



RenderX XEP User Guide

XEP User Guide

© Copyright 2005-2008 RenderX, Inc. All rights reserved.

This documentation contains proprietary information belonging to RenderX, and is provided under a license agreement containing restrictions on use and disclosure. It is also protected by international copyright law.

Because of continued product development, the information contained in this document may change without notice. The information and intellectual property contained herein are confidential and remain the exclusive intellectual property of RenderX. If you find any problems in the documentation, please report them to us in writing. RenderX does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means - electronic, mechanical, photocopying, recording or otherwise - without the prior written permission of RenderX.

RenderX

Telephone: 1 (650) 328-8000

Fax: 1 (650) 328-8008

Website: <http://renderx.com>

Email: support@renderx.com

Table of Contents

1. Preface	7
1.1. What's in this Document?	7
1.2. Prerequisites	7
1.3. Acronyms	8
1.4. Technical Support	8
2. Overview	9
2.1. Overview	9
2.2. Introduction	9
2.3. Basic Terms	10
3. XEP Assistant	13
3.1. What is XEP Assistant?	13
3.2. Opening XEP Assistant	13
3.3. Rendering an XML File using XEP Assistant	13
3.3.1. Opening a File	13
3.3.2. Formatting a File	14
4. Using the Command Line	19
4.1. Running XEP	19
4.2. XEP Options	19
4.3. XEP Switches	20
4.4. XEP Arguments	21
4.5. Examples of Running XEP from the Command Line	22
5. Configuring XEP	25
5.1. Configuring XEP using XEP Assistant	25
5.1.1. Configuring Main Settings	25
5.1.2. Configuring Backends	27
Configuring the Backend for PDF Files	27
Configuring the Backend for PostScript Files	30
Configuring the Backend for AFP Files	33
5.1.3. Configuring Languages	36
5.1.4. Configuring Fonts	37
5.2. Configuring XEP via the XEP Configuration File	39
5.2.1. Configuration Structure	39
5.2.2. Core Options	40
5.2.3. Configuring Output Formats	42
Unicode Strings in Annotations (PDF, PostScript)	43
Initial Zoom Factor (PDF, PostScript)	44
PDF Initial View (PDF, PostScript)	45
Logical Page Numbering (PDF)	45
Page Layout (PDF)	45
PDF Viewer Preferences (PDF)	45
Treatment of Unused Destinations (PDF, PostScript)	46
ICC Profile (PDF)	46

PDF/X Support (PDF)	47
PDF/A Support (PDF)	47
Prepress Support (PDF, PostScript)	47
PDF Version (PDF)	48
Compression of PDF Streams (PDF)	49
Linearization (PDF)	49
Document Security (PDF)	49
PostScript Language Level (PostScript)	51
EPS Graphics Treatment (PostScript)	51
Page Device Control (PostScript)	51
Page Labeling (PostScript)	52
Inserting Custom Comments (PostScript)	52
Images Treatment in XML Output (XML)	53
5.2.4. Configuring Fonts	53
Fonts and Font Families	53
Font Groups	57
Font Aliases	57
5.2.5. Configuring Languages	57
Configuring Hyphenation	57
Language-Specific Font Aliases	58
5.3. Resolution of External Entities and URIs	58
6. XEP AFP Generator	59
6.1. Generating AFP Documents	59
6.2. Fonts	59
6.2.1. Font Mapping	59
6.3. Images	60
6.3.1. Image Support	60
6.3.2. Referencing Images	60
6.3.3. Image Clipping	60
6.4. Highlight Color Support	61
6.5. Graphics Support	61
6.6. Barcodes Support	65
6.7. FORMDEF Resource	67
6.7.1. What is a FORMDEF Resource?	67
6.7.2. Generating a Document with FORMDEF Resource	67
6.7.3. FORMDEF Processing Instructions	68
FORMDEF Syntax	68
6.8. Configuring the XEP AFP Generator	70
6.8.1. Configuring Character Sets	71
6.8.2. Configuring Fonts	74
6.8.3. Configuring Highlight Color Table	75
6.8.4. Configuring Shading Patterns	75
6.8.5. Other Configuration Options	77
6.9. Bullets support	79
6.10. International Character Set Support	80

6.11. Limitations of the XEP AFP Generator	81
6.12. Frequently Asked Questions	83
A. XSL-FO Conformance	85
A.1. XSL-FO Support	85
A.1.1. Formatting Objects Supported by XEP	85
A.1.2. Formatting Properties Supported by XEP	87
A.1.3. Notes on Formatting Objects Implementation	98
A.1.4. Supported Expressions	99
A.1.5. Color Specifiers	100
A.1.6. XSL 1.1 Support	102
Document Outline (Bookmarks)	103
Indexes	103
Last Page Number Reference	104
Change Bars	104
Folio Prefix and Suffix	104
A.1.7. Extensions to the XSL 1.0 Recommendation	105
Document Information	105
Document Outline (Bookmarks)	106
Indexes	106
Flow Sections	108
Last Page Number Reference	108
Change Bars	109
Background Image Scaling and Content Type	109
Initial Destination	110
Omitted Initial Header in Tables	110
Base URI Definition: <code>xml:base</code>	110
Border and Padding on Regions	110
Floats Alignment	110
Multicolumn Footnotes	111
Unique Footnotes	111
Watermark	111
Transpromo	112
B. Linguistic Algorithms	113
B.1. Line-Breaking Algorithm	113
B.2. Hyphenation	114
B.2.1. Hyphenation Patterns	114
B.3. Support for Right-to-Left Writing Systems	114
B.3.1. Bidirectionality	115
B.3.2. Glyph Shaping	115
C. Supported Fonts	117
C.1. Supported Fonts	117
C.1.1. PostScript Type 1 Fonts	117
PostScript Fonts and Unicode	117
Standard Adobe Fonts	119

C.1.2. TrueType Fonts	119
C.1.3. OpenType/CFF Fonts	120
C.1.4. Supported AFP Fonts	120
D. Supported Graphic Formats	121
D.1. Supported Graphic Formats	121
D.1.1. Bitmap Graphics	121
PNG	121
JPEG	121
GIF	122
TIFF	122
D.1.2. Vector Graphics	122
SVG	122
PDF	125
EPS	125
E. XEP Intermediate Output Format Specification	127
E.1. XEP Intermediate Output Format Specification	127
F. Accessibility Support in XEP	139
F.1. Accessibility Support in XEP	139
F.1.1. Tagged PDF	139
G. List of Output Generators' Options	141
G.1. List of Output Generators' Options	141
H. Configuration File DTD	143
H.1. Configuration File DTD	143
I. Additional Components	147
I.1. XEP Connector for jEdit version 2.1	147
I.1.1. Changes since version 1.*	147
I.1.2. Overview	147
I.1.3. Terms of use	147
I.1.4. Installation	147
I.1.5. Copyright notices	148
I.2. XEP ANT Task 2.0 User's Guide	148
I.2.1. Changes since version 1.*	148
I.2.2. Overview	148
I.2.3. Configuration	148
I.2.4. Parameters	149
I.2.5. Parameters specified as nested elements	150
I.2.6. Examples	150
Index	153

Chapter 1. Preface

1.1. What's in this Document?

The RenderX User Guide provides background information about what XEP does and explains how to use the product. The manual is divided into the following sections:

1. Overview
2. XEP Assistant
3. Using the Command Line
4. Configuring XEP
5. XEP AFP Generator

1.2. Prerequisites

XEP runs on most systems where Java Virtual Machine 1.1.8 or newer is available. This includes:

- Unixes;
- Microsoft Windows;
- Linux;
- Mac OS X;
- Other platforms and Operation Systems.

The Java edition of XEP requires a Java VM 1.2 or higher to run properly. Sun JRE version 1.3 or later is highly recommended. AFP Backend works with all versions of JRE 1.4 up to 1.6.0_01.

AFP Backend requires **charsets.jar** installed with JRE. By default, JRE is installed without **charsets.jar** file. Please run JRE installer and check the "**additional languages support**" checkbox.

Note: Actual checkbox name may vary for different versions of JRE.

XEP 4.13 demonstrates best performance running under JRE 1.6. This version of JRE is shipped within XEPWin distributions.

In order to view PDF output, a viewer is required. Adobe provides a free one which can be downloaded and installed from the [Adobe website](#).

To view PostScript files, one option is to use [GhostView](#), which may be used for viewing PDF as well. Versions are available for most operating systems.

1.3. Acronyms

The following table lists acronyms used in this manual:

Table 1.1. Acronyms

Acronym	Full Term
CJK	Chinese Japanese Korean (Unicode UTF-8 encoding standard for Asian character set)
CMYK	Cyan-Magenta-Yellow-Key/black (4-color ink model used for printing)
DTD	Document Type Definition
IPA	Internet Protocol Address
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator (world wide web address)
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSL-FO	eXtensible Stylesheet Language Formatting Objects
JRE	Java Runtime Environment
JDK	Java Development Kit

1.4. Technical Support

You can contact RenderX technical support by:

- Using the RenderX support portal at <http://renderx.com/support/index.html>
- Sending an email to support@renderx.com
- Calling 1 (650) 328-8000

Chapter 2. Overview

2.1. Overview

This section contains introductory information about XEP.

2.2. Introduction

XEP is a library of Java classes that converts XML data to printable formats, such as PDF, PostScript and AFP. XEP accepts either an XSL-FO file, or an XML file paired with an XSL stylesheet, as input. In the latter case, XEP uses an internal XSLT transformer to preprocess the XML file according to the XSL stylesheet, thereby converting it to an XSL-FO file. The XEP engine then processes the XSL-FO file.

The logical flow of document processing can be divided into three phases, as illustrated in the following figure:

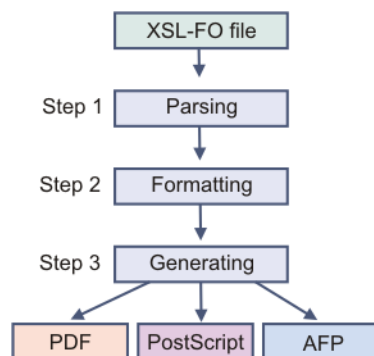


Figure 2.1. Three-step process

- **Parsing** - XEP reads the XSL-FO file and creates an internal representation of the file in memory.
- **Formatting** - The XSL-FO is fed into the formatter which creates and fills pages according to the specification defined in the XSL-FO document. Results of the formatting stage can be output as XML to be further processed later.
- **Generating** - The XSL-FO file is converted to the requested output format - PDF, PostScript or AFP.

XEP can be run in three different environments:

- **XEP Assistant** - XEP includes a GUI-based tool for more comfortable transformation of files, suitable for users that prefer graphic interface. For a detailed description, refer to [Chapter 3, XEP Assistant](#).

- **Command Line** - XEP can be run from the command line as described in [Chapter 4, Using the Command Line](#).
- **Integration** - XEP can be integrated into other tools.

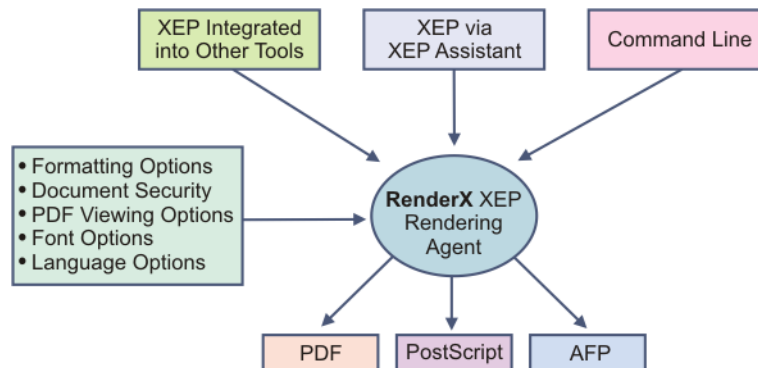


Figure 2.2. XEP

XEP can be configured to allow users to apply settings, such as fonts, languages and formatting options, according to their preferences. For a detailed reference please refer to [Chapter 5, Configuring XEP](#).

2.3. Basic Terms

This section provides an introduction to the basic terms that are used throughout this documentation.

PDF (Portable Document Format)

PDF is a universal file format that preserves the fonts, images, graphics, and layout of any source document, regardless of the application and platform that were used to create it. See the Adobe Web site <http://www.adobe.com> for more information.

PostScript®

Adobe® PostScript® is the worldwide printing and imaging standard. Used by print service providers, publishers, corporations and government agencies around the globe, Adobe PostScript 3 gives you the power to print visually-rich documents. See the Adobe Web site <http://www.adobe.com> for more information.

AFP (Advanced Function Printing)

AFP is an architecture standard for High Volume Transaction Output, supported by such vendors of printing equipment as IBM, Kodak and Xerox. AFP has built-in support for text and raster graphic output, vector graphic, vector and raster fonts, as well as many other features. The entire document structure of AFP document is organized by means of a higher level protocol called MO:DCA which links all printable objects together and builds the whole document.

XSL-FO

XSL, Formatting objects. A standard way of specifying how content should be presented. A World Wide Web Consortium specification. See the W3C website <http://www.w3.org/TR/xsl/> for more information.

Chapter 3. XEP Assistant

3.1. What is XEP Assistant?

XEP contains a user-friendly GUI tool, called XEP Assistant. Use of XEP Assistant simplifies rendering from XML or XSL-FO into the desired output format.

3.2. Opening XEP Assistant

To open XEP Assistant, browse to the XEP Installation directory and launch `x4u.bat` or `x4u` bash script.

3.3. Rendering an XML File using XEP Assistant

3.3.1. Opening a File

To render a file, first of all, you must open the XML or XSL-FO file you wish to publish.

To open an existing XML or XSL-FO file:

1. From the main menu, click **File**.

The **File** menu is displayed.

2. From the **File** menu, click **Open**.

A dialog box is displayed.

3. Browse to the file you wish to open.

The file is opened within XEP Assistant.

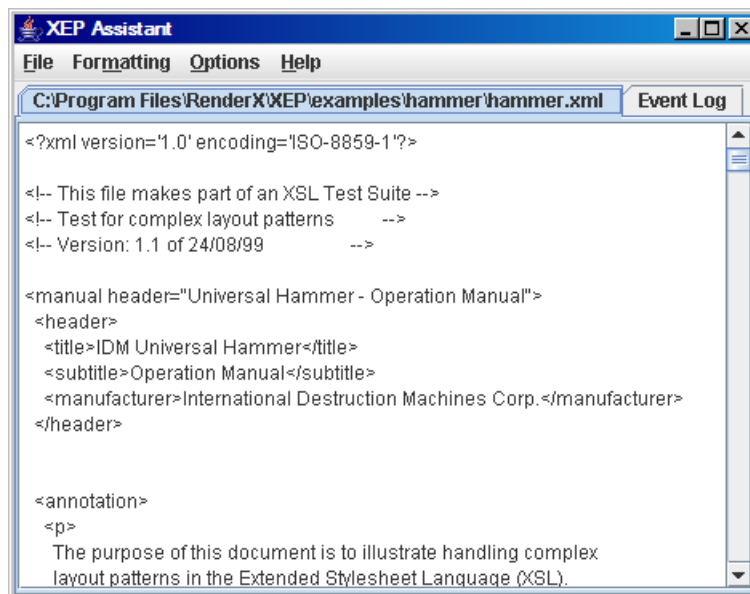


Figure 3.1. XML file displayed in XEP Assistant

3.3.2. Formatting a File

Once an XML file is open, it must first be "transformed" before it can be formatted to PDF, PostScript or AFP output. "Transforming" refers to the assignment of various settings required to apply an XSL stylesheet to your XML file. The result of the transforming is that the XML file is transformed into an XSL-FO. The XSL-FO is then formatted to your final output format (PDF, PS or AFP).

To format an XML file:

1. From the main menu, click **Formatting**.

The **Formatting** menu is displayed.

2. From the **Formatting** menu, click **Start**.

The **Formatting settings** dialog box appears.

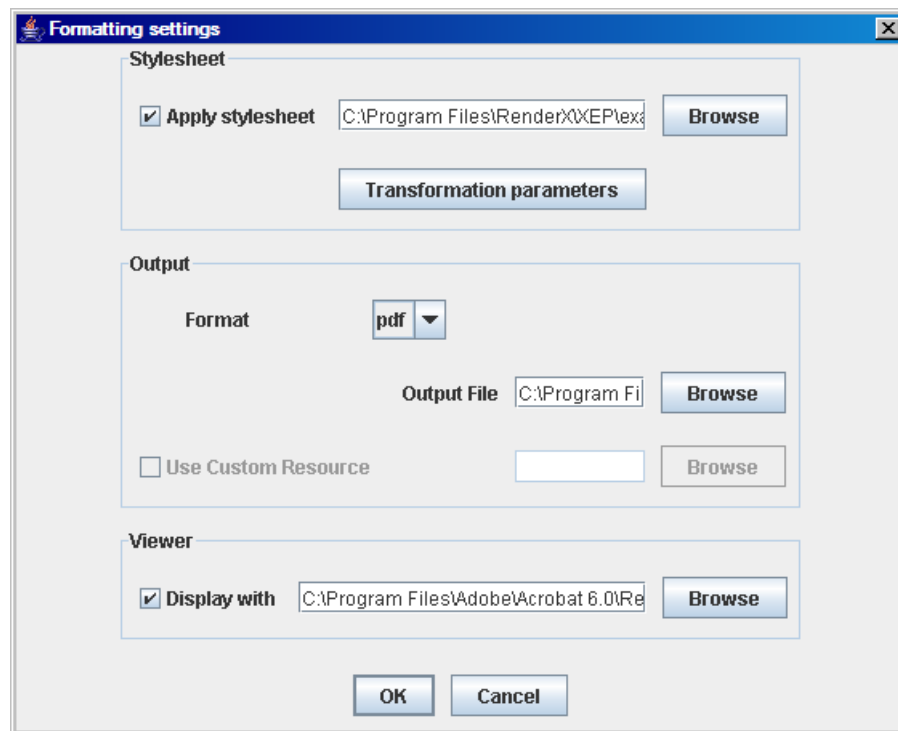


Figure 3.2. Formatting settings dialog box

- Set the desired settings as described in the following table.

Table 3.1. Formatting Settings

Parameter	Description
Stylesheet	
Apply stylesheet	Check the Apply stylesheet checkbox to apply a stylesheet (XSL) to the XML file. Click Browse to browse to the location of the XSL file you wish to apply as a stylesheet to your XML file.
Transformation parameters	This button is only enabled when the Apply Stylesheet checkbox is selected. Refer to Figure 3.3, “XSL Parameters” , and Table 3.2, “XSL Parameters” for a complete description.
Output	
Format	Select the format to which you want to render the XML file. Available options are PDF, PS and AFP.
Output File	Select the location and name of the file to which the output will be saved. The default output file name is the identical path and

Parameter	Description
	file name as the current XML file with the file extension of the chosen output type. Note: If a file with the same name already exists in the chosen location, the new file will overwrite the preexisting file with no warning.
Set Resource	The Use Custom Resource section is only enabled when AFP is selected as the output format. Click Browse to select the location of the resource file. A resource can be attached to an AFP document to control certain reusable objects like images or FORMDEFs.
Viewer	
Display With	Check the Display With checkbox to automatically display the output once rendering is complete. Browse to the location of the program with which you wish to view the rendered file.

- Click **OK** to format the file, and **Cancel** to cancel the formatting.

To add XSL parameters:

- From the **Formatting settings** dialog box, click the **Transformation parameters** button (only enabled when the **Apply stylesheet** checkbox is selected).

The **XSL Parameters** dialog box appears.

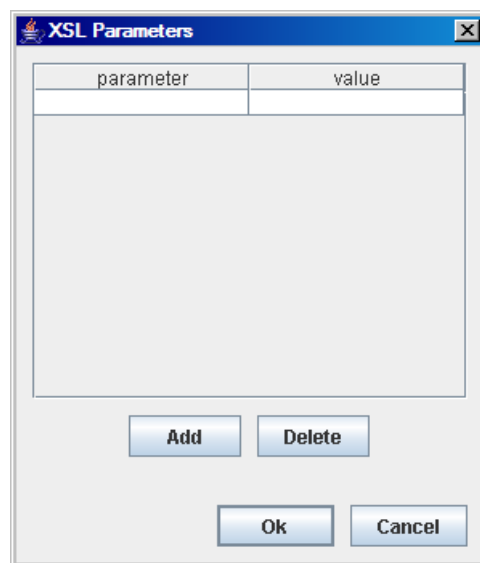


Figure 3.3. XSL Parameters

2. Fill in the fields as described in the following table:

Table 3.2. XSL Parameters

Field	Description
parameter	The name of the variable used in the XSL file to represent a parameter value.
value	The value corresponding to the variable.

3. Click **Add** to add a new parameter, or highlight a parameter and click **Delete** to delete the selected parameter.
4. Click **OK** to apply changes, or click **Cancel** to close the dialog box without applying your changes.

To cancel formatting:

1. From the main menu, select **Formatting**.

The **Formatting** menu is displayed.

2. Select **Stop**.

Formatting is canceled.

Chapter 4. Using the Command Line

This topic describes how to run XEP from the command line.

4.1. Running XEP

XEP can be run from the command line, as follows:

- On all platforms, by invoking Java directly from the command line.
- On Windows, XEP can be run from a Command Prompt window via the `xep.bat` batch file.
- On Linux, MAC, and UNIX, XEP can be run from a command shell, via the `xep` bash script.

To learn more about the `xep.bat` batch file or the `xep` bash script, open the file in a text editor. These files use standard scripting features available in the operating system.

The syntax of the Java command is:

```
java com.renderx.xep.XSLDriver {options} {switches} {arguments}
```

The above syntax has been simplified by assuming that the directory containing the Java executable is specified in your `PATH` environment variable, and that the full path of the `xep.jar` file is in your `CLASSPATH` environment variable. If you specify an XSL file to convert an XML source document into XSL-FO, then it is assumed that `saxon.jar` or `xt.jar` are also specified in your `CLASSPATH` environment variable.

The syntax of the Windows batch and Linux/MAC shell command is:

```
xep {options} {switches} {arguments}
```

The above syntax assumes that the full path to the Windows batch file `xep.bat` or the Linux/MAC shell script `xep` is specified in the `PATH` environment variable, or that the current directory is the directory containing the Windows batch file or Linux/MAC shell script.

The options, switches, and arguments are the same whether XEP is run via Java, via a Windows batch file, or via a Linux/MAC shell script.

4.2. XEP Options

The XEP options are used to configure and customize the behavior of the XEP rendering engine.

XEP requires a configuration file in order to run. By default, the formatter looks for a file named `xep.xml` in the current directory. If a different configuration file is used, the path to the configuration file must be specified on the command line.

XEP is a flexible tool in which the configuration can be customized according to your preferences. There are several methods to customize XEP. These methods are summarized in the following table:

Table 4.1. Customizing XEP Configuration

Customization	Description	Syntax
Editing the configuration file.	<p>The <code>xep.xml</code> configuration file can be customized, thereby customizing all transformations. There are two ways to customize the file:</p> <ul style="list-style-type: none"> Edit the <code>xep.xml</code> file in a text editor. From the Options tab in the XEP Assistant. 	<p>For editing <code>xep.xml</code> in a text editor, see Section 5.2, “Configuring XEP via the XEP Configuration File”.</p> <p>For the XEP Assistant, see Section 5.1, “Configuring XEP using XEP Assistant”.</p>
Setting a custom configuration file.	<p>You can set a custom configuration file in the command line for a single file transformation. The location of the custom configuration file can be specified as either a file name in the local file system or as a URL.</p> <p>All subsequent file transformations will continue to use the standard <code>xep.xml</code> file.</p>	<code>-DCONFIG=<CUSTOM_FILE_PATH></code>
Customizing the XEP configuration through the command line.	<p>In the command line, the configuration can be customized for a single file transformation. The <code>xep.xml</code> file is not changed, and all subsequent file transformations are not affected.</p> <p>Note: It is possible to specify multiple options in the same command line.</p> <p>Note: If there is a contradiction between the configuration file and the customization through the command line, the command line overrides the settings specified in the configuration file.</p>	<code>-D<OPTION_NAME>=<OPTION_VALUE></code>

Note: If any string contains spaces, the entire string must be enclosed in quotation marks.

4.3. XEP Switches

The XEP switches configure the behavior of the command line utility.

Table 4.2. XEP Switches

Switch	Description
-help	Displays the detailed syntax of the XEP switches and arguments.
-hosted	The Java Virtual Machine continues to run after the renderer has completed rendering the file.
-quiet	By default, XEP writes detailed messages to the command line console. These messages indicate the current status and progress of the rendering process, as well as any warnings or errors that may occur during the rendering process. Specify this switch to suppress the detailed informational messages. In this case, the renderer outputs only warning and error messages.
-valid	The validation is turned off; XEP does not validate the input when rendering. Note: The rendering runs faster, but since the XML source is not validated, there is a chance that the output will not be correct.
-version	Displays detailed version information of the XEP rendering engine.

4.4. XEP Arguments

The XEP arguments instruct XEP how to process a file. For example, arguments may specify the input file, the target format to render to, and the output filename. When multiple arguments are specified, they must be specified in the following order:

```
( [-xml] <infile> [-xsl <stylesheet>] {-param <name=value>}
    | -fo <infile>
    | -xep <infile> )
[-format]
[[-<output format>] <outfile>]
```

The XEP arguments are described in the following table.

Table 4.3. XEP Arguments

Argument	Description
-xml	The specified input file is an XML source document. When the input file is an XML document, this argument may be omitted.
-fo	The specified input file is an XSL-FO document ready to be rendered.
-xep	The specified input file is an XEP file, generated previously using the at output format. The XEP file is an XML document that is an internal representation of the rendered document.
<infile>	Specifies the input file. This argument is required.

Argument	Description
-xsl <stylesheet>	Specifies the XSL stylesheet XEP must use to transform the input XML document into XSL-FO. <stylesheet> is the path (absolute or relative to the working directory) of the XSL stylesheet.
-param <name=value>	If the XSL stylesheet supports global parameters, they can be set via the -param argument. Each XSL parameter you want to set on the command line requires a separate -param argument.
-<output format>	Specifies the output format to render to. Available output formats are: XEP, PDF, PostScript (PS), and AFP. <ul style="list-style-type: none"> • <code>at</code> — Internal XEP format. This is an XML file that represents the rendered document. • <code>pdf</code> — PDF format. This is the default output format if no output format is specified. • <code>afp</code> — AFP format. AFP is an architecture standard for High Volume Transaction Output supported by vendors of printing equipment. • <code>ps</code> — PostScript format. PostScript is useful when preparing a file to send to a printing service provider. • <code>xep</code> — Internal XEP format (same as <code>at</code>).
-format	Another way of specifying the output format.
-<outfile>	Specifies the path and name of the output file. If no output file is specified, the default output file is the same file path and name as the input file with the extension corresponding to file format.

4.5. Examples of Running XEP from the Command Line

This section presents a number of examples of how to run XEP from the command line.

To list all available options and switches:

- At the system prompt, enter:

xep -help.

A list of all available commands is displayed.

```
c:\myfiles>xep -help
XEP 4.13 build 20080425
java com.renderx.xep.XSLDriver
```

```

    {<option>}
    {-quiet | -version | -valid | -hosted | -help}
    ( [-xml] <infile> [-xsl <stylesheet>] {-param <name=value>}
      | -fo <infile>
      | -xep <infile> )
    [-f]
    [[-<output format>] <outfile>]
Available output formats: at (XEP), pdf (PDF), afp (AFP), ps (Postscript),
xep (XEP).

```

To view the version of XEP you are currently running:

1. At the system prompt, enter:

xep -version.

The version you are currently running as well as the build are displayed.

```

c:\myfiles>xep -version
XEP 4.13 build 20080814
(document [system-id file:stdin]

```

2. Press **<Ctrl> + <C>** to exit XEP interactive mode.

To render an XML document to PDF:

- To render the XML document `CommandLine.xml` to PDF, using the stylesheet `custom-fo.xsl` to transform the XML to an XSL-FO document and relying on the default settings for the output format and output filename, at the system prompt, enter:

xep CommandLine.xml -xsl custom-fo.xsl

Chapter 5. Configuring XEP

5.1. Configuring XEP using XEP Assistant

This section describes how to configure XEP according to your preferences by using XEP Assistant.

To configure XEP:

1. From the main menu, select **Options**.

The **Options** menu is displayed.

2. From the **Options** menu, select **Edit**.

The **XEP Configuration** dialogue box is displayed.

