpose:**

Adds data to a sequentially generated PDF. Embedded page, line and line part breaks will be observed. A line break will always occur at the end of the input data.

Format:

CROSS.PDFADDLINE (ERRORS , PDFHANDLE , TEXT , PDF.STATE ,
PDF.OBJ)

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF.
TEXT	Text to add to the PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.PDFADDPAGE (Subroutine)

Purpose:

Add data to a sequentially generated PDF. Embedded page, line and line part breaks will be observed. A page break will always occur at the end of the input data.

Format:

CROSS.PDFADDPAGE(ERRORS , PDFHANDLE , TEXT , PDF.STATE , PDF.OBJ)

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF.
TEXT	Text to add to the PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.PDFBREAKLINE (Subroutine)

Purpose:

Starts a new line in a sequentially generated PDF.

Format:

CROSS.PDFBREAKLINE(ERRORS, PDFHANDLE, PDF.STATE, PDF.OBJ)

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.PDFBREAKPAGE (Subroutine)

Purpose:

Starts a new page in a sequentially generated PDF.

Format:

`CROSS.PDFBREAKPAGE(ERRORS, PDFHANDLE, PDF.STATE, PDF.OBJ)`

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.PDFEND (Subroutine)**Purpose:**

Finishes a sequentially written PDF. Any pending output (unfinished lines or pages) will be flushed to the file. An EOF marker will be written to the file. The sequential file handle PDFHANDLE will be closed.

Format:

CROSS.PDFEND(ERRORS, PDFHANDLE, PDF.STATE, PDF.OBJ)

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.PDFSTART (Subroutine)**Purpose:**

Begins a sequentially written PDF.

Format:

`CROSS.PDFSTART(ERRORS, PDFHANDLE, OPTIONS, FORMAT, PAPER, CONV, IMAGES, PGART, PDF.STATE, PDF.OBJ)`

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
PDFHANDLE	The sequential file handle to the generated PDF. This should be opened for writing by the calling program.
OPTIONS	A dynamic array containing an OPTIONS item for the PDF.
FORMAT	A dynamic array containing a FORMAT item for the PDF.
PAPER	A dynamic array containing a PAPER item for the PDF.
CONV	A dynamic array containing a CONV item for the PDF.
IMAGES	A dynamic array of preprocessed images to be used in page art.
PGART	A dynamic array of preprocessed page art to be used in the PDF.
PDF.STATE	PDF Generator State
PDF.OBJ	PDF Generator Objects

CROSS.SPOOL2PDF (Program)

Purpose:

Generates a PDF file from an input text file.

Format:

CROSS.SPOOL2PDF FILE, ITEM, FORMAT, PAPER, CONVENTION, FORMATFILE, IMAGEFILE, IMAGELIST, PAGEARTFILE, PGARTLIST

Arguments:

FILE	Input file name (defaults to &HOLD&)	
ITEM	Input item name.	Required
OPTIONS	Options item name (defaults to ODEFAULT)	
FORMAT	Format item name (defaults to FDEFAULT)	
PAPER	Paper item name (defaults to PDEFAULT)	
CONVENTION	Convention item name (defaults to CDEFAULT)	
FORMATFILE	File name containing OPTIONS, FORMAT, PAPER and CONVENTION items (defaults to PDFFORMATS)	
IMAGEFILE	Input file for images (defaults to &HOLD&)	
IMAGELIST	Item names of processed images to be used in PDF	
PAGEARTFILE	Input file for page art (defaults to &HOLD&)	
PGARTLIST	Item names of processed page art to be used in PDF	

CROSS.TEXT2PDF (Subroutine)**Purpose:**

Processes JPEG data for use by the PDF generator.