3. Click the **Main** tab, the **Backends** tab, the **Languages** tab or the **Fonts** tab.

For	See
Main tab	Section 5.1.1, “Configuring Main Settings”
Backends tab	Section 5.1.2, “Configuring Backends”
Languages tab	Section 5.1.3, “Configuring Languages”
Fonts tab	Section 5.1.4, “Configuring Fonts”

4. Configure the required parameters and click **Save** to save and close, or **Exit** to close without saving your changes.

5.1.1. Configuring Main Settings

The default configuration settings can be set in the **Main** tab.

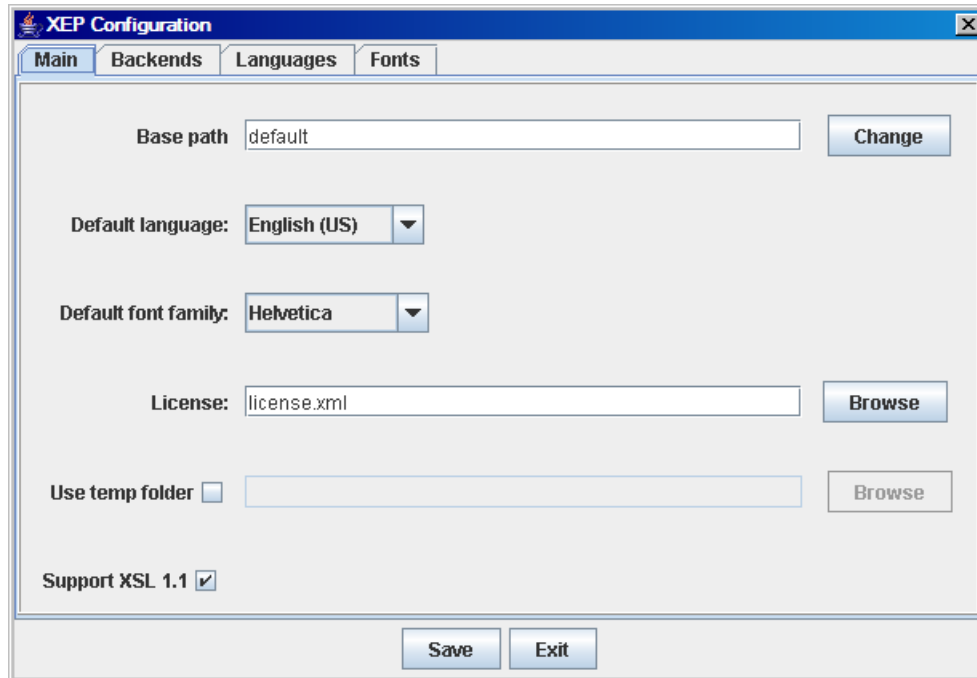


Figure 5.1. XEP Configuration Main tab

Table 5.1. XEP Configuration Main Tab Parameters

Parameter	Possible Values	Description
Base Path	free text	The location of the configuration file (<code>XEP.xml</code>). The Base path is used to resolve relative URLs where parameters accept URLs as values. Click Change to select the location of the configuration file.
Default Language	all supported languages, unspecified. Default: English (US)	Select the language to use when no language is specified.
Default font family	all supported font families, unspecified.	Select the font family to use when no font family is specified.
License	free text	The location of the XEP license file. Click Browse to select the location of the license file.
Use temp folder	checked, unchecked Default: unchecked	Check to enable writing temporary files to disk. Once checked, click Browse to set the path to the directory where the temporary files are written.

5.1.2. Configuring Backends

Using Backends, you can control certain properties in the output documents. There are different available properties for each output type. Select the output type and then configure the properties for the specific output type selected. Refer to the appropriate figure and table for more information on each output type.

To select the output type:

1. On the **Backends** tab, click **Select backend**.
2. Select **PDF**, **PostScript**, or **AFP**.

The **Backend Parameters** screen populates with parameters based on the selected backend.

For	See
PDF	Configuring the Backend for PDF Files
PostScript	Configuring the Backend for PostScript Files
AFP	Configuring the Backend for AFP Files

Configuring the Backend for PDF Files

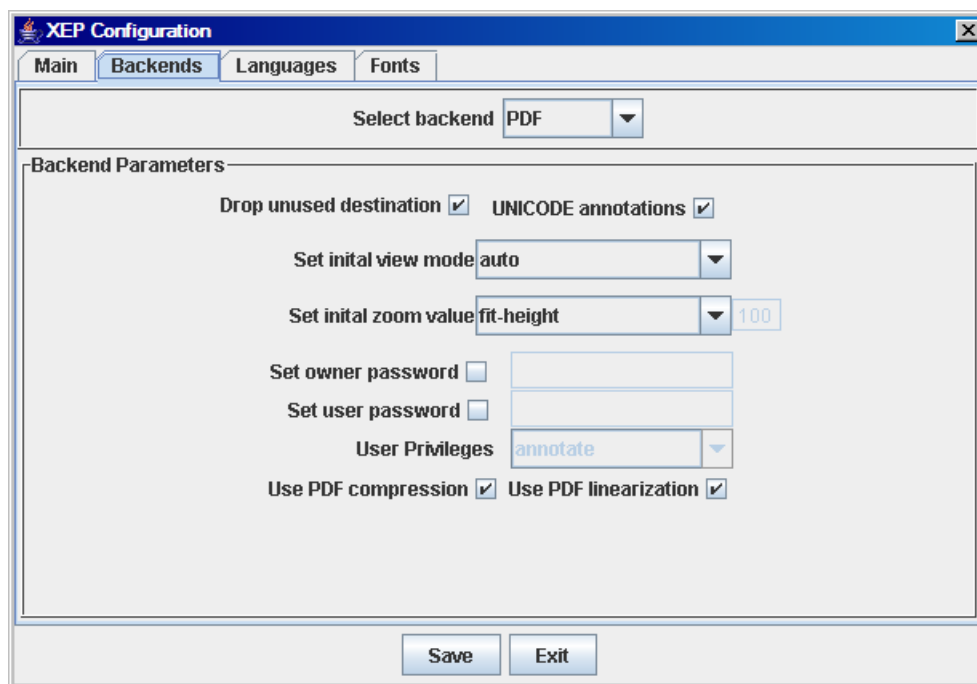


Figure 5.2. XEP Configuration Backend's tab for PDF files

Table 5.2. XEP Configuration Backends Tab PDF Parameters

Parameter	Possible Values	Description
Select backend	PDF, PostScript, AFP Default: PDF	Select the output type for which you are configuring the backend.
Backend Parameters		
Drop unused destination	checked, unchecked Default: checked	Specify whether named destinations are created for objects not referenced within the document.
UNICODE annotations	checked, unchecked Default: checked	Enable or disable use of Unicode to represent PDF annotations strings, such as bookmark text and document info.
Set initial view mode	<ul style="list-style-type: none"> • auto - If there are bookmarks in the document, the bookmark pane is displayed. Otherwise, all auxiliary panes are hidden. • show-none - All auxiliary panes are hidden. • show-bookmarks - The bookmarks pane is displayed. • show-thumbnails - The thumbnails pane is displayed. • full-screen - The document is displayed in full-screen mode. Default: auto	Set the view mode to be activated in the PDF viewer when the PDF file is rendered and viewed.
Set initial zoom value	<ul style="list-style-type: none"> • auto - Page scaling is not specified. • fit - The page is scaled to fit completely into the view port. • fit-width - The page is scaled so that its width matches the width of the view port. • fit-height - The page is scaled so that its height matches the height of the view port. 	Specify the magnification factor to be applied when the file is first opened in the PDF viewer.

Parameter	Possible Values	Description
	<ul style="list-style-type: none"> number-or-percentage - The page is scaled by the number or percentage specified in the enabled box. <p>Default: auto</p>	
Set owner password	<p>checked, unchecked</p> <p>Default: unchecked</p> <p>If the check box is checked, then the text box is enabled so that you can type in a password.</p>	Select this option to set an owner password for the PDF document. Owner passwords give the owner full control over the PDF document.
Set user password	<p>checked, unchecked</p> <p>Default: unchecked</p> <p>If the check box is checked, then the text box is enabled so that you can type in a password.</p>	Select this option to set a user password for the PDF document. Holders of user passwords are subject to access restrictions specified in User Privileges .
User Privileges	<ul style="list-style-type: none"> annotate - Enables adding annotations to the document and changing form field values. copy - Enables copying text and images from the document onto the clipboard. modify - Enables editing the document. Print - Enables printing the document. <p>Default: annotate</p>	Select the privilege for users accessing the resulting document with user password.
Use PDF compression	<p>checked, unchecked</p> <p>Default: checked</p>	Check to compress content streams in PDF using Flate algorithm.
Use PDF linearization	<p>checked, unchecked</p> <p>Default: unchecked</p>	Check to linearize (or optimize for the Web) the PDF output.

Configuring the Backend for PostScript Files

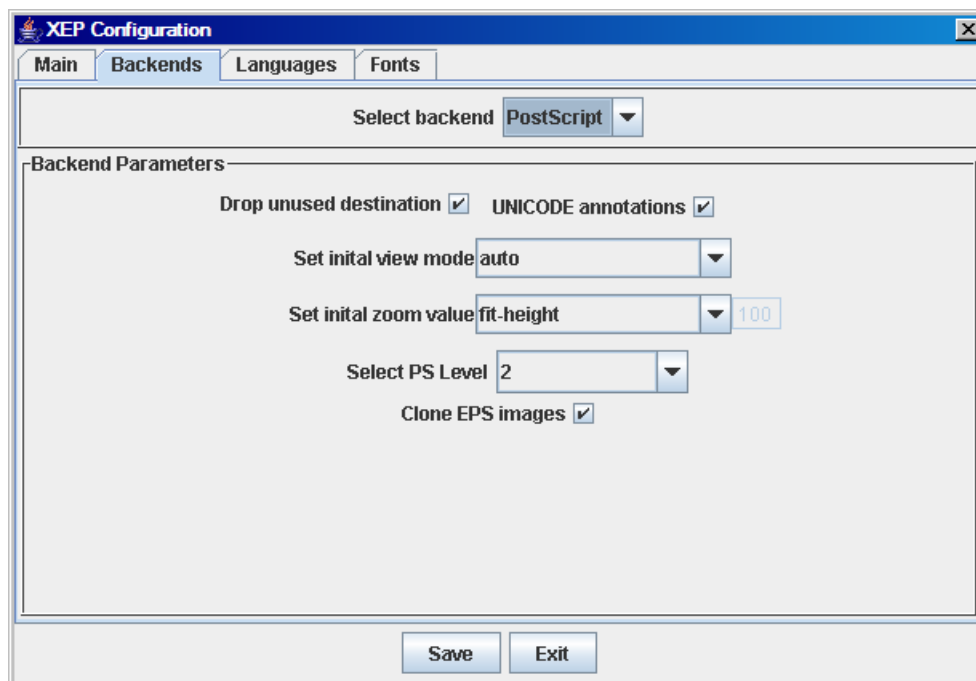


Figure 5.3. XEP Configuration Backends tab for PostScript files

Table 5.3. XEP Configuration Backend Tab for Configuring PostScript Parameters

Parameter	Possible Values	Description
Select backend	PDF, PostScript, AFP Default: PDF	Select the output type for which you are configuring the backend.
Backend Parameters		
Drop unused destination	checked, unchecked Default: checked	Specify whether named destinations are created for objects not referenced within the document. This information is utilized when the file is further converted to PDF.
UNICODE annotations	checked, unchecked Default: checked	Enable or disable use of Unicode to represent PDF annotations strings, such as bookmark text, and document info. This information is utilized when the file is further converted to PDF.

Parameter	Possible Values	Description
Set initial view mode	<ul style="list-style-type: none"> • auto - If there are bookmarks in the document, the bookmarks pane is displayed. Otherwise, all auxiliary panes are hidden. • show-none - All auxiliary panes are hidden. • show-bookmarks - The bookmarks pane is displayed. • show-thumbnails - The thumbnails pane is displayed. • full-screen - The document is displayed in full-screen mode. <p>Default: auto</p>	<p>The PDF document may contain definition of default view mode which is activated by the PDF viewer upon rendering and viewing the file. This option allows specifying this mode.</p> <p>This information is utilized when the file is further converted to PDF.</p>
Set initial zoom value	<ul style="list-style-type: none"> • auto - Page scaling is not specified. • fit - The page is scaled to fit completely into the view port. • fit-width - The page is scaled so that its width matches the width of the view port. • fit-height - The page is scaled so that its height matches the height of the view port. • number-or-percentage - The page is scaled by the number or percentage specified in the enabled box. <p>Default: auto</p>	<p>Specify the magnification factor to be activated when the file is first opened in the PDF viewer.</p> <p>This information is utilized when the file is further converted in PDF.</p>
Select PS Level	<p>2,3</p> <p>Default: 3</p>	<p>Select the target PostScript language level.</p> <p>Note: If the language level is set to 2, some advanced features and improved font definitions are not available.</p>

Parameter	Possible Values	Description
Clone EPS images	<ul style="list-style-type: none">• checked - EPS graphics are pasted into the output stream at each occurrence. This may lead to a substantial growth of the resulting file size.• unchecked - EPS graphics are posted into the PostScript form. This minimizes the file size, however, some EPS images cannot be processed this way and it may corrupt the PostScript code. <p>Default:checked</p>	Specify whether EPS graphics are included in the PostScript output using the forms mechanism, or by pasting their contents at each occurrence.

Configuring the Backend for AFP Files

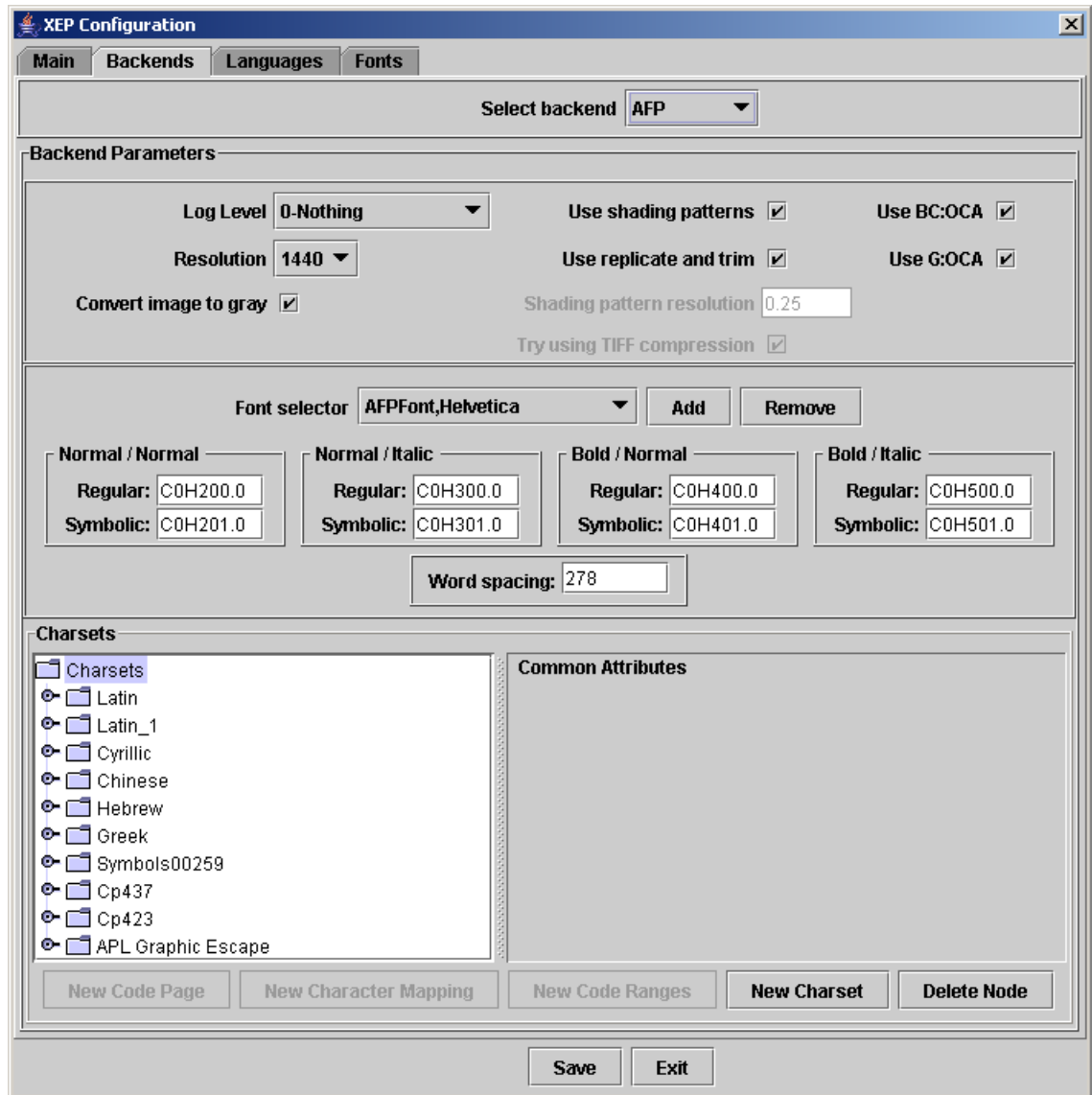


Figure 5.4. XEP Configuration Backends tab for AFP files

Table 5.4. XEP Configuration Backends Tab AFP Parameters

Parameter	Possible Values	Description
Select backend	PDF, PostScript, AFP Default: PDF	Select the output type for which you are configuring the backend.

Parameter	Possible Values	Description
Backend Parameters		
Log Level	0,1,2 Default: 0 (Nothing)	Select a number to determine the level of log detail. <i>0</i> - Nothing <i>1</i> - Warnings only <i>2</i> - Warnings and information messages
Resolution	positive integer Default: 1440	Defines which document resolution will be output within the document. It must be positive integer value supported by target AFP device
Convert images to gray	checked, unchecked Default: unchecked	If checked, turns on embedding of raster images as grayscale images, 8 bit per pixel, uncompressed. Unchecked - embed raster images in their original format
Use shading patterns	checked, unchecked Default: checked	Specifies whether grayscale-filled areas should be filled with bi-level pattern. Percentage rate of black points will be closest match to required grayscale value. <i>Checked</i> - shading patterns will be used <i>Unchecked</i> - shading patterns will not be used
Use replicate and trim	checked, unchecked Default: unchecked	property specifies whether the "replicate-and-trim" feature will be used for shading patterns. <i>Checked</i> - "replicate-and-trim" is used <i>Unchecked</i> - "replicate-and-trim" is not used
Shading pattern resolution	floating point number Default: 1.0	Defines zoom factor for shading pattern raster.
Try using TIFF compression	checked, unchecked Default: checked	This option allows the user to specify whether AFP backend attempts to compress shading patterns raster images with TIFF encoding. <i>Checked</i> - AFP Backend attempts compressing shading pattern rasters <i>Unchecked</i> - AFP Backend does not attempt compressing shading pattern rasters

Parameter	Possible Values	Description
Use BC:OCA	checked, unchecked Default: checked	Defines the upper level of BC:OCA commands subset. <i>Unchecked</i> - Do not use BC:OCA commands <i>Checked</i> - Use Level 1 only
Use G:OCA	checked, unchecked Default: checked	Defines the upper level of G:OCA commands subset. <i>Unchecked</i> - Do not use G:OCA commands <i>Checked</i> - Use Level 1 only

Please refer to [Section 6.8, “Configuring the XEP AFP Generator”](#) of this document for details.

AFP Fonts

To view and edit an AFP font and its sub values:

1. Click the **AFPFonts** drop down box (see [Figure 5.4, “XEP Configuration Backends tab for AFP files”](#)).
2. Select the font you wish to view/edit.

Note: AFP font names are comprised of the word <AFPFont> followed by a comma and the XEP font name, such as <AFPFont, Verdana>.

All sub values are populated based on the font selected.

3. View or edit all AFP font sub values.

To add an AFP font:

1. Click **AddAFPFont** (see [Figure 5.4, “XEP Configuration Backends tab for AFP files”](#)).

A dialog box opens containing a list of all supported fonts as displayed in the following figure:

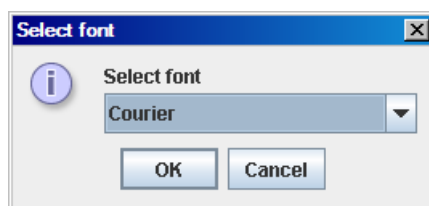


Figure 5.5. Add Font

2. Select the font you wish to add to the AFP fonts.

3. Click **OK** to add the font or **Cancel** to close the box without adding a new font.

The selected font is added.

To remove an AFP font:

1. From the **AFPFonts** drop down box, select the font you wish to remove (see [Figure 5.4, “XEP Configuration Backends tab for AFP files”](#)).
2. Click **RemoveAFPFont**.

The selected font is removed.

5.1.3. Configuring Languages

Languages can be configured in the **Languages** tab.

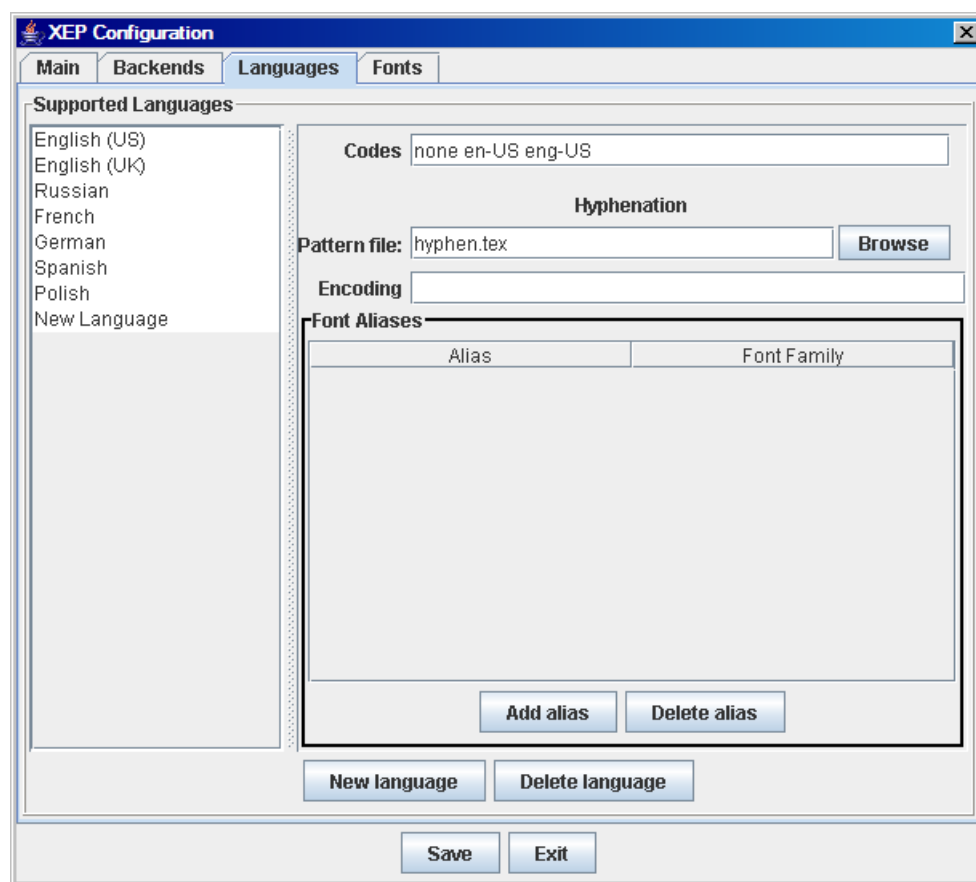


Figure 5.6. XEP Configuration Languages tab

Table 5.5. XEP Configuration Languages Tab

Parameter	Possible Values	Description
Supported Languages		
Codes	free text	A list of codes used to refer to the language in the XSL-FO input data. Note: Separate multiple codes with spaces.
Pattern File	free text	The location of the <code>Pattern</code> file associated with the language selected. Click Browse to select the location of the <code>Pattern</code> file.
Encoding	Default: ISP-8859-1	The encoding of the pattern file.
Font Aliases	Font aliases are activated when the language which is associated with them is selected. They take precedence over the aliases specified in the fonts section and may mask them.	
Alias	free text	Provide an alternate name for a font family.
Font Family	free text	Select the font family corresponding to the alias.
Add alias		Click to add a new alias.
Delete alias		To delete an alias, highlight the alias you wish to delete and click Delete alias .

5.1.4. Configuring Fonts

Fonts are categorized into families, which is the basic configuration unit in XEP, and then further into groups. A font family is a set of fonts that share a common design but differ in stylistic attributes, such as upright or italic, light or bold. A group consists of several font families wrapped into one container element. Groups can be nested, forming complex font hierarchies.

In the left column, there is the font hierarchy that contains groups, families, and fonts. Click on a node to display and edit its common attributes. Double-click a node to open its children.

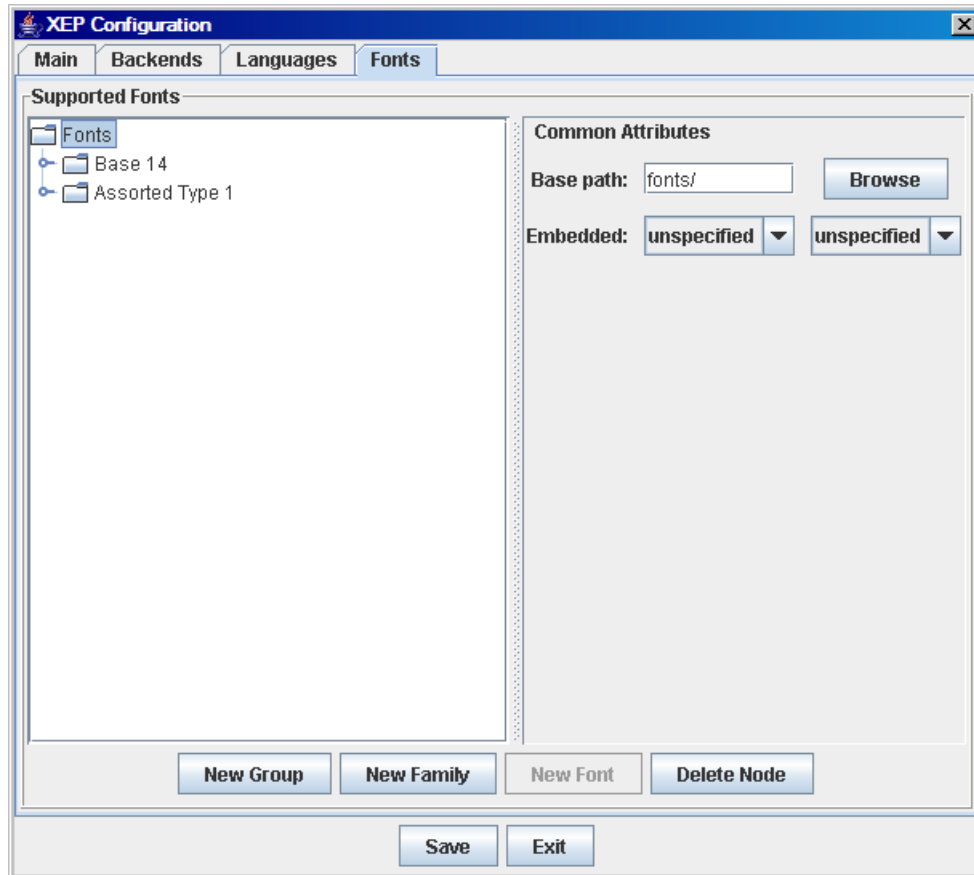


Figure 5.7. XEP Configuration Fonts tab

Table 5.6. XEP Configuration - Fonts Tab Parameters

Parameters	Possible Values	Description
Supported Fonts		
Common Attributes		
Base Path	free text	Specifies a common base directory for a group of font families that form a package. Click Browse to select a file location.
Embedded	unspecified, true, false	Specifies whether the font is embedded in the document or it is external to the file. Note: If the font is external, the rendered file can only be viewed on systems that have the font configured for use with viewing or printing the application.
Subsetted	unspecified, true, false	Specifies whether the font is subsetted. Note: If a font is subsetted, the file is not editable.

Parameters	Possible Values	Description
Font Aliases		
Alias	free text	Provide an alternate name for a font family.
Font Family	all font families defined	Select the font family corresponding to the alias.
Add alias		Click to add a new alias.
Delete alias		To delete an alias, highlight the alias you wish to delete and click Delete alias .
New Group		Click to create a new group.
New Family		Click to create a new family.
New Font		Click to create a new font.
Delete Node		Click on the node you wish to delete and click Delete Node to delete.

5.2. Configuring XEP via the XEP Configuration File

This topic describes in detail how to configure XEP by creating or modifying an XEP configuration file.

5.2.1. Configuration Structure

XEP is controlled by a single configuration file which contains core formatting options, fonts available to the formatter, and language-specific data.

The XEP configuration file must always be accessible to the formatter. Methods for locating the configuration file are platform-dependent. Please refer to specific platform documentation for details. By default, the formatter looks for a file named `xep.xml` in the directory where it is currently running.

The configuration file is an XML document in a special namespace: "http://www.renderx.com/XEP/config". The root of the configuration file is a `<config>` element which includes three major subsections:

- `<options>` - Options for XEP rendering core and backends are defined inside the `<options>` element.
- `<fonts>` - Fonts configuration is contained inside the `<fonts>` element.
- `<languages>` - Hyphenation and language-dependent parameters are configured in the `<languages>` element.

Some parameters can accept URLs as values. In such cases, the location of the configuration file is used as a base to resolve relative URLs. The base URL can be overridden for any subtree of the configuration file, by utilizing the [xml:base](#) attribute.

Note: All relative URLs in parameter values stored in a referenced file are resolved with respect to that file, rather than the top-level configuration file. Attribute [xml:base](#) in the referrer file has no effect on URLs that are contained in another file.

The use of a monolithic configuration file is usually the most convenient way to store the configuration, as it simplifies switching between different XEP configurations, and facilitates environmental tuneup. However, occasionally it may be wiser to move parts of the configuration into separate files, such as when font configuration is reused across multiple setups. The configuration file supports modularization. Any container element can be moved into a separate XML file whose location is specified by an `href` attribute.

5.2.2. Core Options

XEP is controlled by several **options** which can be set in the configuration file. An option is defined by an `<option>` element. It has a name and an associated value: `name=value`. XEP core options are always specified as direct children of the `<options>` element. The following core options are defined for XEP 4.13:

Table 5.7. Core Options

Option	Possible Values	Description
<i>LICENSE</i>	free text Default: <code>license.xml</code>	<p>The location of the license file.</p> <p>At startup, XEP looks for a license file, and only runs if the signature on the license matches the public key associated with the specific edition of the formatter. Additionally, this file is used as an access key to XEP online update service.</p> <p>The parameter can be specified either as a file name in the local file system, or as a URL. In addition to common protocols, <code>data:</code> and <code>resource:</code> URL schemes are supported.</p>
<i>VALIDATE</i>	true, false Default: true	<p>Controls the input validation.</p> <p>Caution</p> <p>In non-validating mode, XEP uses less memory, and runs faster. However, less errors are intercepted, and the results of formatting are less predictable for malformed print. This setting is discouraged unless your stylesheets are thoroughly debugged.</p>

Option	Possible Values	Description
<i>DIS-CARD_IF_NOT_VALID-ID</i>	true, false Default: true	Controls the termination of processing upon unsuccessful validation.
<i>STRICTNESS</i>	<ul style="list-style-type: none"> • 0 - Relaxed • 1 - Normal • 2 - Strict Default: 1	Determines the validator's level of strictness.
<i>SUPPORT_XSL11</i>	true, false Default: true	Turns on/off XSL-FO 1.1 support. Note: If false is selected, XEP runs faster and uses less memory.
<i>TMPDIR</i>	Default: none	The path to the directory for temporary files. If set, this parameter must point to a writable directory, specified either as a path in the local file system or as a file URL. To disable writing temporary files to disk, specify none as the value for this option. Note: To avoid file name clashes, a separate temporary directory should be specified for each process running XEP.
<i>BROKENIMAGE</i>	free text Default: images/404.gif	The icon inserted as a replacement for broken or missing images. The parameter can be specified either as a file name in the local file system, or as a URL. In addition to the common protocols, data: and resource: URL schemes are supported.
<i>PAGE_WIDTH</i>	Default: 576pt (8 in)	Sets the default page width.
<i>PAGE_HEIGHT</i>	Default: 792pt (11 in)	Sets the default page height.
<i>KERN</i>	true, false Default: true	Controls whether the formatter uses or ignores glyph kerning data to determine character positions.
<i>ENABLE_ACCESSIBILITY</i>	true, false Default: false	Controls whether the formatter uses a special mode to create accessible PDF documents.

Option	Possible Values	Description
<code>OMIT_FOOTER_AT_BREAK</code>	true, false Default: false	Defines whether tables footers are omitted at breaks by default.
<code>SPOT_COLOR_TRANSLATION_TABLE</code>	free text	<p>Path to Spotcolor-to-CMYK translation table file for use in <code>rgb-icc()</code> function with <code>#SpotColor</code> pseudo profile.</p> <p>The parameter can be specified either as a file name in the local file system, or as an URL. In addition to the common protocols, <code>data:</code> and <code>resource:</code> URL schemes are supported.</p> <p>Default: none, all spot colors come out black.</p> <p>Note: This table was hard-coded in previous versions of XEP. Due to license restrictions, it has been removed from the current version. Users are recommended to download this table from PANTONE® site and specify this option accordingly.</p>

5.2.3. Configuring Output Formats

XEP can render to several different output formats including PDF, PostScript and AFP. Certain properties of output documents can be controlled in two ways:

- **Processing Instructions** - The processing instructions are used to specify information that does not affect formatting and is safely ignored by the XSL-FO processors.

Each processing instruction begins with a prefix that identifies the output generator to which the instruction is addressed. For the standard PDF generator, the prefix is `<?xep-pdf-*>`, for PostScript, the prefix is `<?xep-postscript-*>`, and for AFP, the prefix is `<?xep-afp-*>`. Generators ignore processing instructions that do not start with their assigned prefixes. In particular, PDF generator instructions are invisible to the PostScript generator, and vice versa.

Instructions that pertain to an entire document should be placed at the top of the document, before or right after the `<fo:root>` start tag. Instructions that pertain to a single page of the documentation should be specified inside `<fo:simple-page-master>` object used to generate that page.

- **Generator Options** - Generator options affect the entire output document. Some features affect only parts of the input document and can only be expressed with processing instructions.

Generator Options can be used to set default settings for output generators. They are specified inside the `<options>` element in the configuration file. To distinguish them from

the core options, they are wrapped in the `<generator-options>` element. The following table describes the attribute of the `<generator-options>` tag:

Table 5.8. Generator-Options Attributes

Attribute	Possible Values	Description
<i>FORMAT</i>	PDF, PS, AFP	Format defines the target output format for the generator.

The following is an example of a fragment which turns on the linearization for the PDF generator and sets initial zoom factor to `fit-width` for both PostScript and PDF backends:

```
<generator-options format="PDF">
  <option name="LINEARIZE" value="true"/>
  <option name="INITIAL_ZOOM" value="fit-width"/>
</generator-options>

<generator-options format="PostScript">
  <option name="INITIAL_ZOOM" value="fit-width"/>
</generator-options>
```

All options can be controlled using processing instructions, and some options can be controlled by use of generator options. The following sections describe available processing instructions and generator options as well where they can be utilized.

Unicode Strings in Annotations (PDF, PostScript)

```
<?xep-pdf-unicode-annotations value?>
<?xep-postscript-unicode-annotations value?>
```

These processing instructions enable or disable the use of Unicode to represent PDF annotations strings, such as bookmark text and document info. In PostScript, the information is coded in `pdfmark` operators and used for further conversion to PDF.

The following are possible values:

- **true** - Enable use of 16-bit Unicode to represent annotation strings. In this mode, XEP uses 8-bit PDF Encoding for strings that can be represented in AdobeStandard character set and 16-bit Unicode for strings containing characters not included in AdobeStandard.
- **false** - Unicode is not used. Annotations are always represented in 8-bit PDF Encoding; characters not included in the AdobeStandard set are replaced by bullet symbols. This option may be used to enforce compatibility with older versions of PDF software that do not support Unicode, such as Adobe Acrobat 3.0.

Default: **true**

This feature can also be controlled by `UNICODE_ANNOTATIONS` option in the configuration file for PDF and PostScript generators.

Initial Zoom Factor (PDF, PostScript)

```
<?xep-pdf-initial-zoom value?>
<?xep-postscript-initial-zoom value?>
```

These processing instructions specify the magnification factor to be activated when the file is first opened in the PDF viewer. In PostScript, the information is coded in `pdfmark` operators and used for further conversion to PDF.

The following are possible values:

- **auto** - Page scaling is not specified.
- **fit** - The page is scaled to fit completely into the `view port`.
- **fit-width** - The page is scaled so that its width matches the width of the `view port`.
- **fit-height** - The page is scaled so that its height matches the height of the `view port`.
- **number or percentage** - The page is scaled by the number or percentage specified in the enabled box.

Default: **auto**

This feature can also be controlled by the `INITIAL_ZOOM` option in the configuration file for PDF and PostScript generators.

PDF Initial View (PDF, PostScript)

```
<?xep-pdf-view-mode value?>
<?xep-postscript-view-mode value?>
```

These processing instructions set the view mode to be activated in the PDF viewer when the PDF file is rendered and viewed. In PostScript, the information is coded in `pdfmark` operators and used for further conversion to PDF.

The following are possible values:

- **auto** - If there are bookmarks in the document, the bookmarks pane is displayed. Otherwise, all auxiliary panes are hidden.
- **show-none** - All auxiliary panes are hidden.
- **show-bookmarks** - The bookmarks pane is displayed.
- **show-thumbnails** - The thumbnails pane is displayed.

- **full-screen** - The document is displayed in full screen-mode.

Default: **auto**

This feature can also be controlled by the `VIEW_MODE` option in the configuration file for PDF and PostScript generators.

Logical Page Numbering (PDF)

```
<?xep-pdf-logical-page-numbering value?>
```

This processing instruction controls a page numbering scheme for the PDF document.

The following are possible values:

- **true** - Logical page numbers are written to the PDF file.
- **false** - Logical page numbers are ignored.

Default: **true**

Note: Adobe Acrobat has a special check box **Use logical page numbers**. To show logical page numbers of a PDF document, make sure this control is enabled.

This feature can also be controlled by the `LOGICAL_PAGE_NUMBERING` option in the configuration file for PDF generator.

Page Layout (PDF)

```
<?xep-pdf-page-layout value?>
```

This processing instruction controls initial page layout when a PDF document is open.

The following are possible values:

- **auto** - Uses settings of viewer application.
- **single-page** - Displays one page at a time.
- **continuous** - Displays pages continuously in one column.

Default: **auto**

This feature can also be controlled by the `PAGE_LAYOUT` option in the configuration file for PDF generator.

PDF Viewer Preferences (PDF)

```
<?xep-pdf-viewer-preferences value?>
```

This processing instruction controls viewer preferences for a PDF document.

The value is a comma or space separated list of keywords. Each one enables the respective viewer option. The following are supported keywords:

- **hide-toolbar** - Hides the viewer application's tool bars when the document is active.
- **hide-menubar** - Hides the viewer application's menu bar when the document is active.
- **hide-window-ui** - Hides user interface elements in the document's window (such as scroll bars and navigation controls), leaving only the document's contents displayed.
- **fit-window** - Resizes the document's window to fit the size of the first displayed page.
- **center-window** - Positions the document's window in the center of the screen.
- **display-document-title** - Controls whether the window's title bar displays the document title taken from the "title" entry of `<rx:meta-info>`. If absent, the title bar instead displays the name of the PDF file containing the document.

Default: **empty list**

This feature can also be controlled by the `VIEWER_PREFERENCES` option in the configuration file for PDF generator.

Treatment of Unused Destinations (PDF, PostScript)

```
<?xep-pdf-drop-unused-destinations value?>
<?xep-postscript-drop-unused-destinations value?>
```

These processing instructions specify whether named destinations are created for objects not referenced within the document. In PostScript, the information is coded in `pdfmark` operators and used for further conversion to PDF.

The following are possible values:

- **true** - Named destinations are created only for objects used as targets in `internal-destination` attributes.
- **false** - Named destinations are created for all objects that have an `id` attribute.

Default: **true**

This feature can also be controlled by the `DROP_UNUSED_DESTINATIONS` option in the configuration file for PDF and PostScript generators.

ICC Profile (PDF)

```
<?xep-pdf-icc-profile URL?>
```

These processing instructions specify a characterized printing condition. PDF/X and PDF/A-1 specifications require the presence of the characterized printing condition (`/OutputIntent`

entry in the PDF catalog dictionary). *URL* is the URI of the ICC file. It should follow the XSL-FO notation for uri-specification: `url()`.

PDF/X Support (PDF)

```
<?xep-pdf-pdf-x value?>
```

This processing instruction sets PDF/X compliance level.

The following are possible values:

- **none** - No PDF/X restrictions are applied.
- **pdf-x-1a** - Sets PDF/X-1a compliance level. The rendered PDF will comply with the PDF-X-1a:2001 spec.
- **pdf-x-3** - Sets PDF/X-3 compliance level. The rendered PDF will comply with the PDF-X-3:2001 spec.

Default: **none**

PDF/A Support (PDF)

```
<?xep-pdf-pdf-a value?>
```

This processing instruction sets PDF/A compliance level.

The following are possible values:

- **none** - No PDF/A restrictions are applied.
- **pdf-a-1a** - Sets PDF/A-1a compliance level. The rendered PDF will comply with level A of the PDF/A-1:2005 spec.
Note: PDF/A-1a compliant documents must be tagged. Set `ENABLE_ACCESSIBILITY` core option to **true**.
- **pdf-a-1b** - Sets PDF/A-1b compliance level. The rendered PDF will comply with level B of the PDF/A-1:2005 spec.

Default: **none**

Prepress Support (PDF, PostScript)

The following processing instructions define features that support the prepress production workflow.

```
<?xep-pdf-crop-offset value?>
<?xep-postscript-crop-offset value?>
```

These processing instructions specify offsets from the meaningful content on the page to the edges of the physical media (`/MediaBox` entry in the PDF page dictionary). Its *value* is a series of 1 to 4 length specifiers that set offsets from the edges of the page area (as specified in the XSL-FO input document) to the corresponding edges of the `/MediaBox`. Rules for expanding the value are the same as for the `padding` property in XSL-FO.

```
<?xep-pdf-bleed value?>
<?xep-postscript-bleed value?>
```

These processing instructions specify the bleeds — an extra space around the page area into which the contents of the page may protrude (`/BleedBox` entry in the PDF page dictionary). Its *value* is a series of 1 to 4 length specifiers that set offsets from the edges of the page area (as specified in the XSL-FO input document) to the corresponding edges of the `/BleedBox`. Rules for expanding the value are the same as for the `padding` property in XSL-FO.

If bleed values exceed the respective crop offsets, the latter are increased to make room for the bleeds.

```
<?xep-pdf-crop-mark-width value?>
<?xep-postscript-crop-mark-width value?>
```

These processing instructions display crop marks on the page. *value* defines line width for the marks; setting it to 0 disables drawing of crop marks.

```
<?xep-pdf-bleed-mark-width value?>
<?xep-postscript-bleed-mark-width value?>
```

These processing instructions display bleed marks on the page. *value* defines line width for the marks; setting it to 0 disables drawing of bleed marks.

```
<?xep-pdf-printer-mark URL?>
<?xep-postscript-printer-mark URL?>
```

These processing instructions specify additional SVG images to be drawn in the offset area surrounding the page (specified by `crop-offset` and `bleed` parameters). Printer marks are clipped to the outside of the bleed rectangle. This facility can be used to create registration targets and color bars; the respective sample SVG images are enclosed in XEP distribution. *URL* is the URL to the location of the SVG file. It should follow the XSL-FO notation for uri-specification: `url()`.

PDF Version (PDF)

```
<?xep-pdf-pdf-version value?>
```

This processing instruction sets target PDF version.

The following are possible values:

- **1.3**

- **1.4**
- **1.5**
- **any higher version** is allowed here, since PDF versions are backward compatible.

Default: **1.4**

Note: When set to 1.3, advanced features of PDF 1.4 are disabled.

This feature can also be controlled by `PDF_VERSION` option in the configuration file for the PDF generator.

Compression of PDF Streams (PDF)

```
<?xep-pdf-compress value?>
```

This processing instruction controls compression of content streams in PDF.

The following are possible values:

- **true** - PDF streams are compressed using the Flate algorithm.
- **false** - PDF streams are not compressed. This option is useful for debugging.

Default: **true**

This feature can also be controlled by the `COMPRESS` option in the configuration file for the PDF generator.

Linearization (PDF)

```
<?xep-pdf-linearize value?>
```

This processing instruction controls linearization (also known as Web optimization) of the PDF output.

The following are possible values:

- **true** - PDF is linearized. This options is used to prepare documents for HTML output.
- **false** - PDF is not linearized.

Default: **false**

This feature can also be controlled by the `LINEARIZE` option in the configuration file for the PDF generator.

Document Security (PDF)

The following processing instructions control PDF security settings.

```
<?xep-pdf-ownerpassword value?>
```

This processing instruction sets an owner password for the PDF document to **value**. Owner password gives its holder full control over the PDF document. This unlimited access includes the ability to change the document's passwords and access privileges.

Note: Adobe Acrobat by default applies user's access restrictions to owners too. To remove some of these restrictions, go to 'Document Properties -> Security' and choose 'Change Settings' option.

```
<?xep-pdf-userpassword value?>
```

This processing instruction sets a user password for the PDF document to **value**. Holders of user password are subject to access restrictions; only operations included in the privilege list are authorized.

```
<?xep-pdf-userprivileges value?>
```

Sets the default privilege list for users accessing the rendered document with user password. The value must be a sequence composed of the following tokens:

- **print** - Enables printing the document.
- **modify** - Enables editing the document.
- **copy** - Enables copying text and images from the document to the clipboard.
- **annotate** - Enables adding notations to the document and changing the field values.

Tokens can be specified in any order, separated by commas and/or spaces.

Note: If neither user password nor owner password is set, security is disabled and the rendered PDF is not encrypted.

If the user password is set and the owner password is not set, then the latter is set equal to the former. This enables password protection on the PDF file, but gives password holder full control over the document: no distinction is made between user and owner.

If the owner password is set and the user password is not set, the rendered PDF document can be viewed by anyone without entering a password. However, operations on this file will be restricted to privileges specified in the user privilege list; other operations will require authentication with the owner password.

Default: **Security disabled** (neither of the passwords are set). Default privilege list is **annotate**.

These features can also be controlled by the `USERPASSWORD`, `OWNERPASSWORD`, and `USERPRIVILEGES` options in the configuration file for the PDF generator.

Note: Setting passwords through a configuration file poses obvious security risks, and is not recommended. Use processing instructions to enable file protection.

PostScript Language Level (PostScript)

```
<?xep-postscript-language-level value?>
```

This processing instruction sets target PostScript language level.

The following are possible values:

- **2**
- **3**

Note: When the language level is set to 2, some advanced features and font flavours are not available.

Default: **3**

This feature can also be controlled by the `LANGUAGE_LEVEL` option in the configuration file for the PostScript generator.

EPS Graphics Treatment (PostScript)

```
<?xep-postscript-clone-eps value?>
```

This processing instruction controls whether EPS graphics are included in the PostScript output using forms mechanism, or by pasting their contents at each occurrence.

The following are possible values:

- **true** - EPS graphics are pasted into the output stream at each occurrence. This may lead to a substantial growth of the resulting file size.
- **false** - EPS graphics are in PostScript form. This minimizes the file size, however, some EPS images cannot be processed this way and it may corrupt the PostScript code.

Default: **true**

This feature can also be controlled by `CLONE_EPS` option in the configuration file for the PostScript generator.

Page Device Control (PostScript)

```
<?xep-postscript-page-device entryname entryvalue?>
```

This processing instruction sets a single entry *entryname* in the page device dictionary to value *entryvalue*. Entry name must be a valid PostScript name (with or without leading slash). The value is specified as an arbitrary PostScript expression. Entry name and value must be separated by whitespace. There can be more than one such instruction, each setting its entry.

Warning

XEP does not check the spelling of either the entry name or the value supplied in this instruction. Wrong code passed with this option can invalidate the whole output file.

To set page device options for the whole document, the respective instructions should appear at the top of the document, before the `<fo:root>` element. Such entries are set in the document setup section and cleaned up in the document trailer.

To control page device settings for a single page, the instructions should be specified inside the `<fo:simple-page-master>` object used to generate the page. In this case, page setup parameters are modified in the page setup section and reset in the page trailer.

Page Labeling (PostScript)

```
<?xep-postscript-page-label value?>
```

This processing instruction changes the **label** argument of **%%Page** PostScript command. This PI should be inserted to `fo:simple-page-master` element.

The following are possible values:

- **value** - Any text. The text may contain an optional token **%d** that will be automatically replaced with incrementing integer values, starting with **1**.

Note: Any time the document page contains `xep-postscript-custom-comment` Processing Instruction with value different to the previous one, the incrementing counter will be automatically reset to **1**.

Default: **blank**

Inserting Custom Comments (PostScript)

```
<?xep-postscript-custom-comment value?>
```

This processing instruction allows inserting custom comments into PostScript document.

The following are possible values:

- **value** - Any valid PostScript comment.

Note: If the PI is inserted into `fo:root` element or before it, the value is placed in the document header, before **%%EndComments**. If the PI is inserted into `fo:simple-page-master` element, the value is placed in every page which uses this `fo:simple-page-master` as a template, after **%%EndPageSetup** comment. If the PI is inserted into `fo:page-sequence` element, the value will be placed for each page of the sequence, after **%%EndPageSetup** comment. The value will be validated before inserting to document,

all "%" symbols will be removed, the first symbol will be capitalized and the value will be prepended with one (for page level comments) or two (for document level comments) "%" symbols.

Default: **no comment**.

Images Treatment in XML Output (XML)

```
<?xep-out-embed-images value?>
```

This processing instruction controls whether the XML output generator embeds external images referenced in the document in the resulting document instance as Base64 strings.

The following are possible values:

- **true** - All images are stored inside the resulting file using the `data:` URL scheme.
- **false** - Images are not embedded. In the generated XML file, images are referenced by their original URLs.

Default: **false**

This feature can also be controlled by the `EMBED_IMAGES` option in the configuration file for the XML output generator.

5.2.4. Configuring Fonts

Fonts can be configured inside the `` element. It contains descriptors for font families, font groups, and font aliases. The formatter uses them to map XSL-FO font properties to actual fonts.

Fonts and Font Families

Fonts are categorized into families, which is the basic configuration unit in XEP, and then further into groups. A font family is a set of fonts that share a common design but differ in stylistic attributes, such as upright or italic, light or bold. All data pertinent to one font family is contained inside a `<font-family>` element.

The `<font-family>` element contains the attribute described in the following table:

Table 5.9. Font-Family Attributes

Attribute	Possible Values	Description
name	free text Note: Family names must be unique within the configuration file. They are matched against the respective XSL-FO property value.	Identifies the font family.

When no font family is specified in the input file, the default is defined by `default-family` attribute of the `` element. Its value is a family name that must be present in the file, otherwise a configuration error occurs.

The following is an example of a font family descriptor:

```
<font-family name="Courier">
  <font>
    <font-data afm="Courier.afm"/>
  </font>
  <font style="oblique">
    <font-data afm="Courier-Oblique.afm"/>
  </font>
  <font weight="bold">
    <font-data afm="Courier-Bold.afm"/>
  </font>
  <font weight="bold" style="oblique">
    <font-data afm="Courier-BoldOblique.afm"/>
  </font>
</font-family>
```

Inside the family descriptor, there are one or more entries for individual fonts that belong to the family. A font entry is specified by a `` element. It has attributes to specify features of the font within the family, such as `weight`, `style`, and `variant`. For a font to be selected by a formatter, these attributes should match `font-weight`, `font-style`, and `font-variant` specified in the XSL-FO document.

Embedding and Subsetting Fonts

Most fonts can be either **embedded** into the resulting PDF or PostScript document or specified as fonts external to the file. If the font is external, the rendered file can only be viewed on systems that have the font configured for use with viewing or printing the application. Typically, all fonts are embedded except for 14 standard Adobe PDF fonts. For some applications, embedding basic fonts may also be required. Embedding of a font is controlled by the `embed` attribute of the `` element describing the font.

An embedded font can be **subsetting**, which means that instead of storing the entire font in the document, XEP leaves only those glyphs that are actually used in the text. This option reduces the document size but makes it unsuitable for subsequent editing. Subsetting is governed by the `subset` attribute of the `` element.

To provide a more compact notation, the `embed` and `subset` properties are **inheritable** down the configuration tree: when specified on a node in the configuration file, they affect all `` descendants of that node. For example, `embed/subset` attributes specified in `<font-family>` will affect all fonts in that family; placing them on `<font-group>` will set the respective para-

meters for all fonts in all families in the group (unless overridden on some descendant node), etc.

XEP does not support embedding and subsetting of native AFP fonts in AFP documents so far.

Note: TrueType and OpenType fonts may contain internal flags that prohibit their embedding or subsetting. XEP honors these flags and may refuse to embed or subset your font if the respective action is not authorized by the flags inside it.

AFP Fonts

To use an AFP font with XEP, it is necessary to obtain AFP font files containing Codepage and Charsets. An URL location to the Codepage file should be specified in the `codepage-file` attribute of `<font-family>` element and attribute `codepage-name` should contain the name of corresponding Codepage. Font encoding can be specified in `encoding` attribute of `<font-family>` element (default value is Cp500).

The size (for raster AFP fonts) should be specified in the `size` attribute of the `` element. URL to Charset file should be specified in `charset-file` attribute of `<font-data>` element and attribute `charset-name` should contain the name of Charset respectively.

Example: suppose we have a raster AFP font with Codepage file `T1ED0500.CDP` and Charset file `C0V08000.CHS` containing metrics for characters (size 10, italic). Its descriptor in the configuration file can look like this:

```
<font-family name="AfpFont"
  codepage-name="T1ED0500"
  codepage-file="T1ED0500.CDP"
  encoding="Cp1146">
  <font size="10" style="italic">
    <font-data charset-name="C0V08000" charset-file="C0V08000.CHS"/>
  </font>
  ...
</font-family>
```

Algorithmic Slanting

Algorithmic slanting can be applied to fonts in order to produce oblique or backslanted versions of fonts that do not have separate outlines for these styles. This is done by placing a `<transform>` element inside the `` descriptor. The slant angle is specified in the `slant-angle` attribute on the `<transform>` node. Its value sets the angle in degrees. Positive angles slant the text clockwise, producing oblique versions; negative ones rotate it counterclockwise, producing backslanted font styles.

XEP does not support algorithmic slanting of AFP fonts so far.

If a font family contains no entry for `oblique` or `italic` font style, the oblique font is produced algorithmically by applying a default slanting of 12°. Similarly, a missing backslant font is synthesized from the nearest upright version, slanting it by -12°.

Ligaturization

Fonts can be instructed to contract certain sequences of characters into ligatures. A set of ligature characters is specified in the `ligatures` attribute of the `` element, as a space- or comma-separated list of ligature characters. The characters must be Unicode ligature codepoints.

Note: In XEP, ligaturization support is basic: only ligatures registered in Unicode can be used. Moreover, ligaturization does not work for characters that undergo contextual shaping: this excludes all Arabic ligatures from consideration. Further versions of XEP are expected to improve ligaturization support.

Initial Encoding

Type 1 fonts may have different encoding tables. (Encoding table is an essential part of a Type 1 font and matches character codes to glyph names). According to PDF Spec, there are 3 predefined encodings: **WinAnsi**, **MacRoman**, and **MacExpert**. There is also the built-in font's encoding. All other encodings are treated as custom ones.

In Adobe Acrobat it is possible to see each Type 1 font encoding used in a document (**Document Properties** panel -> **Fonts** tab -> **Encoding** field for each Type 1 font). The value of this field may be one of:

- **Standard** - The font's built-in encoding
- **Ansi** - Windows Code Page 1252 (Windows ANSI)
- **Roman** - Mac OS standard encoding for Latin text in Western writing systems
- **Expert** - An encoding for use with expert fonts
- **Custom** - A custom encoding

The same values (but 'Custom') may be used for `initial-encoding`.

To provide a more compact notation, the `initial-encoding` is **inheritable** down the configuration tree: when specified on a node in the configuration file, it affects all `` descendants of that node. For example, `initial-encoding` attribute specified on `<font-family>` will affect all fonts in that family; placing it on `<font-group>` will set the respective parameter for all fonts in all families in the group (unless overridden on some descendant node), etc.