Format:

`CROSS.TEXT2PDF(ERRORS, TEXT, PDF, OPTIONS, FORMAT, PAPER, CONV, IMAGES, PGART, FIELDS)`

Arguments:

ERRORS	Dynamic array of errors which occurred during generation.
TEXT	The input text for the PDF.
PDF	The generated PDF.
OPTIONS	A dynamic array containing an OPTIONS item for the PDF.
FORMAT	A dynamic array containing a FORMAT item for the PDF.
PAPER	A dynamic array containing a PAPER item for the PDF.
CONV	A dynamic array containing a CONV item for the PDF.
IMAGES	A dynamic array of preprocessed images to be used in page art.
PGART	A dynamic array of preprocessed page art to be used in the PDF.
FIELDS	A dynamic array of field data to be filled into page art FIELD elements at runtime.

Appendix D Adobe Standard PDF Fonts INF license

This file and the 14 PostScript(R) AFM files it accompanies may be used, copied, and distributed for any purpose and without charge, with or without modification, provided that all copyright notices are retained; that the AFM files are not distributed without this file; that all modifications to this file or any of the AFM files are prominently noted in the modified file(s); and that this paragraph is not modified. Adobe Systems has no responsibility or obligation to support the use of the AFM files.

Index

A		L	
AFM Font Files	31	LABEL Element	51
APPLY Element	45	LINE Element	52
ASSIGN Statement	20	Line Art	15
B		Examples	18
BEZIER Element	46	Preparing XML, TCL	19
BEZTO Element	47	Preparing XML, BASIC	29
C		Reference	42
CIRCLE Element	48	LINEART Element	43
Colour Space	16	LINETO Element	53
CONVENTION Items	35	M	
CROSS.COMPILEFONT	4, 8, 31, 62, 63	MD Entries, Creating	<i>See</i> VOC Entries, Creating
CROSS.GENART	4, 10, 19, 25, 29, 62, 64	O	
CROSS.GENARTSUB	4, 29, 62, 65	OPTION Items	40
CROSS.GENFORMAT	4, 9, 35, 39, 40, 41, 62, 66	P	
CROSS.JPEG4PDF	4, 10, 18, 19, 25, 30, 59, 62, 67	PAGE Element	44
CROSS.JPEG4PDFSUB	4, 30, 59, 62, 68	PAPER Items	41
CROSS.PDFADD	4, 26, 28, 62, 69	PDF	20
CROSS.PDFADDDLINE	4, 26, 28, 62, 70	Binary	
CROSS.PDFADDPAGE	4, 28, 62, 71	Reading, Writing, Copying	20
CROSS.PDFBREAKLINE	4, 28, 62, 72	Element	54
CROSS.PDFBREAKPAGE	4, 28, 62, 73	Generating at TCL	10
CROSS.PDFEND	4, 27, 62, 74	Generating in Memory	21
CROSS.PDFSTART	4, 26, 27, 28, 29, 30, 62, 75	Generating Sequentially	26
CROSS.SPOOL2PDF	2, 4, 8, 9, 10, 11, 12, 19, 21, 34, 35, 39, 40, 41, 62, 66, 76	PDF.INSTALLER	7, 8
CROSS.TEXT2PDF	4, 21, 26, 27, 28, 29, 30, 62, 77	PDFDOCS File	7
D		PDFDOTS File	7
Defaults, TCL PDF Generation	9	PDFDOTS.O	<i>See</i> PDFDOTS
F		PDFFORMATS File	7, 8, 9, 10, 11, 12, 34, 76
FIELD Element	49	PDFINST File	6, 7
FILL Element	50	PDFINST.O	<i>See</i> PDFINST
Fonts		PDFPGMS File	7
Adding	31	PDFPGMS.O	<i>See</i> PDFPGMS
Encodings	32	PDFSAMPLES File	7
Required	8	PDFTEST File	7
Standard	8	PDFTEST.O	<i>See</i> PDFTEST
FORMAT Items	39	PFB Font Files	31
I		POLY Element	55
INF Font Files	31	Q	
J		QUADRANT Element	56
JPEGS		QUADTO Element	57
Preparing	19	R	
Preparing for use	30	RECT Element	58
K		RESOURCE Element	59
L		S	
M		STROKE Element	60

C.R.O.S.S PDF Generation Suite Version 1.0

	T		V	
TCL		10	VOC Entries, Creating	7
	U		X	
Uninstallation units, Line Art		7 15	XML	13