Note: This attribute only affects the first encoding table for a Type 1 font it is specified on. If the document contains glyphs (from this font) that do not belong to the specified first encoding table, XEP will add more encoding tables which will all be treated as **Custom**.

Font Groups

Several font families can be wrapped into a `<font-group>` container element. Groups can be nested, forming complex font hierarchies. This element does not affect font mapping and serves only for logical grouping of font families. In particular, it is often convenient to use it as a host for the `xml:base` property, to specify a common base directory for a group of font families that form a package. Another suggested use of `<font-group>` is for remoting: contents of the font group can be placed into a separate file and reused across multiple font configurations.

The only attribute specific to `<font-group>` is `label`, which assigns a name to the group. The name serves only for record keeping, no constraints are imposed on it.

Font Aliases

XEP uses **font aliases** to provide alternate names for font families and group several families into one “logical” family. A font alias is defined by a `<font-alias>` element. The element has two attributes, both required: `name` is the name of the “logical” font family, and `value` is a comma-separated list of font family names to which it should resolve. The list may contain a single font family; in this case, the alias merely provides an alternate name for it.

Note: Aliases always resolve to “real” families and not to the other aliases. Chained alias resolution is not possible in XEP.

5.2.5. Configuring Languages

Language-specific configuration parameters are stored in the third major section of the configuration file, inside a `<languages>` element. The `<languages>` element contains one or more `<language>` elements, and each `<language>` element stores information pertaining to a single language. The language is identified by two attributes:

- **name** - The name of the language.
- **code** - A list of codes used to refer to the language in the XSL-FO input data. Multiple codes are separated by spaces.

In XEP two kinds of data are configurable in this section of the configuration files:

- Hyphenation patterns
- Language-specific font aliases

Configuring Hyphenation

XEP uses T_EX hyphenation patterns for hyphenation data. Details on hyphenation algorithm are described in [Appendix B, Linguistic Algorithms](#).

A hyphenation pattern file is associated with a language by placing a `<hyphenation>` element into the language section in the configuration file. Its `pattern` attribute specifies the URL to the T_EX pattern file. An optional `encoding` attribute specifies the encoding of the pattern file; if it is missing, ISO-8859-1 is assumed.

Language-Specific Font Aliases

Language sections may also contain `<font-alias>` elements, described above in [Font Aliases](#). These aliases are activated when the language is selected in the input XSL-FO document; they take precedence over aliases specified in the `<fonts>` section of the configuration file and may mask them.

5.3. Resolution of External Entities and URIs

XEP can be configured to use a specific entity resolver for all SAX parsing calls inside it. The resolver class is specified by a Java system property `com.renderx.sax.entityresolver`. It must have a public constructor with no arguments, and implement `org.xml.sax.EntityResolver` interface.

Similarly, XEP can assign a user-defined class to resolve URIs in calls to `document()` function, `<xsl:import>`, and `<xsl:include>` XSLT directives. The class name is specified in `com.renderx.jaxp.uriresolver` system property; it must provide a public default constructor, and implement `javax.xml.transform.URIResolver` interface.

The principal use of these features is to add support for XML catalogs to XEP, to avoid repeated loading of common DTDs and stylesheets from the internet. For example, the following setting configures XEP to use XML entity and URI resolver from Apache project (provided that you have included resolver classes in the classpath, and properly configured it):

```
java
-Dcom.renderx.sax.entityresolver=org.apache.xml.resolver.tools.CatalogResolver
-Dcom.renderx.jaxp.uriresolver=org.apache.xml.resolver.tools.CatalogResolver
...
```

XML catalogs resolver is included into xml-commons tools available as a part of Apache project. For further information about catalogs and entity resolution, and for resolver download please proceed to Apache website: <http://xml.apache.org/commons/components/resolver/index.html>.

Chapter 6. XEP AFP Generator

6.1. Generating AFP Documents

AFP documents can be generated through the following:

- XEP Assistant - When formatting the XML file using the File Assistant, select AFP as the format, as described in [Chapter 3, XEP Assistant](#).
- Command Line - Using the command line, AFP documents as well as AFP resource files can be generated.

- To generate an AFP document, use the parameter `-afp`:

```
-afp <afp document file name>
```

For more information, please refer to [Chapter 4, Using the Command Line](#).

- To generate an AFP resource file, use the parameter `-DH4AFP.RESOURCE`:

```
-DH4AFP.RESOURCE=<afp resource file name>
```

Note: Since `-DH4AFP.RESOURCE` is a generator option parameter, it must precede all other parameters like `-xml`, `-xsl`, `-fo`, `-xep`, `-pdf`, `-ps`, `-afp`.

Alternatively, you can use the configuration file variable. For more details, refer to [Section 6.8, "Configuring the XEP AFP Generator"](#).

6.2. Fonts

Non-CID OTF fonts are currently supported which allows for higher AFP standard conformance. Fonts larger than 36 pt can be processed as well, which produces better AFP documents.

6.2.1. Font Mapping

XEP supports two different ways using fonts in AFP generator:

- The first variant is based on native Non-CID Open Type Fonts (OTF) that correspond F:OCA specifications. It is described in section [Section C.1.4, "Supported AFP Fonts"](#). This variant requires a set of native AFP font files and allows using native AFP fonts metrics.

You may also find useful information on [AFP Fonts](#).

- Another variant of configuration lets to map AFP native fonts to non-AFP fonts supported by XEP to obtain font metrics. In this case, the metrics of TrueType/OpenType fonts are

used for formatting; after that, when generating AFPDS stream, XEP uses mapped font values to refer in result document.

Therefore, AFP generator for XEP supports all kinds of fonts supported by XEP.

Mapping FO fonts and native AFP fonts can be configured in the XEP configuration file, in the AFP generator configuration section. Please refer to [Section 5.1.4, “Configuring Fonts”](#) for details.

6.3. Images

6.3.1. Image Support

Raster image formats are not supported by AFP, and color images are rarely supported by AFP printing systems. Therefore, depending on the configuration options, all raster images are either converted to bi-level images or grayscale images.

Image	Properties
Bi-level images	1 bit per pixel, compressed
Grayscale images	8 bits per pixel, uncompressed

Note: In AFP, bi-level images are mixed with their background. Therefore, white points appear transparent.

6.3.2. Referencing Images

Images can be included once, and referenced multiple times. This allows for reduced output size and improved performance. This feature is very useful for repeating images, like corporate logos, headers, footers, etc.

The image can be included in the main AFP output file, or in a separate resource file as described in [Section 6.8.5, “Other Configuration Options”](#).

6.3.3. Image Clipping

If a source document refers to an image, and the image is bigger than the containing block, normally it should be clipped. AFP Backend correctly clips only those images having resolution equal to (or higher than) the AFP document's resolution. Otherwise, the entire image appears, probably overlapping with other page elements positioned to the right or below the image container.

6.4. Highlight Color Support

Highlight color is a special case of color encoding, when a solid colorant used (contrary to other schemes like RGB, CMYK, etc). In XSL-FO, this kind of encoding is called Spot Color. XEP AFP Backend treats spot color in source document and produces AFP Highlight Color instructions within the MO:DCA-P stream.

Highlight Color has the following major attributes:

- Colorant name - usually, includes colorant vendor name and catalogue ID of the color. For example, "**PANTONE Orange 021 M**".
- Tint - percentage of colorant covering target area. AFP printers are capable to cover certain percentage of target area, making the color opaque.

Each value must be given either in percents (from 0% to 100%) or as a number in the interval from 0 to 1.

Note: When the Highlight Color space is specified in a target repeating group, the percent coverage parameter is normally supported only for areas such as object areas and graphic fill areas. For other data types this parameter is normally simulated with 100% coverage.

- Shading - besides tint, AFP devices are capable to cover certain percentage with main color (usually Black). This attribute defines which percentage will be covered with main color.
- Alt (alternative) color - used in XSL-FO to define most-close analogue to Spot Color. This can be either CMYK, RGB, or Grayscale value.

AFP Generator uses the following algorithm of Spot Colors identification:

- AFP Backend for XEP finds spot-color in source document. It looks up the [configuration file](#) for Highlight Color Index defined within.
- If Colorant ID found, AFP Backend uses the ID associated.
- Otherwise (Colorant ID not found), AFP Backend registers the Highlight Color within the range of Custom Colors defined in MO:DCA (ID 0x0100-0xFF00).
- Next time spot-color with same colorant used within the same document, it will obtain the same ID. Automatically obtained ID's are not saved for further use after the operation is completed.

6.5. Graphics Support

Scalable Vector Graphics (SVG) is a language for describing two-dimensional vector graphics in XML.

AFP Backend 4.13 has limited support of SVG primitives. They are rendered as instructions listed within G:OCA command set.

The following G:OCA objects are currently supported:

- Lines (**svg:line**).

Lines are considered to be strokes of a pen that draws on the canvas.

The size, color, and style of the pen stroke are part of the line's presentation. These characteristics are in the style attribute.

Currently "stroke" (color of line) and "stroke-width" (width of line) characteristics of Style attribute are supported.

For example:

```
<svg:line x1="20" x2="20" y1="62" y2="8"
          style="stroke-width:6; stroke: blue;" />
```

- Rectangles.

The interior of the rectangle can be filled with the specified fill color.

If a fill color was not specified, the interior of the shape will be filled with white.

The outline of the rectangle is drawn with strokes, whose characteristics can be specified by the same way as for lines.

"fill" (fill color), "stroke" (outline color) and "stroke-width" (width of outline) attributes are supported.

If the fill color specified as "none", then only outline of the rectangle will be drawn with color specified in "stroke" attribute.

For example:

```
<rect x="60" y="60" width="80" height="70"
      fill="none" stroke="yellow" stroke-width="5" />
```

- Paths.

Paths represent the geometry of the outline of an object, defined in terms of <moveto> (set a new current point), lineto (draw a straight line), curveto (draw a curve using a cubic Bézier), arc (elliptical or circular arc), and closepath (close the current shape by drawing a line to the last moveto) elements.

Full implementation of Path processing has been made.

Supported commands:

- The "moveto" command.

The "moveto" commands (M or m) establish a new current point.

- The "closepath" command.

The "closepath" (Z or z) ends the current subpath and causes an automatic straight line to be drawn from the current point to the initial point of the current subpath.

- The "lineto" commands.

The various "lineto" commands (L, l, H, h, V, v) draw straight lines from the current point to a new point.

- Cubic Bézier curves (C, c, S and s).

A cubic Bézier segment is defined by a start point, an end point, and two control points.

Also includes filled areas with border in form of Bézier curve.

- Quadratic Bézier curves (Q, q, T and t).

A quadratic Bézier segment is defined by a start point, an end point, and one control point.

Also includes filled areas with border in form of Bézier curve.

- Elliptical arcs (A and a).

An elliptical arc segment draws a segment of an ellipse. Various kinds of elliptical arcs are supported.

Note: Internally, XEP transforms all arcs to Bézier curves.

Also includes filled areas with border in form of arc.

Filled areas may be formed by any combination of lines, arcs, and Bézier curves. AFP Generator correctly processes such case of *svg:path* and makes areas filled correctly.

Coordinate system transformations are also supported.

A new user space can be established by specifying transformations in the form of a "transform" attribute on a container element or graphics element or "viewBox" attribute on an "svg", "symbol", "marker", "pattern" and the "view" element.

The "transform" and "viewBox" attributes transform user space coordinates and lengths on sibling attributes on the given element and all of its descendants.

The following logic elements are currently supported:

- Groups (*svg:g*).

- Rotation of SVG block.

```
<fo:block-container reference-orientation="270">
```

Possible values are: "0", "90", "180", and "270".

- Nested SVG (*svg:svg*) and Viewbox (*svg:viewbox*) are supported.

Partial implementation of "transform" and "viewBox" attributes processing has been made.

Transformations can be nested, in which case the effect of the transformations are cumulative.

- SVG text.

XEP has limited support of SVG text. Only raster fonts are supported.

If SVG text is enclosed into viewBox, the actual font size is calculated accordingly.

In the following sample, SVG block is stretched 2 times by X and Y axis, and font of size 10 is used.

```
<svg:svg width="400" height="100" viewBox="0 0 200 50">
  <svg:text style="font-family:Helvetica; font-size:10; font-weight:bold"
    text-anchor="middle" x="100" y="25">
    Sample
  </svg:text>
</svg:svg>
```

The actual font size will be equal to 20.

If viewBox zoom factors by X and Y axis are different, root-mean-square value for font size is used.

In the following sample, SVG block is stretched 3 times by X axis and remain the same by Y axis.

```
<svg:svg width="600" height="50" viewBox="0 0 200 50">
  <svg:text style="font-family:Helvetica; font-size:10; font-weight:bold"
    text-anchor="middle" x="100" y="25">
    Sample
  </svg:text>
</svg:svg>
```

The actual font size will be calculated as $\text{original_size} * \text{square_root}((\text{zoomX}^2 + \text{zoomY}^2)/2)$ and equal to 22.

For SVG text, only the following rotation values are possible: "0", "90", "180", and "270". Nested rotations of svg blocks and viewBox are supported.

- Markers.

The "marker" element defines the graphics that is to be used for drawing arrowheads or polymarkers on a given "path", "line", "polyline" or "polygon" element.

Not supported or limited support:

- Gradient fill for rectangles is not supported: solid fill color used.
- Other primitives not listed above are not supported by current version.

6.6. Barcodes Support

A barcode is a machine-readable representation of information in a visual format.

Most types of barcodes stores data in the width and spacings of printed parallel lines.

Barcodes are simple to represent as black rectangles separated by white spaces, but they have proved to be difficult to generate accurately. Bar and space widths are often computed in a complex manner and checking digits additionally complicates the process.

AFP Backend uses XSLT stylesheets to implement the computational part and SVG to draw the image and text. These stylesheets are available for free download from RenderX site:

<http://www.renderx.com/demos/barcodes.html>

As soon as XSL-FO 1.0 does not have native support of barcodes, technical implementation of them is based on SVG. These stylesheets render barcodes as SVG primitives, including additional `<desc>` tag containing the value and parameters of barcode rendered.

AFP Backend processes this tag and decides whether it is barcode or another SVG graphic.

If barcode tag is found, the barcode is rendered with BC:OCA or G:OCA, depending on which is enabled. This is done with `USE_BCOCA_LEVEL` and `USE_GOCA_LEVEL` configuration parameters.

If both BC:OCA and G:OCA are enabled, BC:OCA overrides. Please refer [configuration parameters](#) section about configuring BC:OCA and G:OCA.

Please refer [Section 6.5, "Graphics Support"](#) section for more information about G:OCA implementation.

Note: AFP Barcodes (BC:OCA) are verified to be compatible with barcodes generated by WordML2FO stylesheets (the latest version). RenderX WordML2FO stylesheets are available for free download from RenderX site:

<http://www.renderx.com/tools/word2fo.html>

Currently, the following types of Barcodes are supported by AFP Backend:

- EAN-13

- EAN-8
- UPC-A
- Codabar
- Code2of5
- Code3of9 **(with limitations)**
- Code128 **(with limitations)**
- 4state-AU **(with limitations)**

Please refer [Section 6.11, “Limitations of the XEP AFP Generator”](#) for more details.

For example:

```
<svg:svg xmlns:svg="http://www.w3.org/2000/svg"
  width="34.98mm" height="27.39mm">
  <desc xmlns:mydoc="http://example.org/mydoc">
    <barcode value="9780444505156" type="EAN-13"/>
  </desc>
  <svg:rect y="0" height="24.915mm" x="3.63mm"
    width="0.33mm" style="fill: black;"/>
  <svg:rect y="0" height="24.915mm" x="4.29mm"
    width="0.33mm" style="fill: black;"/>
  ...
  <svg:rect y="0" height="23.1mm" x="4.95mm"
    width="0.99mm" style="fill: black;"/>
</svg:svg>
```

`<desc/>` tag must have single child node `<barcode/>`. The following attributes are available for `<barcode/>` tag:

- `value` contains string value of the barcode
- `type` one of the values listed in supported types:
 - EAN-13
 - EAN-8
 - UPC-A
 - Codabar
 - Code2of5
 - Code3of9

- Code128
- 4state-AU

Note: If you develop custom stylesheets implementing Barcodes, please note that barcode MUST be alone item within SVG block if barcodes are generated with BC:OCA. That is, AFP Generator skips SVG content after `<desc>` tag.

6.7. FORMDEF Resource

6.7.1. What is a FORMDEF Resource?

FORMDEF is the AFP 'forms definition.' It defines the parameters of the physical page environment. The parameters including the following:

- Paper size and rotation.
- Simplex or duplex printing mode.
- Printing several logical document pages on the same sheet.
- Number of copies.
- Paper cutting, punching, etc.
- Paper source selection.
- Overlays for physical and logical pages.
- Finishing documents.

FORMDEF can be attached to a single document, or to multiple documents. It is contained in the resource section of a printfile.

Note: The XEP AFP Generator has limited support for generation of FORMDEF resources. In general, complex features like overlays and page segments are not supported.

6.7.2. Generating a Document with FORMDEF Resource

FORMDEF is a MO:DCA resource. Therefore, two steps are required in order to generate a document with FORMDEF.

1. You must specify the resource file on the XEP command line.
2. You must concatenate the resource file and the document file or files, in order to form a print file.

Note: If a resource file is not specified, the FORMDEF instructions are processed but not generated in the resulting AFP file.

6.7.3. FORMDEF Processing Instructions

FORMDEF resource is described in the source document as a set of special processing instructions. These instructions may appear at the top of the XSL-FO document before or after the **<fo:root>** element, or within page masters. Each processing instruction has a pseudo-element which is commonly used in XML similar to an **<?xml ?>** instruction.

FORMDEF processing instructions are divided into two main groups:

- Non-repeating instructions - They may only have one instance within a document.

The following are non-repeating instructions:

- `xep-afp-form-definition`
- Repeating instructions - They may appear in a document multiple times.

The following are repeating instructions:

- `xep-afp-page-definition`
- `xep-afp-copy-group`

Repeating instructions contain a pseudo-element "id," which uniquely identifies each instance of an instruction of the same type within a document. When XEP processes repeating instructions, only the first instance of an instruction of the id is processed. All other instructions with the same id are ignored. If an instruction is inside a page master, it is reproduced on each page generated by this page master, but XEP processes only the first occurrence of this instruction, as instruction ids are same.

FORMDEF Syntax

The syntax and semantics of FORMDEF processing instructions are as follows:

- **xep-afp-form-definition** - Defines the FORMDEF resource.

Format:

```
<?xep-afp-form-definition  
  sheet-height="size-value" sheet-width="size-value"?>
```

Attributes:

- **sheet-height** - Defines the sheet height. Values may contain simple size expressions such as, "3in" or "10pt."
- **sheet-width** - Defines the sheet width. Values may contain simple size expressions such as, "3in" or "10pt."

Both **sheet-height** and **sheet-width** are mandatory attributes.

- **xep-afp-page-definition** - Defines a logical page on a sheet of paper or other medium.

Format:

```
<?xep-afp-page-definition id="id-number" x0=" size-value" y0="size-value"
orientation="ori-value" type="type-value"?>
```

Attributes:

- **id** - Instruction identifier. The value must be a decimal number.
- **x0, y0** - The coordinates of the logical page presentation on a physical sheet. Values may contain simple size expressions.
- **orientation** - The orientation of the logical page on a physical sheet. Possible values are 0, 90, 180, or 270.
- **type** - The type of logical page which defines the common page layout. Possible values are the following:

default-front-page, default-back-page, p1-front-page, p1-back-page, p2-front-page, p2-back-page, p3-front-page, p3-back-page, p4-front-page, p4-back-page, default-front, default-back, p1-front, p1-back, p2-front, p2-back, p3-front, p3-back, p4-front, p4-back

- **xep-afp-copy-group** - Defines a copy group as well as its attributes and keywords.

Format:

```
<?xep-afp-copy-group id="id-number" copy-count="number"
mode="mode-val" [key="value"]* ?>
```

Attributes:

- **id** - Instruction identifier. The value must be a decimal number.
- **copy-count** - The number of copies in the copy group. The value must be a decimal number.
- **mode** - Printing mode. Possible values are simplex, duplex, and tumble-duplex.

id, **copy-count** and **bold** are mandatory attributes.

xep-afp-copy-group may contain several optional pairs of keywords and values. Each keyword-value pair must conform to the following rules:

Format of key-value pair:

```
key-value-pair = key, '=', "'", value, ''';
```

```

key = ( 'x' | 'X' ) ,
      ( '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7'
        | '8' | '9' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' ) ,
      ( '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7'
        | '8' | '9' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' );

value = ( '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | 'A' | 'B' | 'C' | 'D'
          | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7'
          | '8' | '9' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' );

```

Examples of valid keys: "x12", "XFF", "xA1".

Examples of valid values: "AF", "15", "FD".

Each key-value pair adds a keyword with value to MO:DCA MMC structured field content.

Note: Printing mode such as simplex, duplex or tumble duplex is controlled separately by the **mode** attribute which effectively generates XF4 MMC keyword. If the mode of printing is duplex, the copy group is generated twice in MCC automatically, and you do not need to repeat the group twice. You need not use XF4 MMC keyword explicitly.

6.8. Configuring the XEP AFP Generator

Configuration of the AFP generator is performed in a usual way all XEP generators are configured. All configuration options for AFP generator are child elements of XEP configuration file element `<generator-options format="AFP">`. Each AFP generator configuration option is an element `option` and looks like

```
<option name="OPTION_NAME" value="OPTION_VALUE"/>.
```

Note: AFP parameters can be set in three different ways, depending on your specific needs:

- **Configuration file** - in this case, the parameter value applies to all documents processed with this configuration file.
- **Environment variable (Generator option)** - passed within command line and applies for current run of XEP. **Generator option value overrides Configuration file values.**
- **Processing Instruction** - passed within Processing Instruction (PI) inside XSL:FO document. Please refer [Output Format Settings](#) section for more details on processing instructions. **Processing Instruction value overrides Generator option and Configuration file values.**

In this section, all parameters will be described in **Configuration file** format.

AFP generator's prefix for **Generator options** is H4AFP. So, `RESOLUTION` parameter will look like this:

```
-DH4AFP.RESOLUTION=1440
```

AFP generator's prefix for **Processing instructions** is `xep-afp-`. So, `RESOLUTION` parameter will look like this:

```
<?xep-afp-resolution 1440?>
```

Processing Instruction may appear at document level or page level. Every time the page level parameter is set, it applies until the same parameter is set to another value. For example, if `RESOLUTION` option is set to: 1440 in config file; to 720 for page #5; and to 1440 for page #10 - the value of 720 will apply for **ALL pages ##5 till 9**.

6.8.1. Configuring Character Sets

AFP generator for XEP has extended support for international Character sets. The following chapter describes how to configure the necessary character sets and corresponding Codepages.

One should configure only those Character sets and Codepages supported by the fonts residing in target AFP device.

Note: Even if the printer is configured to support various Codepages, it is still reasonable to remove from XEP configuration file these Character sets not used in the documents to improve performance.

The element `<charsets>` is a container for Character sets configuration.

Each Character set is configured within `<charset>` element. It has the attribute `name` describing the name of Character set. The attribute `name` must be unique.

Character set definition contains the following child elements:

- List of Unicode character ranges applying the Character set, represented `<code-ranges>` element;
- List of specially mapped characters, represented by `<character-mapping>` element;
- Single `<codepage>` element defining the Unicode to Codepage mapping;

Unicode character range is defined by `<code-range>` element. It has the following attributes:

- `from` and `to` defining beginning and ending values of Unicode characters belonging to the character range; Must be hexadecimal value; **Required**;

Codepages are defined by `<codepage>` element. It has the following attributes:

- `name` string value defining Java standard name of Codepage; In Java environment, there must be a registered charset provider with the given name; **Required**;
- `ibm-name` string value defining AFP codepage specification ("**Txxxxxxx**"); **Required**;

- `forcelatin` a boolean (**true** or **false**) value defining whether the codepage contain Base Latin characters in lower half of code bytes (0x00..0x7F); **Optional**; **Default=true**;
- `desc` providing text description of the code page; **Optional**;

Specially mapped characters are defined by `<character>` element. It has the following attributes:

- `unicode` defining two-byte hexadecimal value of specially mapped character; **Required**;
- `afp` defining one-byte hexadecimal value of mapped character within target Codepage; **Required**;
- `desc` providing text description of the character; **Optional**;

Note: All the remaining characters belonging to Unicode character ranges and not listed with `<character>` elements, are translated to target Codepage using Java standard mechanisms of string translations;

Please refer [Section 6.10, "International Character Set Support"](#) section to find more details on how the AFP generator works with Unicode ranges and Codepages.

The following example demonstrates how to configure necessary Character sets for AFP generator:

```
<generator-options format="AFP">
  <charsets>
    <!--languages-->
    <charset name="Latin">
      <code-range from="0x0000" to="0x007F"/>
      <codepage name="Cp500" ibm-name="T1V10500" forcelatin="true" desc="Base Latin"/>
    </charset>
    <charset name="Latin_1">
      <code-range from="0x0080" to="0x00FF"/>
      <codepage name="Cp819" ibm-name="T1000819" forcelatin="true" desc="Latin_1"/>
    </charset>
    <charset name="Cyrillic">
      <code-range from="0x0400" to="0x04FF"/>
      <codepage name="Cp866" ibm-name="T1000866" forcelatin="true" desc="ANSI Cyrillic"/>
    </charset>
    <charset name="Chinese">
      <code-range from="0x4E00" to="0x9FFF"/>
      <codepage name="Cp950" ibm-name="T1094700" forcelatin="false" desc="Chinese"/>
    </charset>
    <charset name="Hebrew">
      <code-ranges>
        <code-range from="0x0590" to="0x05FF"/>
      </code-ranges>
    </charset>
  </charsets>
</generator-options>
```

```

    </code-ranges>
    <codepage name="Cp424" ibm-name="T1000424" forcelatin="true" desc="Hebrew"/>
</charset>
<charset name="Greek">
    <code-ranges>
        <code-range from="0x0370" to="0x03ff"/>
    </code-ranges>
    <codepage name="Cp875" ibm-name="T1000875" forcelatin="false"/>
</charset>
<!--symbol-->
<charset name="Symbols00259">
    <code-ranges>
        <code-range from="0x03C0" to="0x03C0"/>
        <code-range from="0x2020" to="0x2020"/>
        <code-range from="0x003C" to="0x003C"/>
        <code-range from="0x02C6" to="0x02C6"/>
        <code-range from="0x00B0" to="0x00B0"/>
        <code-range from="0x25CF" to="0x25CF"/>
        <code-range from="0x25C6" to="0x25C6"/>
        <code-range from="0x25A1" to="0x25A1"/>
        <code-range from="0x2341" to="0x2341"/>
        <code-range from="0x25BA" to="0x25BA"/>
    </code-ranges>
    <character-mapping>
        <character unicode="0x03C0" afp="0x46" desc="pi small"/>
        <character unicode="0x2020" afp="0x4b" desc="dagger"/>
        <character unicode="0x003C" afp="0x4c" desc="less"/>
        <character unicode="0x02C6" afp="0x5f" desc="circumflex accent"/>
        <character unicode="0x00B0" afp="0x7c" desc="degree symbol"/>
        <character unicode="0x25CF" afp="0xbc" desc="large bullet"/>
        <character unicode="0x25A1" afp="0xda" desc="open square"/>
    </character-mapping>
    <codepage name="Cp259" ibm-name="T1000259" forcelatin="false"/>
</charset>
<charset name="Cp437">
    <code-ranges>
        <code-range from="0x2022" to="0x2022"/>
        <code-range from="0x266A" to="0x266A"/>
    </code-ranges>
    <character-mapping>
        <character unicode="0x266a" afp="0x0d" desc="musical note"/>
        <character unicode="0x2022" afp="0x07" desc="bullet"/>
    </character-mapping>
    <codepage name="Cp437" ibm-name="T1000437" forcelatin="false"/>
</charset>

```

```

<charset name="Cp423">
  <code-ranges>
    <code-range from="0x03CA" to="0x03CA"/>
  </code-ranges>
  <character-mapping>
    <character unicode="0x03CA" afp="0xb4" desc="acute accent"/>
  </character-mapping>
  <codepage name="Cp423" ibm-name="T1000423" forcelatin="false"/>
</charset>
<charset name="APL Graphic Escape">
  <code-ranges>
    <code-range from="0x25CA" to="0x25CA"/>
    <code-range from="0x25A0" to="0x25A0"/>
    <code-range from="0x203E" to="0x203E"/>
  </code-ranges>
  <character-mapping>
    <character unicode="0x25CA" afp="0x70" desc="acute accent"/>
    <character unicode="0x25A0" afp="0xC3" desc="down caret"/>
    <character unicode="0x203E" afp="0xA0" desc="overbar"/>
  </character-mapping>
  <codepage name="Cp310" ibm-name="T1000310" forcelatin="true"/>
</charset>
</charsets>
</generator-options>

```

Note: Due to the nature of Character set configuration, it must be specified in `xep.xml` configuration file only and cannot be specified or overridden neither in command line parameters nor processing instructions within the XSL-FO document.

6.8.2. Configuring Fonts

- AFPFont options are used for mapping XSL FO fonts to AFP fonts.

Each AFPFont option name starts with "AFPFont" and after a comma contains face name of a XSL FO font. Each AFPFont option value contains a list of nine subvalues separated with commas.

Example:

```

<option name="AFPFont,Helvetica"
  value="C0H200.0, C0H300.0, C0H400.0, C0H500.0,
  C0H201.0, C0H301.0, C0H401.0, C0H501.0, 278"/>

```

Subvalues in the list have following meaning:

1. AFP substitution font for font-weight="normal" font-style="normal"

2. AFP substitution font for font-weight="normal" font-style="italic"
3. AFP substitution font for font-weight="bold" font-style="normal"
4. AFP substitution font for font-weight="bold" font-style="italic"
5. AFP substitution font for symbolic subset and font-weight="normal" font-style="normal"
6. AFP substitution font for symbolic subset and font-weight="normal" font-style="italic"
7. AFP substitution font for symbolic subset and font-weight="bold" font-style="normal"
8. AFP substitution font for symbolic subset and font-weight="bold" font-style="italic"
9. Word spacing value in font relative units (please reference AFP FOCA reference for details)

Note: If the font is not found within the table above, AFP Generator uses **Helvetica** to substitute.

6.8.3. Configuring Highlight Color Table

`HighlightColor` option is used for configuring mapping of colorant to Highlight Color ID within the target AFP device.

Each `HighlightColor` option starts with "HighlightColor" prefix and after comma should contain Color ID (hex or decimal). Value contains symbolic name of colorant.

Example:

```
<option name="highlightcolor,0x301" value="PANTONE Orange 021 M" />
```

or (the same)

```
<option name="highlightcolor,769" value="PANTONE Orange 021 M" />
```

6.8.4. Configuring Shading Patterns

- `USE_SHADING_PATTERNS` specifies whether grayscale-filled areas should be filled with bi-level pattern. Percentage rate of containing black points will be close to required grayscale value.

1 or *true* or *yes* - Shading patterns will be used

0 or *false* or *no* - Shading patterns will not be used (**default**)

Example:

```
<option name="USE_SHADING_PATTERNS" value="yes" />
```

Shading patterns work for rectangular areas only.

Shading patterns are limited for only those areas filled with grayscale color.

There are several patterns hard-coded into AFP backend: 0%, 3.125%, 6.25%, 10%, 12.5%, 20%, 25%, 30%, 37.5%, 40%, 50%, 60%, 62.5%, 70%, 75%, 80%, 87.5%, 90%, 95%, and 100%. If greyscale value does not exactly match any of listed values, the closest match will be used.

Shading patterns, as all bilevel images, are mixed with their background. Their white points appear transparent.

- `USE_REPLICATE_AND_TRIM` specifies if "replicate-and-trim" feature will be used for shading patterns.

`1` or `true` or `yes` - "replicate-and-trim" is used

`0` or `false` or `no` - "replicate-and-trim" is not used (**default**)

If set to "`no`", shading pattern raster image will be created for entire dimensions of rectangle. If set to "`yes`", only 8x8 pixels image will be created. Thus, this feature significantly reduces size of documents with shading patterns enabled, and produces best quality.

Example:

```
<option name="USE_REPLICATE_AND_TRIM" value="yes"/>
```

This option applies only if `USE_SHADING_PATTERNS` equals to `true`.

"Replicate-and-trim" feature is not supported by every AFP device, so it should be turned *off* for older printers without support of this feature.

- `SHADING_PATTERN_RESOLUTION` defines zoom factor for shading pattern raster.

(Default: 1.0)

Can contain any positive decimal value greater than `0` and no greater than `1`

Example:

```
<option name="SHADING_PATTERN_RESOLUTION" value="0.25"/>
```

Shading pattern raster image size is limited to 32kbytes. Thus, if the resolution is set high, it may exceed this limit. To avoid this, `SHADING_PATTERN_RESOLUTION` defines divider for actual raster size. For example, if rectangle area size is 1000x1000 px and `SHADING_PATTERN_RESOLUTION` is set to `0.25` (25%), AFP Backend will produce raster image of size 250*250, and command AFP to stretch it to required dimensions. Note that quality of `0.25` (1/4) will produce raster image 16 times smaller.

This option applies only if `USE_SHADING_PATTERNS` equals to `true` and `USE_REPLICATE_AND_TRIM` equals to `false`.

- `TRY_USING_TIFF_COMPRESSION` option allows the user to specify whether AFP backend attempts to compress shading patterns raster images with TIFF encoding.

1 or *true* or *yes* - AFP Backend attempts to compress shading pattern rasters (**default**)

0 or *false* or *no* - AFP Backend does not attempt to compress shading pattern rasters

Example:

```
<option name="TRY_USING_TIFF_COMPRESSION" value="yes"/>
```

Some rasters cannot be compressed with TIFF. In this case, uncompressed raster image is sent to output. Hard-coded rasters are known to be compressible or not, so AFP Backend does not try to compress uncompressible ones.

The only reason to set this value to *"no"* is when your AFP device does not support TIFF compression.

This option applies only if `USE_SHADING_PATTERNS` equals to *true* and `USE_REPLICATE_AND_TRIM` equals to *false*.

6.8.5. Other Configuration Options

- `AFPLogLevel` option lets users turn on output of additional information related to internal details of processing document elements in AFP generator. This information has various levels of detail, from 0 to 2.

0 - AFP logging is turned off (**default**)

1 - AFP generator prints only warnings

2 - AFP generator prints warnings and information messages

Example:

```
<option name="AFPLogLevel" value="0"/>
```

- `RESOURCE` option lets users turn on generating AFP resources (images, graphics, etc.) into separate resource file. If specified, this option should target to particular file name. If omitted, all resources are put within the main AFP output document

Default: (empty string)

Example:

```
<option name="RESOURCE" value="myresourcefile.afp.res"/>
```

Resource file is always rewritten.

- **RESOLUTION** defines which document resolution will be output within the document. It must be positive integer value supported by target AFP device.

Default: 1440

Example:

```
<option name="RESOLUTION" value="1440"/>
```

- **AFPGrayImage** option, if set to *yes*, turns on embedding of raster images as grayscale images, 8 bit per pixel, uncompressed.

1 or *true* or *yes* - embed raster images as 8 bit

0 or *false* or *no* - embed raster images in their original format (**default**)

Example:

```
<option name="AFPGrayImage" value="no"/>
```

- **USE_PTOCA_LEVEL** defines maximal level of PT:OCA commands subset.

1 - Use PT1 only (**default**)

2 - Use PT1 and PT2 only

3 - Use PT1, PT2, and PT3 subsets

Example:

```
<option name="USE_PTOCA_LEVEL" value="3"/>
```

Different AFP-capable devices support different command subsets. In order to comply with this difference and provide maximum compatibility while keeping highest quality and performance, this option must be set according current printer capabilities.

Please refer Presentation Text Object Content Architecture Reference for more details on specific commands belonging to various PT:OCA subsets.

- **USE_GOCA_LEVEL** defines maximal level of G:OCA commands subset.

0 - Do not use G:OCA commands (**default**)

1 - Use Level 1 only

3 - Use Levels 1 and 3

Example:

```
<option name="USE_GOCA_LEVEL" value="1"/>
```

Different AFP-capable devices may or may not support G:OCA command subsets. In order to provide maximum compatibility, this option must be set according current printer capabilities.

Please refer [Section 6.5, “Graphics Support”](#) for more details on G:OCA implementation in XEP AFP Generator.

- `USE_BCOCA_LEVEL` defines maximal level of BC:OCA commands subset.

`0` - Do not use BC:OCA commands (**default**)

`1` - Use Level 1 only

Example:

```
<option name="USE_BCOCA_LEVEL" value="1"/>
```

Set this parameter to `1` in order to enable generating BC:OCA data within output stream.

Please refer [Section 6.6, “Barcodes Support”](#) for more details on supported barcode types and barcodes implementation notes in XEP AFP Generator.

6.9. Bullets support

AFP Backend supports several ways to produce bulleted text.

- Using external image

In order to use image approach, you should define `<fo:list-item-label>` section as in sample below (assuming you have `bullet.png` file in the same folder with FO file):

```
<fo:list-item-label end-indent="label-end()">
  <fo:block>
    <fo:external-graphic src="url(bullet.png)"
      content-height="100%" content-width="100%"/>
  </fo:block>
</fo:list-item-label>
```

- Using special Unicode symbol.

A special symbol can be used, like in sample below:

```
<fo:list-item-label end-indent="label-end()">
  <fo:block>&#x00b0;</fo:block>
</fo:list-item-label>
```

Note: Unicode character used for text bullets must belong to any of Character Sets configured. The most common Unicode characters `0x2022` and `0x2023` used for circle and triangle bullets belong to General Punctuation Unicode Character Set.

- Using SVG primitive

SVG opens bigger variety of possible bullets. This may include circles, diamonds, stars, and other shapes (filled and not filled ones). They also can be enhanced with effects like shadows, outline, and more.

Here is an example of plain filled square bullet using SVG:

```
<fo:list-item-label start-indent="18pt" text-indent="0pt">
  <fo:block>
    <fo:instream-foreign-object display-align="center">
      <svg:svg width="6pt" height="6pt">
        <svg:rect x="1" y="1" width="5pt" height="5pt" fill="black"/>
      </svg:svg>
    </fo:instream-foreign-object>
  </fo:block>
</fo:list-item-label>
```

Other approaches have not been tested and are not supported. Please refer [Section 6.11, “Limitations of the XEP AFP Generator”](#) for more details.

6.10. International Character Set Support

AFP Generator for XEP has multilanguage support. For each character in text blocks, it detects Character Set the character belongs to (out of character sets listed in configuration file). After that, it uses conversion table to convert the character to the CodePage that AFP device is capable to process.

Here is the description how AFP generator for XEP finds out which Codepage to use.

The source Unicode string is analyzed by characters. For each of them, AFP generator determines Character Set (<character> element in config file) the character belongs to (using the Code Ranges listed in configuration file specified in <code-range> elements). If the range not found, the first configured range is assumed (normally, Base Latin). After the range is found, AFP generator checks whether the character is specified within the list of specially translated characters (<character-mapping> element). If so, the character is translated according to the mapping table (afp attribute). If not, AFP generator uses Java libraries to map the character to the corresponding code page (<codepage> element, (name attribute)). After that, it determines if the character belongs the same code page as the previous one in the same text block. For instance, Chinese and Cyrillic characters cannot reside in the same AFP text block due to different encodings. However, Base Latin may follow the Cyrillic character since Cyrillic code pages usually contain Latin characters in lower half of codes (0x00-0x7F). This approach is determined by `forcelatin` attribute. Finally, after the string of the same code page is composed, it becomes assigned with IBM encoded name (`ibm-name` attribute) and placed to AFPDS stream.

Note: The number of Character ranges makes significant impact on the above logic performance.

Moreover, even if some characters of the documents will fall into wrong Character Set, they may not be printed in case if the AFP device does not support corresponding code pages.

So, it is strongly recommended to remove unused Character Sets from configuration file in order to obtain best results and productivity.

Note: PT:OCA bullets seem to be very fast and effective solution, however it strongly depends on the fonts uploaded to the AFP device. So it requires careful attention configuring PT:OCA bullets against the particular target device.

The complete list of Unicode character sets can be found at W3C Web site. Here is the list of most common Unicode Character sets:

Name	AFP Code-Page	Text Code-Page	Unicode Characters Range	Comment
Basic Latin	T1V10500	Cp500	0x0000-0x007F	Basic Latin is automatically included into all character sets
Latin-1	T1000819	Cp819	0x0080-0x00FF	Contains umlaut characters for Western-European languages
Hebrew	T1000424	Cp424	0x0590-0x05FF	Contains characters for Hebrew
Greek/Coptic	T1000875	Cp875	0x0370-0x03FF	Contains characters for Greek language
Cyrillic	T1000866	Cp866	0x0400-0x04FF	Contains characters for Cyrillic languages
Chinese	T1094700	Cp950	0x4E00-0x9FFF	Contains characters for Chinese language (simplified)

Please refer [Section 6.8.1, “Configuring Character Sets”](#) for more details.

6.11. Limitations of the XEP AFP Generator

- AFP generator uses precision of 1/20 of point so its precision is 50 times worse than in other XEP backends.

- AFP generator has limited support of SVG images. For more information about G:OCA implementation please refer [Section 6.5, “Graphics Support”](#).
- AFP generator does not support lines with styles other than *solid*, *dashed*, and *dotted*; all other lines look *solid* in generated AFP documents.
- AFP generator does not support SVG/G:OCA lines with style other than *solid*; all lines look *solid* in generated AFP documents.
- AFP generator does not support all styles of XSL FO borders (see above limitation on lines). All other borders are drawn as *solid* lines.
- AFP generator does not support some of XEPOUT elements.
- AFP generator does not support colors other than RGB, Greyscale, CMYK, and Spot (Highlight).
- Shading option is not supported for Highlight color. More details in [Highlight Color Support](#).
- Image clipping works only for images having the same (or higher) resolution as AFP document.
- Shading patterns work for rectangular areas only.
- Shading patterns are limited for only those areas filled with grayscale color.
- Bilevel images (including shading patterns) are mixed with their background. Their white points appear transparent.
- AFP backend cannot process strip TIFF images with absent 'RowsPerStrip' tag. 'RowsPerStrip' tag may be absent in TIFF image, although this is not recommended by TIFF Specification (Revision 6.0). This is a limitation of used library, AWT.
- Custom stylesheets implementing Barcodes MUST produce Barcode alone item within SVG block if barcodes are generated with BC:OCA. For example:

```
<svg>
  <!-- nothing before desc -->
  <desc />
  <svg:line />      <!-- only lines displaying barcodes -->
  ...
  <svg:line />
  <!-- nothing after barcodes' lines -->
</svg>
```

- Ligatures are not supported yet; they are displayed as question marks ("?"). In order to avoid this, ligatures must be disabled for each font used.

- Code3of9 barcode does not correctly produce characters: dollar sign (\$), slash (/), plus (+), and percent (%).
- Code128 and 4state-AU barcodes may display wrongly in some cases.
- SVG text support only the following rotation values (in degrees): 0, 90, 180, 270.
- SVG text enclosed into viewBox may be distorted if zoom factors by X and Y axis are different; In this case, root-mean-square value is used.
- International Character Set support: Currently, each Character Set has single CodePage and AFP CodePage assigned, and this cannot be configured. More details in [International Character Set Support](#) section.
- Unicode character used for text bullets must belong to any of supported Character Sets. The most common Unicode characters `0x2022` and `0x2023` belonging to General Punctuation Character Set are not supported.
- Currently, XEP does not support outline AFP fonts.
- Embedding, subsetting and algorithmic slanting of native AFP fonts are not supported.

6.12. Frequently Asked Questions

- **Q: Upon every file I'm trying to process with XEP, the following error is displayed:**

```
"UnsupportedEncodingException: Cp037"
```

A: By default, JRE is installed without **charsets.jar** file. This file is required for XEP. Please run JRE installer and check "**additional languages support**" checkbox.

Note: Actual checkbox name may vary for different versions of JRE.

Appendix A. XSL-FO Conformance

A.1. XSL-FO Support

This appendix describes the implementation of XSL Formatting Objects in XEP — an XSL Engine for PDF developed by RenderX, Inc, version 4.13. It lists all supported formatting objects and their properties, provides information about fallbacks for unsupported objects and discusses details of interpretation of XSL specifications adopted in the engine.

Note: XEP implements **Extensible Stylesheet Language version 1.0** as specified in the *XSL 1.0 Recommendation of October 15, 2001*.

A.1.1. Formatting Objects Supported by XEP

§	Object Name	Supported
6.4.2	<fo:root>	Yes
6.4.3	<fo:declarations>	No
6.4.4	<fo:color-profile>	No
6.4.5	<fo:page-sequence>	Yes
6.4.6	<fo:layout-master-set>	Yes
6.4.7	<fo:page-sequence-master>	Yes
6.4.8	<fo:single-page-master-reference>	Yes
6.4.9	<fo:repeatable-page-master-reference>	Yes
6.4.10	<fo:repeatable-page-master-alternatives>	Yes
6.4.11	<fo:conditional-page-master-reference>	Yes
6.4.12	<fo:simple-page-master>	Yes
6.4.13	<fo:region-body>	Yes
6.4.14	<fo:region-before>	Yes
6.4.15	<fo:region-after>	Yes
6.4.16	<fo:region-start>	Yes
6.4.17	<fo:region-end>	Yes
6.4.18	<fo:flow>	Yes

§	Object Name	Supported
6.4.19	<fo:static-content>	Yes
6.4.20	<fo:title>	No
6.5.2	<fo:block>	Yes
6.5.3	<fo:block-container>	Yes
6.6.2	<fo:bidirectional-override>	Yes
6.6.3	<fo:character>	Yes
6.6.4	<fo:initial-property-set>	Yes
6.6.5	<fo:external-graphic>	Yes
6.6.6	<fo:instream-foreign-object>	Yes ¹
6.6.7	<fo:inline>	Yes
6.6.8	<fo:inline-container>	No ²
6.6.9	<fo:leader>	Yes ³
6.6.10	<fo:page-number>	Yes
6.6.11	<fo:page-number-citation>	Yes
6.7.2	<fo:table-and-caption>	Yes
6.7.3	<fo:table>	Yes
6.7.4	<fo:table-column>	Yes
6.7.5	<fo:table-caption>	Yes
6.7.6	<fo:table-header>	Yes
6.7.7	<fo:table-footer>	Yes
6.7.8	<fo:table-body>	Yes
6.7.9	<fo:table-row>	Yes
6.7.10	<fo:table-cell>	Yes
6.8.2	<fo:list-block>	Yes

¹ <fo:instream-foreign-object> can host SVG graphics.

² All content is placed inline.

³ In this version, only plain text can be put inside leaders with `leader-pattern="use-content"`.

§	Object Name	Supported
6.8.3	<fo:list-item>	Yes
6.8.4	<fo:list-item-body>	Yes
6.8.5	<fo:list-item-label>	Yes
6.9.2	<fo:basic-link>	Yes
6.9.3	<fo:multi-switch>	-
6.9.4	<fo:multi-case>	-
6.9.5	<fo:multi-toggle>	-
6.9.6	<fo:multi-properties>	-
6.9.7	<fo:multi-property-set>	-
6.10.2	<fo:float>	Yes ⁴
6.10.3	<fo:footnote>	Yes
6.10.4	<fo:footnote-body>	Yes
6.11.2	<fo:wrapper>	Yes
6.11.3	<fo:marker>	Yes ⁵
6.11.4	<fo:retrieve-marker>	Yes

A.1.2. Formatting Properties Supported by XEP

§	Property Name	Implemented
7.4.1	source-document	No
7.4.2	role	No
7.5.1	absolute-position	Yes ⁶
7.5.2	top	Yes
7.5.3	right	Yes
7.5.4	bottom	Yes

⁴ Top-floats (float="before") area is drawn on top of the **following** page.

⁵ In the current version, markers cannot be specified as children of <fo:wrapper>.

⁶ absolute-position="fixed" works on <fo:block-container> only.

§	Property Name	Implemented
7.5.5	left	Yes
7.6.1	azimuth	-
7.6.2	cue-after	-
7.6.3	cue-before	-
7.6.4	elevation	-
7.6.5	pause-after	-
7.6.6	pause-before	-
7.6.7	pitch	-
7.6.8	pitch-range	-
7.6.9	play-during	-
7.6.10	richness	-
7.6.11	speak	-
7.6.12	speak-header	-
7.6.13	speak-numeral	-
7.6.14	speak-punctuation	-
7.6.15	speech-rate	-
7.6.16	stress	-
7.6.17	voice-family	-
7.6.18	volume	-
7.7.1	background-attachment	Yes
7.7.2	background-color	Yes
7.7.3	background-image	Yes
7.7.4	background-repeat	Yes
7.7.5	background-position-horizontal	Yes ⁷
7.7.6	background-position-vertical	Yes ⁷

⁷ When the background image is repeated along an axis, its offset on this axis is ignored.

§	Property Name	Implemen- ted
7.7.7	border-before-color	Yes
7.7.8	border-before-style	Yes
7.7.9	border-before-width	Yes
7.7.10	border-after-color	Yes
7.7.11	border-after-style	Yes
7.7.12	border-after-width	Yes
7.7.13	border-start-color	Yes
7.7.14	border-start-style	Yes
7.7.15	border-start-width	Yes
7.7.16	border-end-color	Yes
7.7.17	border-end-style	Yes
7.7.18	border-end-width	Yes
7.7.19	border-top-color	Yes
7.7.20	border-top-style	Yes
7.7.21	border-top-width	Yes
7.7.22	border-bottom-color	Yes
7.7.23	border-bottom-style	Yes
7.7.24	border-bottom-width	Yes
7.7.25	border-left-color	Yes
7.7.26	border-left-style	Yes
7.7.27	border-left-width	Yes
7.7.28	border-right-color	Yes
7.7.29	border-right-style	Yes
7.7.30	border-right-width	Yes
7.7.31	padding-before	Yes
7.7.32	padding-after	Yes

§	Property Name	Implemen- ted
7.7.33	padding-start	Yes
7.7.34	padding-end	Yes
7.7.35	padding-top	Yes
7.7.36	padding-bottom	Yes
7.7.37	padding-left	Yes
7.7.38	padding-right	Yes
7.8.2	font-family	Yes
7.8.3	font-selection-strategy	Yes
7.8.4	font-size	Yes
7.8.5	font-stretch	Yes
7.8.6	font-size-adjust	Yes
7.8.7	font-style	Yes
7.8.8	font-variant	No
7.8.9	font-weight	Yes
7.9.1	country	No
7.9.2	language	Yes
7.9.3	script	No
7.9.4	hyphenate	Yes
7.9.5	hyphenation-character	Yes
7.9.6	hyphenation-push-character-count	Yes
7.9.7	hyphenation-remain-character-count	Yes
7.10.1	margin-top	Yes
7.10.2	margin-bottom	Yes
7.10.3	margin-left	Yes
7.10.4	margin-right	Yes
7.10.5	space-before	Yes

§	Property Name	Implemented
7.10.6	space-after	Yes ⁸
7.10.7	start-indent	Yes
7.10.8	end-indent	Yes
7.11.1	space-end	Yes
7.11.2	space-start	Yes
7.12.1	relative-position	No
7.13.1	alignment-adjust	Yes
7.13.2	alignment-baseline	Yes
7.13.3	baseline-shift	Yes
7.13.4	display-align	Yes
7.13.5	dominant-baseline	Yes
7.13.6	relative-align	Yes ⁹
7.14.1	block-progression-dimension	Yes
7.14.2	content-height	Yes
7.14.3	content-width	Yes
7.14.4	height	Yes
7.14.5	inline-progression-dimension	Yes
7.14.6	max-height	No ¹⁰
7.14.7	max-width	No ¹¹
7.14.8	min-height	No ¹²
7.14.9	min-width	No ¹³
7.14.10	scaling	Yes

⁸ space-after.conditionality="discard" is not implemented, fallback value is "retain".

⁹ Supported on <fo:list-item>. On <fo:table-cell> elements, falls back to relative-align="before".

¹⁰ Maps to height.

¹¹ Maps to width.

¹² Maps to height.

¹³ Maps to width.

§	Property Name	Implemented
7.14.11	scaling-method	No
7.14.12	width	Yes
7.15.1	hyphenation-keep	No
7.15.2	hyphenation-ladder-count	No
7.15.3	last-line-end-indent	Yes
7.15.4	line-height	Yes
7.15.5	line-height-shift-adjustment	Yes
7.15.6	line-stacking-strategy	Yes
7.15.7	linefeed-treatment	Yes ¹⁴
7.15.8	white-space-treatment	Yes
7.15.9	text-align	Yes ¹⁵
7.15.10	text-align-last	Yes
7.15.11	text-indent	Yes
7.15.12	white-space-collapse	Yes ¹⁶
7.15.13	wrap-option	Yes
7.16.1	character	Yes
7.16.2	letter-spacing	Yes
7.16.3	suppress-at-line-break	No
7.16.4	text-decoration	Yes
7.16.5	text-shadow	Yes ¹⁷
7.16.6	text-transform	Yes ¹⁸
7.16.7	treat-as-word-space	No

¹⁴ Value *"treat-as-zero-width-space"* for linefeed-treatment is not implemented. This property does not work on inlines.

¹⁵ `<string>` values for text-align are not implemented. text-align on `<fo:table-and-caption>` is not implemented.

¹⁶ This property does not work on inlines.

¹⁷ Blurred shadows are not supported; blur radius is ignored.

¹⁸ To transform a Unicode character to uppercase/lowercase, XEP uses methods provided by the runtime (Java or .NET). In order for this property to work as expected, you should use correct Unicode values for glyphs in your fonts, and set up local information in your environment properly.

§	Property Name	Implemen- ted
7.16.8	word-spacing	Yes
7.17.1	color	Yes
7.17.2	color-profile-name	No
7.17.3	rendering-intent	No
7.18.1	clear	Yes
7.18.2	float	Yes ¹⁹
7.18.3	intrusion-displace	Yes ²⁰
7.19.1	break-after	Yes
7.19.2	break-before	Yes
7.19.3	keep-together	Yes ²¹
7.19.4	keep-with-next	Yes ²¹
7.19.5	keep-with-previous	Yes ²¹
7.19.6	orphans	Yes
7.19.7	widows	Yes
7.20.1	clip	No
7.20.2	overflow	Yes ²²
7.20.3	reference-orientation	Yes
7.20.4	span	Yes
7.21.1	leader-alignment	No
7.21.2	leader-pattern	Yes
7.21.3	leader-pattern-width	Yes

¹⁹ Two additional values, "inside" and "outside", are supported. Their meaning is the same as in text-align property.

²⁰ "indent" value is not implemented.

²¹ .within-page component is unsupported; it is mapped to .within-column. Only "auto" and "always" values are recognized properly: numeric values are treated as "always". In tables, keep-with-previous/keep-with-next traits ignore table headers and footers: e.g. keep-with-previous condition specified on a row will keep it with the previous one regardless of the intervening header. If specified on the first row of the first <fo:table-body> in a table, keep-with-previous will attach the whole table to the preceding block-level element.

²² Supported on side floats and absolutely positioned and rotated block-containers with fixed dimensions. When "error-if-overflow" is specified, a warning is issued on overflow, and the element is discarded in the same way as for "hidden" value.

§	Property Name	Implemen- ted
7.21.4	leader-length	Yes
7.21.5	rule-style	Yes
7.21.6	rule-thickness	Yes
7.22.1	active-state	-
7.22.2	auto-restore	-
7.22.3	case-name	-
7.22.4	case-title	-
7.22.5	destination-placement-offset	No
7.22.6	external-destination	Yes ²³
7.22.7	indicate-destination	No
7.22.8	internal-destination	Yes
7.22.9	show-destination	Yes ²⁴
7.22.10	starting-state	-
7.22.11	switch-to	-
7.22.12	target-presentation-context	-
7.22.13	target-processing-context	-
7.22.14	target-style-sheet	-
7.23.1	marker-class-name	Yes
7.23.2	retrieve-class-name	Yes
7.23.3	retrieve-position	Yes
7.23.4	retrieve-boundary	Yes
7.24.1	format	Yes
7.24.2	grouping-separator	No

²³ In PDF and PostScript generators, URLs starting with explicit "file:" protocol specification are rendered as PDF-to-PDF links ("remote go-to actions"). All other links are treated as Internet URIs, and open in a browser.

²⁴ show-destination is honored for creation of links between PDF documents ("remote go-to actions") in PDF and PostScript generators. In other cases, the attribute is not applicable.

§	Property Name	Implemen- ted
7.24.3	grouping-size	No
7.24.4	letter-value	No
7.25.1	blank-or-not-blank	Yes
7.25.2	column-count	Yes
7.25.3	column-gap	Yes
7.25.4	extent	Yes
7.25.5	flow-name	Yes
7.25.6	force-page-count	Yes
7.25.7	initial-page-number	Yes
7.25.8	master-name	Yes
7.25.9	master-reference	Yes
7.25.10	maximum-repeats	Yes
7.25.11	media-usage	No
7.25.12	odd-or-even	Yes
7.25.13	page-height	Yes
7.25.14	page-position	Yes
7.25.15	page-width	Yes
7.25.16	precedence	Yes
7.25.17	region-name	Yes
7.26.1	border-after-precedence	Yes
7.26.2	border-before-precedence	Yes
7.26.3	border-collapse	Yes
7.26.4	border-end-precedence	Yes
7.26.5	border-separation	Yes
7.26.6	border-start-precedence	Yes

§	Property Name	Implemen- ted
7.26.7	caption-side	Yes ²⁵
7.26.8	column-number	Yes
7.26.9	column-width	Yes
7.26.10	empty-cells	No ²⁶
7.26.11	ends-row	Yes
7.26.12	number-columns-repeated	Yes
7.26.13	number-columns-spanned	Yes
7.26.14	number-rows-spanned	Yes
7.26.15	starts-row	Yes
7.26.16	table-layout	Yes
7.26.17	table-omit-footer-at-break	Yes
7.26.18	table-omit-header-at-break	Yes
7.27.1	direction	Yes
7.27.2	glyph-orientation-horizontal	No
7.27.3	glyph-orientation-vertical	No
7.27.4	text-altitude	Yes
7.27.5	text-depth	Yes
7.27.6	unicode-bidi	Yes ²⁷
7.27.7	writing-mode	Yes ²⁸
7.28.1	content-type	Yes
7.28.2	id	Yes
7.28.3	provisional-label-separation	Yes

²⁵ Only *"before"* and *"after"* values are implemented: *caption-side="start"* falls back to *"before,"* and *caption-side="end"* falls back to *"after."*

²⁶ In the current implementation, all cells present in the source document are shown regardless of whether their content is empty; cells not presented in the source aren't visible at all.

²⁷ Bidi implementation differs from Unicode Bidi algorithm: any markup element opens a new level of embedding. Consequently, *unicode-bidi="normal"* is not supported (treated as *"embed"*); see detailed discussion below.

²⁸ Only *"lr-tb"* and *"rl-tb"* values are supported. All other values are treated as *"lr-tb."*

§	Property Name	Implemen- ted
7.28.4	provisional-distance-between-starts	Yes
7.28.5	ref-id	Yes
7.28.6	score-spaces	No
7.28.7	src	Yes ²⁹
7.28.8	visibility	No
7.28.9	z-index	Yes ³⁰
7.29.1	background	Yes
7.29.2	background-position	Yes
7.29.3	border	Yes
7.29.4	border-bottom	Yes
7.29.5	border-color	Yes
7.29.6	border-left	Yes
7.29.7	border-right	Yes
7.29.8	border-style	Yes
7.29.9	border-spacing	Yes
7.29.10	border-top	Yes
7.29.11	border-width	Yes
7.29.12	cue	-
7.29.13	font	Yes
7.29.14	margin	Yes
7.29.15	padding	Yes
7.29.16	page-break-after	Yes
7.29.17	page-break-before	Yes
7.29.18	page-break-inside	Yes

²⁹ In addition to protocols provided by the runtime (Java or .NET), XEP supports data: URI scheme ([RFC 2397](https://tools.ietf.org/html/rfc2397)).

³⁰ z-index property is supported for block-containers with absolute-position="fixed".

§	Property Name	Implemented
7.29.19	pause	-
7.29.20	position	Yes
7.29.21	size	Yes
7.29.22	vertical-align	Yes
7.29.23	white-space	Yes
7.29.24	xml:lang	No

A.1.3. Notes on Formatting Objects Implementation

`<fo:block>`

According to the specification, an empty block that has a non-null padding and/or border should be visible. XEP suppresses all blocks that have no visible contents regardless of their border or padding attributes.

`<fo:bidirectional-override>`

In the current implementation of bidi algorithm, any markup element opens a new level of embedding. Consequently, `unicode-bidi="normal"` is not supported:

`<fo:bidirectional-override>` behaves as if `unicode-bidi="embed"` were specified.

`<fo:inline-container>`

Unsupported; contents are placed inline.

`<fo:multi-switch>`

`<fo:multi-case>`

`<fo:multi-toggle>`

`<fo:multi-properties>`

`<fo:multi-property-set>`

Unsupported; contents are ignored. These elements deal with interactivity. Since PDF and PostScript are intrinsically static formats, none of them are applicable.

`<fo:float>`

The before-float appears at the top of the **next** page.

`<fo:table-caption>`

Only *"before"* and *"after"* captions are implemented. Side captions are treated as follows: `caption-side="start"` falls back to *"before"*, and `caption-side="end"` falls back to *"after"*.

<fo:table-footer>

Table footer repetition is not implemented. The element is drawn once at the end of table.

<fo:table-column>

In the collapsed border model, only border-start and border-end are supported on <fo:table-column> elements.

<fo:table-row>

In the collapsed border model, only border-before and border-after are supported on <fo:table-row> elements.

<fo:table-cell>

If a cell spans multiple rows in a table with a collapsed border model, its border-after is taken from the row where the cell begins.

<fo:leader>

In this version, leaders with `leader-pattern="use-content"` can only contain plain text inside; all formatting is lost.

<fo:marker>

This version cannot process markers specified as children of an <fo:wrapper>.

A.1.4. Supported Expressions

XEP implements a subset of XSL algebraic expressions. The following operators and functions are recognized:

- Arithmetical operators: +, -, *, div, mod
- floor()
- ceiling()
- round()
- abs()
- max()
- min()
- rgb()
- rgb-icc() (supported partially — see notes below)

- `from-nearest-specified-value()`
- `from-parent()`
- `from-table-column()`
- `inherited-property-value()`
- `proportional-column-width()`
- `body-start()` (standalone use only, cannot be an operand in expressions)
- `label-end()` (standalone use only, cannot be an operand in expressions)

Function `rgb-icc()` recognizes four predefined color profile names: `#Grayscale`, `#CMYK`, `#SpotColor`, and `#Registration` (see details below). For any other value of the fourth parameter, the function returns the fallback RGB color. ICC profiles are not supported.

Support for expressions is subject to the following limitations:

- For compound expressions, the result of evaluation of all intermediate subexpressions must be a valid XSL type. For example, expression `(2in * 2in) div 1in` is not supported because its first subexpression yields dimensionality of square inches, which is not a valid XSL unit; while `2in * (2in div 1in)` works.
- Expressions that require knowledge of layout to evaluate (e.g. Block widths expressed in percentages) can only be used as standalone expressions, not parts of a bigger expression, and cannot be referenced by property-value functions. The same limitation applies to `body-start()` and `label-end()` functions.
- Property value functions (`from-nearest-specified-value()`, `from-parent()`, `from-table-column()`, `inherited-property-value()`) cannot be used in shorthands, and cannot take shorthand property names as their arguments.
- Property value functions that take `start-indent/end-indent` as arguments may not work correctly if the block with indents is placed into another block that has CSS-style `margin-*` attributes. For safety, use either expressions with indents, or CSS margins; mixing these two coding styles in the same stylesheet may yield unpredictable results.

A.1.5. Color Specifiers

XEP can produce PDF and PostScript output using the following color types:

1. **Grayscale.** The following specifiers produce grayscale color output:
 - Predefined HTML and SVG names that correspond to RGB values with $R = G = B$:
`white`, `black`, `silver`, `gray`, `grey`, `lightgray`, `lightgrey`, `darkgray`, `darkgrey`, `dimgray`, `dimgrey`, `whitesmoke`, `gainsboro`.

- HTML-style RGB values with $R = G = B$: #555, #9D9D9D, etc.
- `rgb-icc()` function with built-in #Grayscale pseudo profile. Gray tone intensity is specified as a real value in the range 0.0–1.0, the 5th argument to the function. Example:

```
rgb-icc (128, 128, 128, #Grayscale, 0.5)
```

2. **RGB.** The following specifiers produce RGB color output:

- HTML and SVG predefined names, and RGB specifiers that are not mentioned above.
- `rgb()` function. Values of color components are specified as real values in the range 0.0–255.0. Example:

```
rgb (127.5, 39.86, 255)
```

3. **CMYK.** The following specifier produce CMYK color output:

- `rgb-icc()` function with built-in #CMYK pseudo profile. Ink values are specified as real values in the range 0.0–1.0, arguments from 5th to 8th; order of inks is *cyan-magenta-yellow-black*. Example:

```
rgb-icc (255, 255, 0, #CMYK, 0, 0, 1, 0)
```

4. **Spot colors.** The following specifiers produce spot color output:

- `rgb-icc()` function with built-in #SpotColor pseudo profile. The 5th argument is the colorant name, specified as a string; use quotes if the name contains spaces. The 6th argument is the tint value, specified as a real number in the range 0.0–1.0. These mandatory attributes may be followed by an optional specification of the alternate color for the colorant, in either CMYK or grayscale color space: 7th argument is the color space name (either #CMYK or #Grayscale), and the rest are component intensities (1 for grayscale, 4 for CMYK).

Note: The alternate color specifies an equivalent representation **for the full colorant intensity**. Occurrences of the same spot color with different tints should have the same alternate color specifier.

If the alternate color is not specified, XEP looks it up in SpotColor matching table (path to the table is defined by the <SPOT_COLOR_TRANSLATION_TABLE> option); if no matches found, black color in grayscale color space is used.

Examples:

```
rgb-icc(255,255,0, #SpotColor,'PANTONE Orange 021 C',0.33)
rgb-icc(255,255,0, #SpotColor,'PANTONE 169 M',0.5, #CMYK,0,0.2,0.2,0)
rgb-icc(255,255,0, #SpotColor,MyColor,0.33, #Grayscale,0.5)
```

5. **Registration color.** The following specifier produces registration (all-colorants) color output:

- `rgb-icc()` function with built-in `#Registration` pseudo profile. Tint intensity is specified as a real value in the range 0.0–1.0, the 5th argument to the function. Example:

```
rgb-icc (128, 128, 128, #Registration, 0.5)
```

A.1.6. XSL 1.1 Support

XEP includes stylesheets to convert XSL 1.1 elements and attributes to elements and attributes from XSL 1.0 and RenderX extensions. All elements and attributes which exist in both XSL 1.1 and XSL 1.0 are left without changes. The elements and attributes which exist only in XSL 1.1 are converted into corresponding elements and attributes from RenderX extensions. The stylesheets support conversion for the following features:

1. Document Outline (Bookmarks)
2. Indexes
3. Last Page Number Reference
4. Change Bars
5. Folio Prefix and Suffix

Since there is no correspondence in RenderX extensions for some elements and attributes from XSL 1.1, they will be ignored during conversion. Following is a list of unsupported elements and attributes:

- `fo:page-sequence-wrapper`
- `fo:flow-map`
- `fo:flow-assignment`
- `fo:flow-source-list`
- `fo:flow-target-list`
- `fo:flow-name-specifier`
- `fo:region-name-specifier`
- `@allowed-height-scale`
- `@allowed-width-scale`
- `fo:index-page-number-prefix`

- fo:index-page-number-suffix
- @index-class
- @merge-ranges-across-index-key-references
- @merge-pages-across-index-key-references

Document Outline (Bookmarks)

§	XSL 1.1 Object/Property Name	RenderX Extensions Object/Property Name
6.11.1	<fo:bookmark-tree>	<rx:outline>
6.11.2	<fo:bookmark>	<rx:bookmark>
6.11.3	<fo:bookmark-title>	<rx:bookmark-label>
7.23.6	external-destination	external-destination
7.23.8	internal-destination	internal-destination
7.23.10	starting-state	collapse-subtree

Indexes

§	XSL 1.1 Object/Property Name	RenderX Extensions Object/Property Name
6.10.2	<fo:index-page-number-prefix>	No correspondence, ignored
6.10.3	<fo:index-page-number-suffix>	No correspondence, ignored
6.10.4	<fo:index-range-begin>	<rx:begin-index-range>
6.10.5	<fo:index-range-end>	<rx:end-index-range>
6.10.6	<fo:index-key-reference>	<rx:index-item>
6.10.7	<fo:index-page-citation-list>	<rx:page-index>
6.10.8	<fo:index-page-citation-list-separator>	list-separator
6.10.9	<fo:index-page-citation-range-separator>	range-separator
7.24.1	index-class	No correspondence, ignored
7.24.2	index-key	rx:key

§	XSL 1.1 Object/Property Name	RenderX Extensions Object/Property Name
7.24.3	page-number-treatment	link-back
7.24.4	merge-ranges-across-index-key-references	No correspondence, ignored
7.24.5	merge-sequential-page-numbers	merge-subsequent-page-numbers
7.24.6	merge-pages-across-index-key-references	No correspondence, ignored
7.24.7	ref-index-key	<rx:index-item>/ref-key
7.30.8	id	id
7.30.13	ref-id	ref-id

Last Page Number Reference

§	XSL 1.1 Object Name	RenderX Extensions Object Name
6.6.12	<fo:page-number-citation-last>	<rx:page-number-citation-last>

The only required attribute, `ref-id`, specifies the id of the element whose last page number is retrieved.

Change Bars

§	XSL 1.1 Object Name	RenderX Extensions Object Name
6.13.2	<fo:change-bar-begin>	<rx:change-bar-begin>
6.13.3	<fo:change-bar-end>	<rx:change-bar-begin>

All properties of `<fo:change-bar-begin>` and `<fo:change-bar-end>` map to themselves.

Folio Prefix and Suffix

§	XSL 1.1 Object Name	XSL 1.0 Object Name
6.6.13	<fo:folio-prefix>	<the content>
6.6.14	<fo:folio-suffix>	<the content>

The content of `<fo:folio-prefix>` (`<fo:folio-suffix>`) is added inline before (after) all occurrences of `<fo:page-number>`, `<fo:page-number-citation>`, and `<fo:page-number-citation-`

last> referring to elements in the respective <fo:page-sequence>. In case of <fo:page-number> it is always the current page sequence.

A.1.7. Extensions to the XSL 1.0 Recommendation

XEP implements several extensions to the Specification, placed into a separate namespace: `xmlns:rx="http://www.renderx.com/XSL/Extensions"`. They add support for useful functionality that cannot be expressed by XSL Formatting Objects.

Document Information

This extension permits passing a set of name/value pairs to the generator of the output format. A typical application is setting PDF document info fields ('Author' and 'Title'). Implementation uses two extension elements: <rx:meta-info> and <rx:meta-field>.

<rx:meta-info>

This element is merely a container for one or more <rx:meta-field> elements. It should be the first child of <fo:root>.

<rx:meta-field>

This element specifies a single name/value pair. It has two mandatory attributes: `name` and `value`. Current implementation of the PDF and PostScript generators recognize four possible values for `name`:

- `name="author"` - fills the 'Author' field in the resulting PDF file with a string specified by the `value` property.
- `name="creator"` - fills the 'Creator' field.
- `name="title"` - fills the 'Title' field.
- `name="subject"` - fills the 'Subject' field.
- `name="keywords"` - fills the 'Keywords' field.

The 'Producer' field in the PDF file is set to `"XEP 4.13"`; there is no means to control it from the source file. All other values for `name` are treated as custom meta-fields and appear in the same dictionaries in PostScript and PDF as predefined meta-fields. Unicode values for `name` are not supported.

In the PostScript generator module, the document info fields are added using the `pdfmark` operator. The respective fields are filled when PostScript is converted to PDF using *Adobe Acrobat Distiller* or *GhostScript*.

Document Outline (Bookmarks)

Implementation of a document outline uses the following three extension elements:

- `<rx:outline>` - The top-level element of the document outline tree. It should be located before any `<fo:page-sequence>` elements, and after the `<fo:layout-master-set>` and the `<fo:declarations>` elements (if present). It contains one or more `<rx:bookmark>` elements.
- `<rx:bookmark>` - This element contains information about a single bookmark. It contains a mandatory `<rx:bookmark-label>` element as its first child, and zero or more nested `<rx:bookmark>` elements that describe nested bookmarks. Bookmark destination is expressed either by `internal-destination` property (for internal navigation), or by `external-destination` (for extra-document links). The initial presentation of the children bookmarks is controlled by `collapse-subtree` attribute. Values are either `"true"` (collapse children) or `"false"` (expand children).
- `<rx:bookmark-label>` - This element contains text of a bookmark label. It must be the first child of its parent `<fo:bookmark>`. Content of this element should be plain text.

Indexes

Building page number lists for back-of-the-book indexes is a common task. It is relatively easy to collect a list of references to index terms in the text; but then, to turn them into a real index entry, you should exclude repeated page numbers and merge adjacent numbers into ranges. Neither of these two operations can be done in XSL 1.0. Therefore, XEP supports an extension for this purpose.

The task of building an index can be split in two subtasks:

- Mark up occurrences of index terms in the main text.
- Specify composition and formatting of page number lists in the index.

Index Term Markup

In order to mark up occurrences of the index terms in the text, XEP introduces a special extension attribute: `rx:key`. It can be specified on any element that can take an `id` attribute; unlike the latter, it need not be unique across the document. Its value is used as a key to select elements for the page number list. For example, an index term to the word "rendering" might look like this:

```
The process of converting XSL-FO to a printable format  
is called <fo:inline rx:key="key.render">rendering.</fo:inline>
```

There is also a mechanism to specify an explicit range, not distinct elements. Two extension elements serve this purpose:

<rx:begin-index-range>

Starts a range. It takes two attributes, both required:

id

A unique identifier used to define the limits of the range.

rx:key

Index key used to select the range into a page number list.

<rx:end-index-range>

Ends a range. It takes one attribute, required:

ref-id

A reference to the **id** attribute of the **<rx:begin-index-range>** that started the range.

These two elements always form a pair. These elements may be located anywhere inside **<fo:flow>**; there are no constraints on their nesting with respect to other elements.

Index Entries

In the index, the actual page reference is created by another extension element, **<rx:page-index>**. It picks elements from the text by their **rx:key** properties, and produces a sorted list of their page numbers, eliminating duplicates.

<rx:page-index> should contain one or more **<rx:index-item>** elements as children. Each **<rx:index-item>** has a required **ref-key** attribute, and selects elements that have an **rx:key** attribute with the same value.

A distinct element bearing the appropriate **rx:key** value is represented as follows:

- If it fits completely onto one page, it is represented as a single page number.
- If it spans multiple pages, its entry is formatted as a range from the first to the last of the spanned pages.

A range (created by a **<rx:begin-index-range>** and **<rx:end-index-range>** element pair) is represented as a range from the page where **<rx:begin-index-range>** is located to the page of its matching **<rx:end-index-range>**.

A basic entry in an index looks like this:

```
<fo:inline rx:key="key.elephant">Elephants</fo:inline> live in Africa. ...
<fo:inline rx:key="key.elephant">African elephants</fo:inline> have big ears ...
...
<fo:block text-align="center" font="bold 16pt Futura">INDEX</fo:block>
<fo:block>
  Elephants <rx:page-index>
    <rx:index-item ref-key="key.elephant"/>
  </rx:page-index>
</fo:block>
```

There are other attributes of `<rx:index-item>` to control the formatting of the index entry:

range-separator

Specifies the string used to separate page numbers that form a continuous range.
Default is en dash: "-" (U+2013).

merge-subsequent-page-numbers

Controls whether sequences of adjacent page numbers should be merged into ranges.
Default is "false."

link-back

If set to "true", page numbers are made into hyperlinks to the corresponding page.
Default is "false."

Besides that, `<rx:index-item>` can take additional inline attributes, applied to each page number generated from this element. This allows for different presentation styles across the list, e.g. To make references to primary definitions bold.

Flow Sections

Flow sections permit splitting the flow into subflows, with different column counts in each subflow. The following element creates flow sections:

```
<rx:flow-section>
```

This element must be a direct child of `<fo:flow>`. It can be mixed with other block-level elements. It takes two attributes: `column-count`, the number of columns for the subflow, and `column-gap`, the space between the columns.

Last Page Number Reference

This extension element retrieves the number of the last page occupied by a particular element. Its syntax and semantics are similar to `fo:page-number-citation`.

<rx:page-number-citation-last>

The only required attribute, `ref-id`, specifies the `id` of the element whose last page number you want to retrieve. In particular, by referencing the `id` of the `<fo:root>` element, it is possible to retrieve the number of the last page in the document.

Note: This element is described in XSL 1.1 Working Draft of 17 December 2003. In subsequent versions of XEP, it is likely to move to the standard XSL-FO namespace.

Change Bars

XEP has support for change regions, as described in XSL 1.1 Working Draft of December 16, 2004.

<rx:change-bar-begin>**<rx:change-bar-end>**

These elements have exactly the same meaning and properties as listed in the Working Draft for elements `<fo:change-bar-begin>` and `<fo:change-bar-end>`, sections 6.3.12 and 6.3.13, respectively. In future versions of XEP, when XSL 1.1 will become the W3C Recommendation, they will be moved to the standard XSL-FO namespace.

Note: The content model for these elements is different than the description in the Working Draft. The Working Draft, Section 6.2, says the following about change-bar-begin/end elements: ["The following formatting objects are "neutral" containers and may be used, provided that the additional constraints listed under each formatting object are satisfied, anywhere where `#PCDATA`, `%block`; or `%inline`; are allowed".] This essentially forbids change-bar-begin/end elements to appear almost anywhere in the lists or tables, for example, it's not possible to mark a whole list-item or table-cell as "changed." XEP implementation does not have such limitations, change bar anchors can be placed almost anywhere in the flow.

Background Image Scaling and Content Type

In XSL 1.0, there is no provision to scale/size a background image. XEP implements this functionality via the following extension properties:

rx:background-content-height**rx:background-content-width****rx:background-scaling****rx:background-content-type**

These properties have exactly the same semantics as `content-height`, `content-width`, `scaling`, and `content-type`, respectively. They apply to the image specified in `background-image` property (or inside `background` shorthand).

Initial Destination

This extension allows you to specify the destination to jump to when the document is first opened. It uses a single extension attribute, `rx:initial-destination` placed on `<fo:root>`; its syntax is the same as the `internal-destination` attribute.

Omitted Initial Header in Tables

This extension permits you to omit a table header at the beginning of a table. This feature can be used to create "continuation headers", which are output only on page breaks. It uses a single extension attribute, `rx:table-omit-initial-header` placed on `<fo:table>`. The property has a Boolean value: `"true"` or `"false"` — same as for `table-omit-header-at-break`.

Base URI Definition: `xml:base`

XEP recognizes and processes `xml:base` attribute, defined in [XML Base Recommendation](#). It permits you to set the base for resolving relative URIs (link targets, image locations, fonts, hyphenation patterns, etc) for the whole document or a single subtree.

Note: The use of `xml:base` in XSL is not authorized by the XSL Specification; therefore, this option should be considered a proprietary extension to XSL.

Border and Padding on Regions

In the XSL Recommendation, border and padding properties are permitted on region elements (`<fo:region-body>`, `<fo:region-before>`, `<fo:region-after>`, `<fo:region-start>`, and `<fo:region-end>`). However, they may accept values of 0 (**sic!**). In XEP, non-zero values of these properties result in a border around the respective region area, and its content rectangle is padded by the specified amount.

Note: When validation strictness level is 2, the validator issues a warning about nonzero borders and padding on regions.

Floats Alignment

Floating figures often need to float towards different sides of the page depending on their parity. However in XSL 1.0 Recommendation there is no means to achieve such effect. XEP supports two additional values for `float` property of the `<fo:float>` element. Those values are: `"inside"` and `"outside"`. Their meaning is the same as in `text-align` property defined by XSL 1.0 Recommendation: `"inside"` value aligns floating block to the inner edge of the page (left for odd pages, right for even pages) and `"outside"` aligns floating block to the outer edge of the page (right for odd pages, left for even pages). This functionality is often used to create margin notes known as "marginalia."

Multicolumn Footnotes

Some documents have many short footnotes per page, and according to the Recommendation all the footnotes are stacked on top of each other. This results in a lot of white space to the right of the footnotes in footnote-reference-area.

XEP supports two additional attributes: `footnote-column-count` and `footnote-column-gap` on `<fo:region-body>`. They have the same meaning as `column-count` and `column-gap` and result in footnote-reference-area having the required number of columns separated with gaps. XEP balances the footnotes among the columns in footnote-reference-area, which makes the area be filled better and have smaller height, leaving more space for the body.

Note: The balancing algorithm is iterative and may affect performance in corner cases. The best quality of balancing is achieved in the most common cases: for short footnotes.

Unique Footnotes

There is a user's request to collapse footnote-bodies on a page if their anchors read the same. This is useful if, for example, several values in table cells must be marked with one and the same note. The Recommendation does not provide a way to achieve this, because one must know beforehand how the footnotes will be distributed among pages.

XEP can handle this request properly. A footnote-body will not be added to the footnote-reference-area if there is a footnote-body starting on this page which has the same value of `id`. In other words, footnotes with equal `footnote-body/@id` collapse to one per page.

Note: If a footnote-body starts on page N and continues on page N+1, there may appear another footnote-body with the same `id` on page N+1: the tail of a footnote may not collapse.

Note: Collapsed footnote-bodies are treated as if they were empty. Any special content (a term for the index, a target for a link, a part of a 'paired' element such as a change-bar or an index-range) will be ignored. Avoid using such content together with the 'unique footnotes' feature.

Watermark

In mass print large number of pages differ in content, but not in static regions. XEP spends a significant share of time formatting static regions on each page. The request is to avoid formatting common parts of pages on each page to save time, and instead pick them up from an XEPOUT file prepared beforehand.

XEP provides an extension for this request: `rx:watermark` attribute on `<fo:simple-page-master>`. The value of `rx:watermark` is an URI reference to an XEP intermediate format file.

For every page created with a given page master, the content of the first `<xep:page>` of master's watermark file will be drawn before anything else on the page.

Note: This extension saves formatting time for static contents, but does not reduce the time required to generate it to an output format.

Note: Do not forget to remove targets and bookmarks from a watermark file.

Note: No scaling is performed on the content of the watermark file, it is 'played' as is.

Transpromo

Empty space often appears at the bottom of pages, especially of the last pages of page sequences. This space may be used for ads. Arbitrary content of a flow makes it impossible to tell how much space will be left on the last page beforehand, so for arbitrary content there is no way to determine the size of the ads box that will fit without making the flow content go to yet another page.

Having a set of ads boxes of different size, users need a way to place the largest such box (just one) that fits on the last page.

'Transpromo' is an extension to the page master selection algorithm that makes XEP iterate over a set of page masters that suite for 'last' until it finds one where all the tail of the flow content fits. For the sake of compatibility, alternatives for 'any' are not considered in the loop.

With the extended algorithm users may specify a set of `<fo:conditional-page-master-references>`, all with `page-position='last'`, in desired order. These page masters may, for example, have different extent on `<fo:region-after>`, from large to small values (in order of reading the `<fo:conditional-page-master-references>` in `<fo:repeatable-page-master-alternatives>`). The `<fo:page-sequence>` will have the respective set of `<fo:static-contents>` with the ads boxes. The largest box that fits together with the flow content will succeed, and formatting will end.

The extended algorithm works similarly for the page masters for 'only'.

Appendix B. Linguistic Algorithms

B.1. Line-Breaking Algorithm

The following rules comprise XEP's line-breaking algorithm:

1. Line-break is permitted if one of the following conditions is fulfilled:
 - Line-break is forced by the explicit linefeed characters: U+000A, U+000D, U+2028, and U+2029. Note, however, that the default behavior of XEP is to perform **linefeed normalization**, which treats all linefeed characters like spaces. Therefore, the linefeed characters actually force a line-break only if the `linefeed-treatment` attribute is set to "preserve."
 - Line-break is permitted at space characters: U+0009, U+0020, U+2000 - U+200B, and U+3000.
2. Line-break is not allowed in the following cases, **unless** one of the conditions of rule 1 is fulfilled:
 - Immediately preceding or following non-breaking spaces (U+00A0) and non-breaking hyphens (U+200C).
 - Immediately preceding trailing punctuation characters, closing brackets and quotes, small Katakana and Hiragana characters, superscript characters, etc.
 - Immediately after opening brackets and quotes, Spanish leading punctuation, currency symbols, etc.
3. If the `hyphenate` attribute is set to "true" and all hyphenation conditions (`hyphenation-push-character-count`, `hyphenation-remain-character-count`, etc.) are satisfied, then line-break is permitted after a soft hyphen (U+00AD). A soft hyphen at the end of a line is replaced by the text specified in the `hyphenation-character` attribute; all other soft hyphens are suppressed.
4. Unless prohibited by the above rules, line-break is permitted before or after CJK ideographic, Katakana, Hiragana, and Hangul characters.
5. In all other cases, line-break is prohibited.

The algorithm will be refined in future versions of XEP, when more feedback about non-European scripting systems is received.

B.2. Hyphenation

XEP uses Unicode soft hyphen characters (U+00AD) to mark possible hyphenation points. These characters can either be contained in the source XSL-FO document (e.g. from an external hyphenation application), or can be added by XEP automatically before the source is passed to the formatter.

The hyphenator implements Liang's algorithm. XEP's distribution includes patterns for the following languages: English (American and British), French, German, Spanish and Russian. All patterns are borrowed from CTAN (the Comprehensive TeX Archive Network, <http://www.ctan.org/>), with some modifications for non-English patterns. More patterns can be added if necessary.

B.2.1. Hyphenation Patterns

The hyphenator uses T_EX format for hyphenation patterns. It recognizes the following sections in the pattern files:

- patterns (for hyphenation patterns)
- hyphenation (for exceptions)

Any other section in the pattern file is ignored. Hexadecimal escape codes (e.g. `^ae`) and control characters (`^A`) are supported; they can be used to encode non-ANSI European characters. Additionally, XEP recognizes a set of `\rm` macros for accented characters: `\a` is â (a with circumflex accent), `\l` is ł (Polish barred l), etc.

B.3. Support for Right-to-Left Writing Systems

XEP supports both left-to-right and right-to-left text. To define ordering of characters within lines and stacking direction of lines into paragraphs, we use the `writing-mode` attribute. It can be specified on the `<fo:simple-page-master>`, `<fo:region-*>`, `<fo:table>`, `<fo:block-container>`, and `<fo:inline-container>` elements. Its primary values are:

- `"lr-tb"`: left-to-right, top-to-bottom. This is the default writing mode in XSL-FO; it is used by the majority of world languages, including English.
- `"rl-tb"`: right-to-left, top-to-bottom. This mode is used in Arabic writing system (adopted by many languages of the Middle East), Hebrew, and Syriac alphabets.
- `"tb-rl"`: top-to-bottom, right-to-left. This way of writing is widely used for Japanese, but also for Chinese and other languages of East Asia.

Note: As of version 4.0, XEP supports only horizontal writing modes: `"lr-tb"` and `"rl-tb"`.

The `writing-mode` attribute defines every aspect of the document organization: binding edge, column ordering in tables, text alignment in blocks, etc. It also sets the correspondence between relative directions (before - after - start - end) and absolutely-oriented ones (top - bottom - left - right).

B.3.1. Bidirectionality

Bidirectionality is the interleaving of text which is to be displayed in both directions: for example, operating instructions are in Hebrew, but the name of the product appears in the middle of the instructions, in English. In simple situations, the renderer handles the bidirectionality on its own; there are, however, many situations where there may be an ambiguity as to the exact resolution desired. For these situations, XSL defines a special element, `<fo:bidirectional-override>` that enables altering the bidirectional behaviour of the whole text or its parts. It has two properties:

`direction`

Sets the dominant direction for a span of text. Possible values are:

- `"ltr"` — from left to right
- `"rtl"` - from right to left

`unicode-bidi`

Specifies behaviour of a text span with respect to the Unicode bidi algorithm. Possible values are the following:

- `"normal"` — order characters by Unicode bidi.
- `"embed"` — open a new level of embedding.
- `"bidi-override"` — ignore directionality of the text and arrange characters in the order specified by the `direction` property.

B.3.2. Glyph Shaping

XEP supports contextual selection of Arabic positional glyph variants, known as **glyph shaping**. Shaping proceeds as follows: each character that belongs to Arabic Unicode range U+0600-U+06FF is replaced by its counterpart in the Arabic Presentation Forms ranges U+FB50-U+FDFF and U+FE70-U+FEFF, in accordance with the Unicode rules for Arabic. Only basic changes are considered:

- Substitution of initial, final, and medial forms
- Insertion of lam-alef ligatures

Shaping occurs before font selection. For the algorithm to work, the following conditions must be met:

- Fonts chosen for Arabic text spans shall cover all positional variants for glyphs used (You can specify a list of fonts. Glyphs will be searched in all of them, following the usual rules for processing of multiple font families).
- Positional variants are accessible through their Unicode codepoints.

This is the case for most TrueType fonts that support Traditional Arabic; however, XEP does not work with Simplified Arabic fonts.

Appendix C. Supported Fonts

C.1. Supported Fonts

This appendix lists font types currently supported in XEP, and describes the details of their use. The overall structure of font configuration is described in the chapter [Chapter 5, Configuring XEP](#); here, details specific to particular font formats are described.

C.1.1. PostScript Type 1 Fonts

To use a Type 1 font with XEP, it is necessary to obtain an AFM (Adobe Font Metrics) file for the font, and specify the URL to it in the `afm` attribute of the `<font-data>` element. If the font is to be embedded into the resulting PDF or PostScript documents, a font outline file in PFA or PFB format is also needed; its location is specified in the respective attribute of the `<font-data>` — either `pfa` or `pfb`.

Example: suppose we have a metrics file `foobar.afm` and an outline file `foobar.pfb`. Its descriptor in the configuration file should look like this:

```
<font embed="true" subset="true">
  <font-data afm="foobar.afm" pfb="foobar.pfb"/>
</font>
```

If your Type 1 font uses non-standard glyph names, you may need an additional step — custom glyph list registration. This is discussed in more detail in the next section.

PostScript Fonts and Unicode

Type 1 font support in XEP is based on direct mapping of Unicode characters to glyph names. Built-in character codes aren't used in the formatting.

XEP follows Adobe's guidelines for mapping Unicode values to glyph names, as described in the following document: *Unicode and Glyph Names, version 2.3* (http://partners.adobe.com/public/developer/opentype/index_glyph.html). By default, *Adobe Glyph List, version 2.0* (hereinafter, AGL; <http://partners.adobe.com/public/developer/en/opentype/glyphlist.txt>) is used to determine Unicode positions for Type 1 glyphs; AGL is hard coded inside XEP.

If a font includes only glyphs comprised in the AGL and all glyphs are named according to Adobe standards, you need no additional steps to use them in XEP. (This is normally the case with most Latin-based Type 1 fonts). However, some fonts cannot be covered by the AGL:

- Some fonts define glyphs outside the scope of AGL — exotic scripts, custom dingbats, etc.

- Some others give non-standard names to glyphs, e.g. Cyrillic or Armenian fonts from TeX.

With XEP, it is possible to use such fonts, and access characters from them by their regular Unicode values. All you need to do is to write an extension to the Adobe Glyph List, and register in the font descriptor: `glyph-list` attribute of a `<font-data>` element which contains a URL to the extension glyph list. Glyph lists are ascribed to fonts individually: different fonts in your system may use different glyph naming systems.

The syntax of a custom glyph list is as follows:

- Lines starting with '#' are comments.
- Empty lines are ignored.
- Each non-comment & non-empty line contains information about a single glyph.
- Within a line, records are separated by semicolons.
- The first record is the Unicode value — 4 hex digits.
- The second record is the glyph name as used in the AFM file.
- The rest of the line is treated as a comment.

Note: The syntax for the glyph list follows the structure of the previous version of AGL, *Adobe Glyph List 1.2* (<http://www.renderx.com/glyphlist-old.txt>). Unfortunately, the two versions of AGL are not compatible with each other.

Duplicate entries are allowed in glyph lists: you can assign different Unicode values to one and the same glyph, and have more than one glyph point to the same Unicode value.

In a custom glyph list, there is no need to cover all symbols present in the font: only non-standard mappings should be included. All glyphs not found in the glyph list are processed according to AGL 2.0 (hard coded into the formatter).

Given below is a schematic example of a custom glyph list:

```
# Sample Glyph List

0020;space
0021;exclam;EXCLAMATION MARK
...
```

...

...

A registration entry for a font with custom glyph mapping looks like this:

```
<font-data afm="foobar.afm"
          pfa="foobar.pfa"
          glyph-list="foo.glyphs"/>
```

A sample glyph list `IPA.glyphs` can be found in the `fonts/` subdirectory of the distribution. It maps IPA (International Phonetic Association) symbols from OmegaSerifIPA font (borrowed from Omega TeX distribution) to Unicode IPA range where possible; characters not covered by Unicode are placed into the private-use area (range starting from U+E000).

Standard Adobe Fonts

An important kind of Type1 fonts are Adobe standard font families: `Times`, `Helvetica`, `Courier`, `Symbol` and `ZapfDingbats`. They are present in every PDF or PostScript installation, and don't require embedding. The default XEP configuration includes settings for them.

All symbols from these fonts are accessed by Unicode, including `Symbol` and `ZapfDingbats` fonts. For `Symbol`, mapping of Unicode to glyph names is contained in the *Adobe Glyph List, version 2.0* (<http://partners.adobe.com/public/developer/en/opentype/glyphlist.txt>); for `ZapfDingbats`, the mapping is taken from a separate document, also available at the Adobe technical support site: <http://partners.adobe.com/public/developer/en/opentype/zapfdingbats.txt>.

XEP samples include three files where all glyphs available from standard Adobe fonts are listed, with their Adobe glyph names and Unicode values:

- `adobe-standard.fo` lists all glyphs from Roman Extended character set.
- `symbol.fo` lists all glyphs from Symbol character set.
- `zapf-dingbats.fo` lists all glyphs from Zapf Dingbats character set.

C.1.2. TrueType Fonts

TrueType fonts are supported in XEP, with the following limitations:

- These fonts are supported by **PDF generator module only**. PostScript generator can only use Type 1 and OpenType/CFF fonts (except for CID ones).
- XEP can only use Unicode-enabled TrueType fonts, i.e. Those with an internal `cmap` table for mapping glyph IDs to Unicode. Most TrueType fonts (both for Windows and Mac) now satisfy this condition, but not all. A notable exception is `Wingdings` font, commonly found on Windows machines.

XEP supports both standalone TrueType fonts (normally stored in files with a *.ttf extension) and fonts in TrueType Collection files (they normally have a *.ttc extension). To use a standalone TrueType font with XEP, a URL to its font file should be specified in a ttf attribute to the <font-data> element, like in the example below:

```
<font-data ttf="FOOBAR.TTF"/>
```

To access a font from a TrueType Collection file, a URL to its font file should be specified in a ttc attribute to the <font-data> element. Additionally, it is necessary to specify the subfont number in a subfont attribute, such as in the example below:

```
<font-family name="Gungsuh" embed="true">
  <font><font-data ttc="batang.ttc" subfont="3"/></font>
</font-family>
```

C.1.3. OpenType/CFF Fonts

OpenType/CFF fonts fall into two groups, depending on whether the CFF font inside them is CID-based. Their level of support in XEP is different.

- Non-CID OpenType fonts are supported by both PDF and PostScript generators (Level 3 only).
- CID-based OpenType fonts are supported by **PDF generator only**. These fonts are mostly used for languages with ideographic scripts, like Chinese, Japanese and Korean. They appear in Asian font packs for Adobe Acrobat; XEP can produce documents that can be viewable by users who have these font packs installed.

To use an OpenType font with XEP, an URL to its font file should be specified in an otf attribute to the <font-data> element, such as in the example below:

```
<font-data otf="KozMinPro-Regular-Acro.otf"/>
```

C.1.4. Supported AFP Fonts

Native AFP fonts are supported in XEP 4.13, with the following limitations:

- These fonts are supported by AFP generator module only.
- Only raster AFP fonts are supported so far.
- Embedding, subsetting and algorithmic slanting of native AFP fonts are not supported.

Native AFP Fonts are Non-CID OpenType fonts described in F:OCA specifications. Usually, they are uploadable to AFP printers. In order to retrieve their metrics, the font files must be located via URL, and the file(s) mapped within XEP configuration file. Please refer [AFP Fonts](#) section for more details on how to configure XEP to use this kind of fonts.

XEP distribution does not include any native AFP fonts.

Appendix D. Supported Graphic Formats

D.1. Supported Graphic Formats

D.1.1. Bitmap Graphics

XEP supports the following raster graphics formats:

- PNG
- JPEG
- GIF
- TIFF

Bitmap graphic that have no built-in resolution or dimension data, default to a resolution of 120 dpi (5 dots of a 600-dpi printer) as prescribed by the CSS2 Spec. This is always the case for GIF images, but may also occur with other image types. The XSL Recommendation suggests using 0.28 mm as a pixel size in such cases, which corresponds to 90 dpi resolution. A smaller pixel size gives better print results because the proportion between pixel size and page width is similar to that of a computer screen. With lower resolutions, often the large GIF/JPEG images fit onto a screen but not into the printable area on the page. For interoperability with other XSL-FO implementations, it is advisable to specify image size explicitly in XSL-FO code.

PNG

XEP recognizes all types of PNG images described in the PNG specification, and reproduces them with the following limitations:

- Alpha channel is completely ignored — sample values are not adjusted by the alpha.
- 16-bit component colors are trimmed down to 8-bit.

Single-color transparency is supported in PDF output only. For indexed-color images with alpha, the first completely transparent color in the palette is used.

Note: Combining single-color transparency with 16-bit color is not safe in XEP because of color depth reduction and consequent merging of adjacent colors.

If the image has an explicit gamma, it is corrected to the sRGB value of 2.2.

JPEG

Grayscale, RGB, and CMYK JPEGs are supported. Data stream is copied directly from the image file to the resultant PDF or PostScript, so there is no additional loss of quality.

For CMYK JPEGs, XEP analyzes the contents of APP14 marker. If the marker indicates that the image is created by Adobe, color polarity is inverted: 0 means "full colorant". Otherwise, standard CMYK conventions apply: 0 is treated as "no colorant".

GIF

XEP supports both interlaced and non-interlaced GIF images and includes implementation of LZW algorithm.

GIF transparency is supported in PDF output.

TIFF

XEP supports the following principal TIFF flavors:

- File organization - strip-based or tiled
- Color model - monochrome, grayscale, RGB or CMYK
- Compression type - uncompressed, CCITT Fax (monochrome images only), PackBits or LZW

TIFF images with separate color planes (`PlanarConfiguration=2`) and/or associated alpha channel (`ExtraSamples=1`) are not supported.

D.1.2. Vector Graphics

XEP supports the following vector graphics formats:

- SVG
- PDF (**PDF generator only**)
- EPS (**PostScript generator only**)

SVG

XEP supports a subset of [Scalable Vector Graphics](#), version 1.1. SVG images can be either referenced as external files (in `src` and `background-image` attributes) or directly embedded into the XSL-FO flow through `<fo:instream-foreign-object>` wrapper.

XEP implements the following SVG elements:

- structure elements - `<svg>`, `<g>`, `<defs>`, `<use>`, `<symbol>`, `<image>`
- styling - `<style>`
- shapes - `<rect>`, `<circle>`, `<ellipse>`, `<polygon>`, `<polyline>`, `<path>`

- basic clipping - `<clipPath>` (see limitations below)
- text - `<text>`, `<tspan>`, `<tref>`
- conditional processing - `<switch>`

The following SVG properties are supported:

- `baseline-shift`
- `clip-path` (see below for limitations on clipping support)
- `color`
- `fill`
- `fill-opacity`
- `fill-rule`
- `font`
- `font-family`
- `font-size`
- `font-stretch`
- `font-style`
- `font-weight`
- `letter-spacing`
- `marker`
- `marker-end`
- `marker-start`
- `marker-mid`
- `stroke`
- `stroke-width`
- `stroke-linecap`
- `stroke-linejoin`
- `stroke-miterlimit`

- `stroke-dasharray`
- `stroke-dashoffset`
- `stroke-opacity`
- `text-anchor`
- `transform`
- `visibility`
- `word-spacing`
- `xml:base`
- `xml:space`

Notes on SVG support in XEP:

1. Color treatment for SVG follows the same rules as for XSL-FO. In particular, `#CMYK`, `#Grayscale`, `#SpotColor` and `#Registration` pseudo profile names can be used in `icc-color()` function to produce CMYK, grayscale, spot, or registration colors.
2. For an SVG image to be processed by XEP, it must have an intrinsic size. If `height` or `width` are expressed in percents, a `viewBox` attribute must be present: the intrinsic size is determined by the `viewBox`, assuming 1 user space unit = 1 pixel.
3. Animation-related elements and attributes are ignored. All objects are drawn at their specified static positions; no attempt is made to reconstruct the initial state of an animated picture.
4. The `clip-path` attribute is not supported on the elements inside `<clipPath>` element and on the `<clipPath>` element itself.
5. Remote references to `clipPath` and `marker` elements are unsupported: only the fragment identifier is used to retrieve them. (Remote links in `use` elements are supported).
6. Character-by-character placement and rotation in text elements are not supported. If an array is used in `x`, `y`, `dx`, `dy`, or `rotate` attributes of `<text>` or `<tspan>` element, only the first number is considered.
7. Bidi reordering and Arabic glyph shaping does not work in SVG text.
8. `xml:base` attribute works only when resolving relative URLs for external images via `<image>` element. It is ignored in `<use>`, `<tref>` and similar elements.
9. XEP supports `fill-opacity` and `stroke-opacity` attributes. Because of the output format limitations, these features are only supported in the PDF generator, and only if the PDF version is set to 1.4 or higher.

10. XEP supports SVG styling via embedded CSS stylesheets (`<style>` element, `style` and `class` attributes). CSS support is limited to Level 1: only ancestor, class, and ID selectors are recognized. Pseudo classes and pseudo elements are not supported.

PDF

PDF images are supported **in PDF generator only**. XEP embeds the first page of a PDF document as a vector image. All related resources (fonts, images, color profiles) are transferred to the output file. Annotations (text notes, hyperlinks, etc) are dropped.

Any unencrypted PDF document which conforms to PDF 1.3 can be embedded as an image, provided that it does not mix LZW and non-LZW compression for parts of the same content stream.¹

EPS

EPS images are supported **in PostScript generator only**. In the PDF generation module, they are replaced by a bitmap preview image (EPSI or TIFF) if available; otherwise, the corresponding area is left blank.

¹ This possibility is purely theoretical: chances that an application uses different compression methods for parts of the same stream are virtually zero.

Appendix E. XEP Intermediate Output Format Specification

E.1. XEP Intermediate Output Format Specification

This section describes the XEP intermediate output format — an XML based representation of the layout that is passed from the generator to the final output generators (PDF, PostScript, etc). All elements reside in a separate namespace, <http://www.renderx.com/XEP/xep> (omitted from the description for brevity). All lengths are measured in units of 0.001 pt (1/72,000 inch), and expressed as integers. The format is represented by the following DTD fragment:

```
<!ENTITY % drawables
    " rotate
    | translate
    | word-spacing
    | letter-spacing
    | font-stretch
    | font
    | text
    | line
    | image
    | rgb-color
    | cmyk-color
    | spot-color
    | registration-color
    | rectangle
    | clip
    | polygon
    | target
    | internal-link
    | external-link
    | internal-bookmark
    | external-bookmark">

<!ELEMENT document (page+)>
<!ATTLIST document
    creator CDATA #REQUIRED
    initial-destination CDATA #IMPLIED>

<!ELEMENT page (%drawables;)*>
<!ATTLIST page
    width CDATA #REQUIRED
    height CDATA #REQUIRED
```

```
page-number CDATA #REQUIRED
page-id CDATA #REQUIRED>

<!ELEMENT rotate EMPTY>
<!ATTLIST rotate
  phi CDATA #REQUIRED>

<!ELEMENT translate EMPTY>
<!ATTLIST translate
  x CDATA #REQUIRED
  y CDATA #REQUIRED>

<!ELEMENT word-spacing EMPTY>
<!ATTLIST word-spacing
  value CDATA #REQUIRED>

<!ELEMENT letter-spacing EMPTY>
<!ATTLIST letter-spacing
  value CDATA #REQUIRED>

<!ELEMENT font-stretch EMPTY>
<!ATTLIST font-stretch
  value CDATA #REQUIRED>

<!ELEMENT font EMPTY>
<!ATTLIST font
  family CDATA #REQUIRED
  weight CDATA #REQUIRED
  style CDATA #REQUIRED
  variant CDATA #REQUIRED
  size CDATA #REQUIRED>

<!ELEMENT text (line*)>
<!ATTLIST text
  x CDATA #REQUIRED
  y CDATA #REQUIRED
  value CDATA #REQUIRED
  width CDATA #REQUIRED>

<!ELEMENT line EMPTY>
<!ATTLIST line
  x-from CDATA #REQUIRED
  y-from CDATA #REQUIRED
  x-till CDATA #REQUIRED
  y-till CDATA #REQUIRED
```

```
thickness CDATA #REQUIRED
style CDATA #REQUIRED>

<!ELEMENT image EMPTY>
<!--ATTLIST image
  src CDATA #REQUIRED
  base CDATA #IMPLIED
  type CDATA #REQUIRED
  x-from CDATA #REQUIRED
  y-from CDATA #REQUIRED
  scale-x CDATA #REQUIRED
  scale-y CDATA #REQUIRED
  role CDATA #IMPLIED>

<!ELEMENT gray-color EMPTY>
<!--ATTLIST gray-color
  gray CDATA #REQUIRED>

<!ELEMENT rgb-color EMPTY>
<!--ATTLIST rgb-color
  red CDATA #REQUIRED
  green CDATA #REQUIRED
  blue CDATA #REQUIRED>

<!ELEMENT cmyk-color EMPTY>
<!--ATTLIST cmyk-color
  cyan CDATA #REQUIRED
  magenta CDATA #REQUIRED
  yellow CDATA #REQUIRED
  black CDATA #REQUIRED>

<!ELEMENT spot-color EMPTY>
<!--ATTLIST spot-color
  colorant CDATA #REQUIRED
  tint CDATA #REQUIRED
  alt-gray CDATA #IMPLIED
  alt-red CDATA #IMPLIED
  alt-green CDATA #IMPLIED
  alt-blue CDATA #IMPLIED
  alt-cyan CDATA #IMPLIED
  alt-magenta CDATA #IMPLIED
  alt-yellow CDATA #IMPLIED
  alt-black CDATA #IMPLIED>

<!ELEMENT registration-color EMPTY>
```

```
<!ATTLIST registration-color
  tint CDATA #REQUIRED>

<!ELEMENT rectangle EMPTY>
<!ATTLIST rectangle
  x-from CDATA #REQUIRED
  y-from CDATA #REQUIRED
  x-till CDATA #REQUIRED
  y-till CDATA #REQUIRED>

<!ELEMENT clip (%drawables;)*>
<!ATTLIST clip
  x-from CDATA #REQUIRED
  y-from CDATA #REQUIRED
  x-till CDATA #REQUIRED
  y-till CDATA #REQUIRED>

<!ELEMENT polygon (point,point+)>
<!ATTLIST polygon
  x-from CDATA #REQUIRED
  y-from CDATA #REQUIRED>

<!ELEMENT point EMPTY>
<!ATTLIST point
  x-till CDATA #REQUIRED
  y-till CDATA #REQUIRED>

<!ELEMENT target EMPTY>
<!ATTLIST target
  name CDATA #REQUIRED
  x CDATA #REQUIRED
  y CDATA #REQUIRED>

<!ELEMENT internal-link EMPTY>
<!ATTLIST internal-link
  destination-id CDATA #REQUIRED
  destination CDATA #REQUIRED
  destination-x CDATA #REQUIRED
  destination-y CDATA #REQUIRED>

<!ELEMENT external-link EMPTY>
<!ATTLIST external-link
  destination CDATA #REQUIRED
  show-destination (new | replace) #REQUIRED>
```

```

<!ELEMENT internal-bookmark EMPTY>
<!--ATTLIST internal-bookmark
  label CDATA #REQUIRED
  id CDATA #REQUIRED
  parent-id CDATA #REQUIRED
  destination-id CDATA #REQUIRED
  destination CDATA #REQUIRED
  destination-x CDATA #REQUIRED
  destination-y CDATA #REQUIRED
  collapse-subtree (true | false) #REQUIRED>

<!ELEMENT external-bookmark EMPTY>
<!--ATTLIST external-bookmark
  label CDATA #REQUIRED
  id CDATA #REQUIRED
  parent-id CDATA #REQUIRED
  destination CDATA #REQUIRED
  collapse-subtree (true | false) #REQUIRED
  show-destination (new | replace) #REQUIRED>

```

The following table provides a detailed description of the output elements:

Table E.1. Output Elements

Element	Description	Attributes
<document>	The root element. At its beginning and at its end, initialization and finalization are usually performed.	<ul style="list-style-type: none"> creator - Information about the application that created the document. initial-destination (optional) - The part of the document that is moved into focus when the document is first opened. There are several other attributes (such as author or title) that transfer unchanged from <rx:meta-field> extension formatting objects (if present) in the source document. Their use is implementation-dependent; for example, PDF generator uses them to fill the fields of the Info object.
<page>	Wraps a single page of the document. There is one <page> element for each page in the document.	<ul style="list-style-type: none"> height - Height of the page width - Width of the page page-id - Page number label page-number - Page number as an ordinal

Element	Description	Attributes
<code><rotate></code>	Rotates the coordinate system.	<ul style="list-style-type: none"> <code>phi</code> - Rotation angle, in degrees. May take values in multiples of 90: 0, 90, 180, 270 degrees.
<code><translate></code>	Shifts the origin of the coordinate system.	<ul style="list-style-type: none"> <code>x</code> - Horizontal shift distance <code>y</code> - Vertical shift distance
<code><word-spacing></code>	Sets word spacing (additional spacing between words).	<ul style="list-style-type: none"> <code>value</code> - The value of word spacing
<code><letter-spacing></code>	Sets letter spacing (additional spacing between non-space characters).	<ul style="list-style-type: none"> <code>value</code> - The value of letter spacing
<code><font-stretch></code>	Sets font-stretch factor.	<ul style="list-style-type: none"> <code>value</code> - The value of font-stretch factor
<code></code>	Changes the current font.	<ul style="list-style-type: none"> <code>family</code> - Font family <code>weight</code> - Font weight (100 to 900) <code>style</code> - Font style (normal, italic, oblique, or backslant) <code>variant</code> - Font variant (normal or small-caps) <code>size</code> - Font size
<code><text></code>	Prints a character string.	<ul style="list-style-type: none"> <code>x</code> - Horizontal position of the initial point on the baseline. <code>y</code> - Vertical position of the initial point on the baseline. <code>value</code> - The text string to print. <code>width</code> - Text width.
<code><line></code>	Draws a line.	<ul style="list-style-type: none"> <code>x-from</code> - Horizontal position of initial point. <code>y-from</code> - Vertical position of initial point. <code>x-till</code> - Horizontal position of final point. <code>y-till</code> - Vertical position of final point.

Element	Description	Attributes
		<ul style="list-style-type: none"> <code>thickness</code> - The thickness of the line. <code>style</code> - The style of the line.
<code><image></code>	Embeds an external image.	<ul style="list-style-type: none"> <code>src</code> - The source URL of the image. <code>base</code> - The base directory to resolve hrefs in the image. <code>type</code> - The image MIME type. <code>x-from</code> - Horizontal position of the lower left corner of the image. <code>y-from</code> - Vertical position of the lower left corner of the image. <code>scale-x</code> - Horizontal scaling factor. <code>scale-y</code> - Vertical scaling factor. <code>role</code> - Alternate description for accessible PDFs.
<code><gray-color></code>	Sets the current color for stroking and filling. The color is chosen in a grayscale color space.	<ul style="list-style-type: none"> <code>gray</code> - Gray color, value: 0 to 1
<code><rgb-color></code>	Sets the current color for stroking and filling. The color is chosen in an RGB color space (additive).	<ul style="list-style-type: none"> <code>red</code> - Red color, value: 0 to 1 <code>green</code> - Green color, value: 0 to 1 <code>blue</code> - Blue color, value: 0 to 1
<code><cmym-color></code>	Sets the current color for stroking and filling. The color is chosen in CMYK color space (subtractive).	<ul style="list-style-type: none"> <code>cyan</code> - Cyan color, value: 0 to 1 <code>magenta</code> - Magenta color, value: 0 to 1 <code>yellow</code> - Yellow color, value: 0 to 1 <code>black</code> - Black color, value: 0 to 1
<code><spot-color></code>	Sets the current color for stroking and filling. The color	<ul style="list-style-type: none"> <code>colorant</code> - Colorant name <code>tint</code> - Color intensity, value: 0 to 1

Element	Description	Attributes
	is chosen in a spot color space (subtractive).	<ul style="list-style-type: none"> alt-gray - Alternative gray color, value: 0 to 1 alt-red - Alternative red color, value: 0 to 1 alt-green - Alternative green color, value: 0 to 1 alt-blue - Alternative blue color, value: 0 to 1 alt-cyan - Alternative cyan color, value: 0 to 1 alt-magenta - Alternative magenta color, value: 0 to 1 alt-yellow - Alternative yellow color, value: 0 to 1 alt-black - Alternative black color, value: 0 to 1 <p>To describe an alternate color, one of the following attribute sets must be present:</p> <ul style="list-style-type: none"> alt-gray — The default color is grayscale. alt-red, alt-green, alt-blue — The default color is RGB. alt-cyan, alt-magenta, alt-yellow, alt-black — The default color is CMYK.
<registration-color>	Sets the current color for stroking and filling. The color is chosen in registration color space (subtractive): it appears on all separations in the document.	<ul style="list-style-type: none"> tint - Color intensity, value: 0 to 1
<rectangle>	Draws a filled rectangle.	<ul style="list-style-type: none"> x-from - The horizontal position of the lower left corner. y-from - The vertical position of the lower left corner. x-till - The horizontal position of the upper right corner. y-till - The vertical position of the upper right corner.

Element	Description	Attributes
<code><clip></code>	Sets the clipping area.	<ul style="list-style-type: none"> <code>x-from</code> - The horizontal position of the lower left corner. <code>y-from</code> - The vertical position of the lower left corner. <code>x-till</code> - The horizontal position of the upper right corner. <code>y-till</code> - The vertical position of the upper right corner.
<code><polygon></code>	Draws a filled polygon. All verticals but the first one are specified in the contained <code><point></code> elements.	<ul style="list-style-type: none"> <code>x-from</code> - The horizontal position of the first vertex. <code>y-from</code> - The vertical position of the first vertex.
<code><point></code>	Adds a vertex to a polygon.	<ul style="list-style-type: none"> <code>x-till</code> - The horizontal position. <code>y-till</code> - The vertical position.
<code><target></code>	Sets the endpoint for an internal destination.	<ul style="list-style-type: none"> <code>name</code> - Internal destination name (id of the element that created the target). <code>x</code> - Horizontal position. <code>y</code> - Vertical position.
<code><internal-link></code>	Specifies an internal link destination.	<ul style="list-style-type: none"> <code>destination-id</code> - Name of the target endpoint. It should match the <code>name</code> attribute of a <code><target></code> element somewhere in the document. <code>destination</code> - The page number to point the link to. <code>x-destination</code> - Horizontal position of the destination point. <code>y-destination</code> - Vertical position of the destination point.
<code><external-link></code>	Specifies an external link destination.	<ul style="list-style-type: none"> <code>destination</code> - a URL <code>show-destination</code> - Controls whether to create a new window when jumping to the link target.

Element	Description	Attributes
<code><internal-bookmark></code>	Specifies an internal bookmark destination.	<ul style="list-style-type: none"> <code>label</code> - Bookmark text. <code>id</code> - Bookmark ID - a positive integer. <code>parent-id</code> - ID of the parent bookmark, or 0 if the bookmark is a top-level element. <code>destination-id</code> - Name of the target endpoint. It should match the <code>name</code> attribute of a <code><target></code> element somewhere in the document. <code>destination</code> - Page number to point the link to. <code>x-destination</code> - Horizontal position of the destination point. <code>y-destination</code> - Vertical position of the destination point. <code>collapse-subtree</code> - Initial state (collapsed or expanded).
<code><external-bookmark></code>	Specifies an external bookmark destination.	<ul style="list-style-type: none"> <code>label</code> - Bookmark text. <code>id</code> - Bookmark ID - a positive integer. <code>parent-id</code> - ID of the parent bookmark, or 0 if the bookmark is a top-level element. <code>destination</code> - A URL. <code>collapse-subtree</code> - Initial state (collapsed or expanded). <code>show-destination</code> - Controls whether to create a new window when jumping to the link target.

Processing instructions may appear in the output. They are taken from the source file and passed straight to the generator with no modification. Processing instructions placement meets the following conditions:

- Instructions placed before `<fo:root>` element in the source file are reproduced at the very top, before the root `<document>` element.
- Instructions placed inside an `<fo:simple-page-master>` element are reproduced on each page generated using that master, immediately after the opening tag of the `<page>` element.

All other processing instructions may vanish during formatting. Except for those specified above, ordering of instructions is not preserved.

Appendix F. Accessibility Support in XEP

F.1. Accessibility Support in XEP

XEP can create accessible PDF documents that are in compliance with *Section 508 Standards* (<http://www.section508.gov>)

This feature is controlled by the `ENABLE_ACCESSIBILITY` core option in the configuration file.

Note: Accessibility support is applied to PDF documents only.

The major requirements for accessible PDF documents are the following:

- Logical reading order
- Alternate text descriptions for images
- Document language

All images in an accessible PDF document should contain alternate descriptions. An alternate text description for images are set by virtue of the `role` attribute in XSL-FO. For example,

```
<fo:external-graphic src="..." role="the alternate description for the image"/>
```

An accessible PDF document should include the document's default language which applies to all text in a PDF document. A document's default language should be specified on `fo:root` element using `xml:lang` attribute. You can change a language on descendant elements by overriding the document's language.

Note: The syntax for the document's default language is the same as for the `xml:lang` attribute.

XEP automatically creates document's logical structure by generating a **tagged** PDF.

F.1.1. Tagged PDF

XEP creates a tagged PDF with a logical structure derived from the structure of the input XSL-FO document.

Note: By default, Adobe Acrobat reads a tagged PDF according to its logical structure, which coincides with the order of the input XSL-FO document. To change the reading order to be in accordance with a visual content, choose the 'Left-to-right, top-to-bottom reading order' from the 'Reading order' combo box and select the 'Override the reading order in tagged documents' check box in the 'Edit->Preferences->Reading' dialog box.

All the elements of the input source will generate a structure item of one of the standard types in the resulting tagged PDF.

Note: Adobe Acrobat's 'Accessibility check' plug-in fails if some table cells do not have a table row as the parent. Make sure that all table cells in your XSL-FO file are within table rows.

Appendix G. List of Output Generators' Options

G.1. List of Output Generators' Options

Feature	Option Name	Default Value	Output Format
Document Security (PDF)	OWNERPASSWORD	null	PDF
Document Security (PDF)	USERPASSWORD	null	PDF
Document Security (PDF)	USERPRIVILEGES	annotate	PDF
PDF Version (PDF)	PDF_VERSION	1.4	PDF
Compression of PDF Streams (PDF)	COMPRESS	true	PDF
Linearization (PDF)	LINEARIZE	false	PDF
Unicode Strings in Annotations (PDF, PostScript)	UNICODE_ANNOTATIONS	true	PDF, Post-Script
Treatment of Unused Destinations (PDF, PostScript)	DROP_UNUSED_DESTINATIONS	true	PDF, Post-Script
Initial Zoom Factor (PDF, PostScript)	INITIAL_ZOOM	auto	PDF, Post-Script
PDF Initial View (PDF, Post-Script)	VIEW_MODE	auto	PDF, Post-Script
PostScript Language Level (PostScript)	LANGUAGE_LEVEL	3	PDF, Post-Script
EPS Graphics Treatment (PostScript)	CLONE_EPS	auto	PostScript
Images Treatment in XML Output (XML)	EMBED_IMAGES	false	XML

Appendix H. Configuration File DTD

H.1. Configuration File DTD

This DTD describes the format of XEP configuration file. Namespace nodes and prefixes are omitted for clarity.

```
<!ELEMENT config (options?, fonts, languages?)>
<!ATTLIST config
    xmlns CDATA #IMPLIED
    xml:base CDATA #IMPLIED>

<!ELEMENT options (option | generator-options)+>
<!ATTLIST options
    href CDATA #IMPLIED>

<!ELEMENT generator-options (option*, charsets?)>
<!ATTLIST generator-options
    format CDATA #REQUIRED>

<!ELEMENT option EMPTY>
<!ATTLIST option
    name CDATA #REQUIRED
    value CDATA #REQUIRED>

<!ELEMENT fonts ((font-family | font-group | font-alias)+)>
<!ATTLIST fonts
    xmlns CDATA #IMPLIED
    default-family CDATA #IMPLIED
    embed CDATA #IMPLIED
    subset CDATA #IMPLIED
    xml:base CDATA #IMPLIED
    href CDATA #IMPLIED>

<!ELEMENT font-group (font-family | font-group | font-alias)+>
<!ATTLIST font-group
    label CDATA #IMPLIED
    embed CDATA #IMPLIED
    subset CDATA #IMPLIED
    xml:base CDATA #IMPLIED
    href CDATA #IMPLIED>

<!ELEMENT font-family (font+)>
<!ATTLIST font-family
```

```
name CDATA #REQUIRED
embed CDATA #IMPLIED
subset CDATA #IMPLIED
ligatures CDATA #IMPLIED
codepage-name CDATA #IMPLIED
codepage-file CDATA #IMPLIED
encoding CDATA #IMPLIED
xml:base CDATA #IMPLIED>

<!ELEMENT font (font-data, transform?)>
<!ATTLIST font
    weight CDATA #IMPLIED
    style CDATA #IMPLIED
    variant CDATA #IMPLIED
    embed CDATA #IMPLIED
    subset CDATA #IMPLIED
    ligatures CDATA #IMPLIED
    size CDATA #IMPLIED
    xml:base CDATA #IMPLIED>

<!ELEMENT font-data EMPTY>
<!ATTLIST font-data
    afm CDATA #IMPLIED
    pfa CDATA #IMPLIED
    pfb CDATA #IMPLIED
    glyph-list CDATA #IMPLIED
    ttf CDATA #IMPLIED
    otf CDATA #IMPLIED
    ttc CDATA #IMPLIED
    subfont CDATA #IMPLIED
    charset-name CDATA #IMPLIED
    charset-file CDATA #IMPLIED
    xml:base CDATA #IMPLIED>

<!ELEMENT transform EMPTY>
<!ATTLIST transform
    slant-angle CDATA #IMPLIED>

<!ELEMENT font-alias EMPTY>
<!ATTLIST font-alias
    name CDATA #REQUIRED
    value CDATA #REQUIRED>

<!ELEMENT languages (language+)>
<!ATTLIST languages
```

```

        href CDATA #IMPLIED
        xml:base CDATA #IMPLIED
        default-language CDATA #IMPLIED>

<!ELEMENT language (hyphenation?, font-alias*)>
<!ATTLIST language
        name CDATA #IMPLIED
        codes NMTOKENS #REQUIRED
        xml:base CDATA #IMPLIED>

<!ELEMENT hyphenation EMPTY>
<!ATTLIST hyphenation
        pattern CDATA #REQUIRED
        encoding CDATA #IMPLIED
        xml:base CDATA #IMPLIED>

<!ELEMENT charsets (charset+)>

<!ELEMENT charset ((code-range+ | code-ranges), character-mapping?, codepage) >
<!ATTLIST charset
        name CDATA #REQUIRED>

<!ELEMENT code-ranges (code-range+ ) >

<!ELEMENT code-range EMPTY>
<!ATTLIST code-range
        from CDATA #REQUIRED
        to CDATA #REQUIRED>

<!ELEMENT codepage EMPTY>
<!ATTLIST codepage
        name CDATA #REQUIRED
        ibm-name CDATA #REQUIRED
        forcelatin CDATA #REQUIRED
        desc CDATA #IMPLIED>

<!ELEMENT character-mapping (character+) >

<!ELEMENT character EMPTY>
<!ATTLIST character
        unicode CDATA #REQUIRED
        afp CDATA #REQUIRED
        desc CDATA #IMPLIED>

```


Appendix I. Additional Components

I.1. XEP Connector for jEdit version 2.1

I.1.1. Changes since version 1.*

XEP Connector for jEdit now uses new RenderX XEP API, introduced in XEP 4.0. Since jEdit 4.2 is final, we've updated the loader to the new interface. jEdit plugin now requires jEdit 4.2.

I.1.2. Overview

XEP Connector for jEdit is a set of interface classes that links XEP to jEdit editor (<http://www.jedit.org>). It registers itself as a jEdit plugin, and permits to apply an XSL FO stylesheet to an XML document open in jEdit, producing a PDF document. There is also a preview option.

I.1.3. Terms of use

XEP Connector is a free software, with source code included in the distribution. Permission to copy and modify is hereby granted, with the following condition: any derived work must bear a clear reference to the original product.

I.1.4. Installation

- In order for this module to work, the following software must be installed and properly configured on your computer:
 - Java VM version 1.3 or higher;
 - jEdit version 4.2 or higher;
 - XEP 4.0 or higher.

Write down the locations of installation directories for XEP and jEdit: you will be prompted for these data during setup.

- Make sure that jEdit is not running: close all documents, and quit IDE.
- Run the setup from the jar file, using a Java VM of your choice. Your Java VM must support Java 2, version 1.3 or higher. To run the jar, type the following on the command prompt:

```
java -jar setupJEditPlugin.jar
```

The system will prompt you for the location of XEP and jEdit root directories.

I.1.5. Copyright notices

This module borrows concepts and structure from the XSLT plugin for jEdit by Greg Merrill [<http://plugins.jedit.org/plugins/?XSLT>].

I.2. XEP ANT Task 2.0 User's Guide

I.2.1. Changes since version 1.*

XEP Ant Task now uses the new RenderX XEP API, introduced in XEP 3.7.

I.2.2. Overview

XEP task uses RenderX XEP XSL Processor to format XML documents to a printable format - PDF or PostScript. It requires XEP 3.7 or later be installed and properly activated.

The task can operate either on a single file or on a file set. Input documents are either XSL FO instances, or generic XML files with associated XSLT stylesheets.

Note: Availability of output formats depends on XEP edition. In some editions, PostScript generator is not available.

I.2.3. Configuration

The user must configure XEPTask to use it with *Ant*.

1. Configure XEP Task entry in `build.xml`, using `<taskdef>`. Here is a typical code snippet that is placed at the prolog of `build.xml` to activate XEP Task:

```
<taskdef name="xep" classname="com.renderx.xepx.ant.XEPTask"
        classpath="XEPTask.jar"/>
```

Refer to Ant documentation for details.

2. Create a classpath reference for XEP task. It must include all JARs that XEP uses, plus `XEPTask.jar` itself. A typical classpath entry looks like the following

```
<path id="xep-classpath">
  <fileset dir="C:\XEP\lib">
    <include name="xep*.jar"/>
    <include name="xt.jar"/>
    <include name="saxon.jar"/>
  </fileset>
  <pathelement path="XEPTask.jar"/>
</path>
```

I.2.4. Parameters

Attribute	Description	Required
in	Specifies a single XML document to process.	Either in or a nested <code><fileset></code> element must be available
out	Used with in, specifies the file name for the formatted document.	No
destDir	Used with nested <code><fileset></code> element(s), specifies a destination directory for the formatted documents. For each file in the set, target file name is created by appending its relative path (as specified in <code><fileset></code>) to this directory, and changing file extension to match the selected output format. By default, the formatted documents are stored in the sources' directories.	No
overwrite	<p>True or False. When "true", the task reformats all documents on each invocation. When "false", the documents are only overwritten if either the source or the stylesheet are newer than the output.</p> <p>If the stylesheet is not local, there is no reliable way to determine its modification date. Such stylesheets are treated as if they are never modified.</p> <p>By default, all documents are reformatted.</p>	No
style	XSLT stylesheet is to apply to source XML documents, either a pathname or a URL. If it is not available, input files are assumed to be XSL FO documents.	No
format	<p>Output format. Possible values:PDF, PostScript, XEP.</p> <p>! Format identifiers are case-sensitive!</p>	Yes

XEP Ant Task can be applied:

- to a single file, specified by in attribute;
- to a batch of files, selected using nested `<fileset>` elements.

These modes are mutually exclusive: if `in` attribute is available, the task will process a single file and ignore all nested `<fileset>` specifiers.

If an XSLT stylesheet is set using `style` attribute, the task will apply it to all input files. Without a stylesheet, the task will attempt to format the files as though they are XSL FO documents.

I.2.5. Parameters specified as nested elements

The following nested elements can appear inside of XEP task entry.

`classpath`

Classpath used to load XEP. The user should set it to match existing XEP installation.

`sysproperty`

Java system property `com.renderx.xep.CONFIG` sets the location of XEP configuration file.

`fileset`

Files to process. Multiple nested `<fileset>` elements are allowed.

`param`

XSLT parameters.

Attribute	Description	Required
<code>name</code>	XSL parameter name	Yes
<code>expression</code>	Ant expression assigned to the parameter. Value of this parameter is interpreted as an Ant expression. To pass a text value to the stylesheet, enclose it into single quotes.	Yes

I.2.6. Examples

Note: In all examples below, we assume that a `classpath` entry with `id="xep-classpath"` is available inside of `build.xml`, and that XEP formatter is installed in `C:\XEP`.

Basic case – render an XSL FO document to produce PDF:

```
<xep in="mydocument.fo" out="mydocument.pdf" format="PDF">
  <classpath refid="xep-classpath"/>
  <sysproperty key="com.renderx.xep.CONFIG" value="C:/XEP/xep.xml"/>
</xep>
```

Transform and render a single XML document to PostScript, passing parameters to stylesheet:

```
<xep in="mydocument.xml" out="mydocument.ps"
      style="stylesheets/mystyle.xsl" format="PostScript">
  <classpath refid="xep-classpath"/>
  <sysproperty key="com.renderx.xep.CONFIG" value="C:/XEP/xep.xml"/>
  <param name="date" expression="13-01-2003"/>
  <param name="time" expression="15:33"/>
</xep>
```

Render a set of XSL FO documents to PDF; put formatted documents into the same directories as the source files:

```
<xep format="PDF">
  <classpath refid="xep-classpath"/>
  <sysproperty key="com.renderx.xep.CONFIG" value="C:/XEP/xep.xml"/>
  <fileset dir="./mydocs/src">
    <include name="*.fo"/>
  </fileset>
</xep>
```

Transform and render a set of XSL FO documents to PostScript; pass one parameter to the stylesheet; put formatted documents into a separate directory:

```
<xep destDir="postscript" style="docbook.xsl" format="PostScript">
  <classpath refid="xep-classpath"/>
  <sysproperty key="com.renderx.xep.CONFIG" value="C:/XEP/xep.xml"/>
  <param name="title-color" expression="'red'"/>
  <fileset dir="docbook">
    <include name="*.dbx"/>
  </fileset>
</xep>
```


Index

A

- Accessibility Support, 139
- Acronyms, 8
- Adding XSL Parameters, 16
- AFP, 10, 59
 - Font Mapping, 59
 - Fonts, 59
 - FORMDEF, 67
 - Image Clipping, 60
 - Image Support, 60
 - Images, 60
 - Referencing Images, 60
- AFP Fonts, 35
 - Adding a Font, 35
 - Removing a Font, 36
 - View and Edit, 35
- AFP parameters
 - Convert image to gray, 34
 - Log Level, 34
 - Resolution, 34
 - Select backend, 33
 - Shading pattern resolution, 34
 - Try using TIFF compression, 34
 - Use BC:OCA, 35
 - Use G:OCA, 35
 - Use replicate and trim, 34
 - Use shading patterns, 34
- Algorithmic Slanting, 55

B

- background Image Scaling and Content Type, 109
- Base Path, 26
- Base URI Definition:xml:base, 110
- Basic Terms, 10
- Bidirectionality, 115
- Bitmap Graphics, 121
 - GIF, 122
 - JPEG, 121
 - PNG, 121
 - TIFF, 122

- Bookmarks, 103, 106
- Border and Padding on Regions, 110
- BROKENIMAGE, 41

C

- Cancel Formatting, 17
- Change Bars, 104, 109
- Color Specifiers, 100
- Command Line, 10
 - AFP, 59
 - Arguments, 21
 - Examples, 22
 - Options, 19
 - Running XEP, 19
 - Switches, 20
- Compression of PDF Streams, 49, 141
- Configuration, 10
 - Backends, 27
 - Fonts, 37
 - Languages, 36
 - Main Settings, 25
- Configuration File, 39
 - Configuring Output Formats, 42
 - Core Options, 40
 - Font Aliases, 57
 - Font Groups, 57
 - Fonts, 53
 - Fonts and Font Families, 53
 - Algorithmic Slanting, 55
 - Embedding and Subsetting Fonts, 54
 - Initial Encoding, 56
 - Ligaturization, 56
 - Languages, 57
 - Hyphenation, 57
 - Language-Specific Font Aliases, 58
- Output Formats
 - Compressions of PDF Streams, 49
 - Document Security, 49
 - EPS Graphics Treatment, 51
 - ICC Profile, 46
 - Images Treatment in XML Output, 53
 - Initial Zoom Factor, 44
 - Inserting Custom Comments, 52
 - Linearization, 49

- Logical Page Numbering, 45
- Page Command, 52
- Page Device Control, 51
- Page Layout, 45
- PDF Initial View, 44
- PDF Version, 48
- PDF Viewer Preferences, 45
- PDF/A Support, 47
- PDF/X Support, 47
- PostScript Language Level, 51
- Prepress Support, 47
- Treatment of Unused Destinations, 46
- Unicode Strings in Annotations, 43
- Structure, 39
- Configuration File DTD, 143
- Configuration Structure, 39
- Configuration using XEP Assistant, 25
- Configuring Backends, 27
 - AFP Files, 33
 - PDF Files, 27
 - PostScript Files, 30
- Configuring Fonts, 37, 53
- Configuring Hyphenation, 57
- Configuring Languages, 36, 57
- Configuring Main Settings, 25
- Configuring Output Formats, 42
- Configuring XEP via the XEP Configuration File, 39
- Core Options, 40

D

- Default font family, 26
- Default Language, 26
- DISCARD_IF_NOT_VALID, 41
- Document Contents, 7
- Document Information, 105
- Document Outline, 103, 106
- Document Security, 49, 141
 - Owner Password, 50
 - User Password, 50
 - Privilege List, 50

E

- Embedding and Subsetting Fonts, 54

- ENABLE_ACCESSIBILITY, 41
- EPS, 125
- EPS Graphic Treatment, 141
- EPS Graphics Treatment, 51
- Extensions
 - Background Image Scaling and Content Type, 109
 - Base URI Definition:xml:base, 110
 - Border and Padding on Regions, 110
 - Change Bars, 109
 - Document Information, 105
 - Document Outline, 106
 - Floats Alignment, 110
 - Flow Sections, 108
 - Index Entries, 107
 - Index Term Markup, 106
 - Indexes, 106
 - Initial Zoom, 110
 - Last Page Number Reference, 108
 - Multicolumn Footnotes, 111
 - Omitted Initial Header in Tables, 110
 - Transpromo, 112
 - Unique Footnotes, 111
 - Watermark, 111
- Extensions to the XSL 1.0 Recommendation, 105

F

- Floats Alignment, 110
- Flow Sections, 108
- Folio Prefix and Suffix, 104
- Font Aliases, 57
- Font Groups, 57
- Font Mapping, 59
- Fonts, 59
 - Add alias, 39
 - Base Path, 38
 - Delete alias, 39
 - Delete Node, 39
 - Embedded, 38
 - Font Family, 39
 - New Family, 39
 - New Font, 39
 - New Group, 39
 - Subsetting, 38

Fonts and Font Families, 53

Formatting, 9

 Apply stylesheet, 15

 Display With, 16

 Format, 15

 Output File, 15

 Set Resource, 16

 Transformation parameters, 15

Formatting a File, 14

Formatting an XML File, 14

Formatting Objects Supported by XEP, 85

FORMDEF Processing Instructions, 68

FORMDEF Resource, 67

FORMDEF Syntax, 68

G

Generating, 9

Generating a Document with FORMDEF Resource, 67

Generating AFP Documents, 59

Generator Options, 42

GIF, 122

Glyph Shaping, 115

GUI Tool, 13

H

Hyphenation, 114

Hyphenation Patterns, 114

I

ICC Profile, 46

Image Clipping, 60

Image Support, 60

Images, 60

Images Treatment in XML Output, 53, 141

Index Entries, 107

Index Term Markup, 106

Indexes, 103, 106

Initial Destination, 110

Initial Encoding, 56

Initial Zoom Factor, 44, 141

Inserting Custom Comments, 52

Integration, 10

Introduction, 9

J

JPEG, 121

K

KERN, 41

L

Language-Specific Font Aliases, 58

Languages

 Add alias, 37

 Alias, 37

 Codes, 37

 Delete alias, 37

 Encoding, 37

 Font Aliases, 37

 Font Family, 37

 Pattern File, 37

Last Page Number Reference, 104, 108

License, 26

LICENSE, 40

Ligaturization, 56

Line-Breaking Algorithm, 113

Linearization, 49, 141

Linguistic Algorithms, 113

List of Output Generators' Options, 141

Logical Page Numbering, 45

M

Multicolumn Footnotes, 111

N

Notes on Formatting Objects Implementation, 98

O

OMIT_FOOTER_AT_BREAK, 42

Omitted Initial Header in Tables, 110

Opening a File, 13

Opening an Existing XML or XSL-FO File, 13

OpenType/CFF Fonts, 120

Output Generators' Options, 141

Overview, 9

P

- Page Device Control, 51
- Page Labeling, 52
- Page Layout, 45
- PAGE_HEIGHT, 41
- PAGE_WIDTH, 41
- Parsing, 9
- PDF, 10, 125
- PDF Initial View, 44
- PDF parameters
 - Drop unused destination, 28
 - Select backend, 28
 - Set initial view mode, 28
 - Set initial zoom value, 28
 - Set owner password, 29
 - Set user password, 29
 - UNICODE annotations, 28
 - Use PDF compression, 29
 - Use PDF linearization, 29
 - User Privileges, 29
- PDF Version, 48, 141
- PDF Viewer Preferences, 45, 141
- PDF/A Support, 47
- PDF/X Support, 47
- PNG, 121
- PostScript, 10
- PostScript Fonts and Unicode, 117
- PostScript Language Level, 51, 141
- PostScript parameters
 - Clone EPS images, 32
 - Drop unused destination, 30
 - Select backend, 30
 - Select PS Level, 31
 - Set initial view mode, 31
 - Set initial zoom value, 31
 - UNICODE annotations, 30
- PostScript Type 1 Fonts, 117
- Prepress Support, 47
- Processing Instructions, 42, 136

R

- Raster images formats, 60
- Referencing Images, 60
- Rendering an XML File using XEP Assistant, 13

- Right-To-Left, 114

S

- SPOT_COLOR_TRANSLATION_TABLE, 42
- Standard Adobe Fonts, 119
- STRICTNESS, 41
- Support for Right-to-Left Writing Systems, 114
- SUPPORT_XSL1, 41
- Supported Expressions, 99
- Supported Fonts, 117
- Supported Graphic Formats, 121
- SVG, 122
- Switch
 - help, 21
 - hosted, 21
 - quiet, 21
 - valid, 21
 - version, 21

T

- Tagged PDF, 139
- Technical Support, 8
- TIFF, 122
- TMPDIR, 41
- Transpromo, 112
- Treatment of Unused Destinations, 46, 141
- True Type Fonts, 119

U

- Unicode Strings in Annotations, 43, 141
- Unique Footnotes, 111
- Use temp folder, 26

V

- VALIDATE, 40
- Vector Graphics, 122
 - EPS, 125
 - PDF, 125
 - SVG, 122

W

- Watermark, 111

X

XEP AFP Generator, 59

XEP Arguments

- <outfile>, 22

- <output format>, 22

- fo, 21

- format, 22

- param <name=value>, 22

- xep, 21

- xml, 21

- xsl <stylesheet>, 22

- <infile>, 21

XEP Assistant, 9, 13

- Access, 13

- Adding XSL Parameters, 16

- AFP, 59

- AFP Fonts, 35

- Backends Tab, 27

- Cancel Formatting, 17

- Configuration, 25

 - Backend for AFP Files, 33

 - Backend for PDF Files, 27

 - Backend for PostScript Files, 30

- Fonts Tab, 37

- Formatting a File, 14

- Formatting an XML File, 14

- Languages Tab, 36

- Main Tab, 25

- Open, 13

- Open an XML or XSL-FO file, 13

- Opening a File, 13

- Rendering an XML File, 13

XEP Intermediate Output Format Specification, 127

XSL 1.1 Support, 102

- Change Bars, 104

- Document Outline, 103

- Folio Prefix and Suffix, 104

- Indexes, 103

- Last page Number Reference, 104

XSL-FO, 11

XSL-FO Conformance, 85

