

XEP 4.9 Reference for Java

Abstract

This document is a reference manual for XEP rendering engine. It contains a detailed description of supported input and output formats, layout control features, and formatter configuration. In the last chapter, issues specific to Java platform are discussed.

Table of Contents

1. Overview	4
2. XEP Configuration	5
2.1. Configuration Structure	5
2.2. Core Options	6
2.3. Parameters for Output Generators	8
2.4. Fonts Configuration	9
2.4.1. Fonts and Font Families	9
2.4.2. Font Groups	11
2.4.3. Font Aliases	12
2.5. Languages Configuration	12
2.5.1. Hyphenation Configuration	12
2.5.2. Language-Specific Font Aliases	13
3. XSL FO Support	13
3.1. Formatting Objects Supported by XEP 4.9	13
3.2. Formatting Properties Supported by XEP 4.9	16
3.3. Notes on Formatting Objects Implementation	26
3.4. Supported Expressions	27
3.5. Color Specifiers	29
3.6. Extensions to the XSL 1.0 Recommendation	30
3.6.1. Document Information	31
3.6.2. Document Outline (Bookmarks)	31
3.6.3. Indexes	32
3.6.4. Flow Sections	34
3.6.5. Last Page Number Reference	34
3.6.6. Change Bars	35
3.6.7. Background Image Scaling and Content Type	35

3.6.8. Initial Destination	36
3.6.9. Omitted Initial Header in Tables	36
3.6.10. Base URI Definition: xml:base	36
3.6.11. Border and Padding on Regions	36
3.6.12. Floats Alignment	36
3.7. XSL 1.1 Support	37
3.7.1. Document Outline (Bookmarks)	38
3.7.2. Indexes	38
3.7.3. Last Page Number Reference	39
3.7.4. Change Bars	39
4. Output Format Settings	40
4.1. Unicode Strings in Annotations (PDF, PostScript)	40
4.2. Initial Zoom Factor (PDF, PostScript)	41
4.3. PDF Viewer Preferences (PDF, PostScript)	41
4.4. Treatment of Unused Destinations (PDF, PostScript)	42
4.5. ICC Profile (PDF)	43
4.6. PDF/X Support (PDF)	43
4.7. Prepress Support (PDF, PostScript)	43
4.8. PDF Version (PDF)	45
4.9. Compression of PDF Streams (PDF)	45
4.10. Linearization (PDF)	45
4.11. Document Security (PDF)	46
4.12. PostScript Language Level (PostScript)	47
4.13. EPS Graphics Treatment (PostScript)	47
4.14. Page Device Control (PostScript)	48
4.15. Images Treatment in XML Output (XML)	48
5. Supported Graphic Formats	49
5.1. Bitmap Graphics	49
5.1.1. PNG	49
5.1.2. JPEG	50
5.1.3. GIF	50
5.1.4. TIFF	50
5.2. Vector Graphics	50
5.2.1. SVG	51
5.2.2. PDF	53
5.2.3. EPS	54
6. Supported Fonts	54

6.1. PostScript Type 1 Fonts	54
6.1.1. PostScript Fonts and Unicode	54
6.1.2. Standard Adobe Fonts	56
6.2. TrueType Fonts	57
6.3. OpenType/CFF Fonts	57
7. Linguistic Algorithms	58
7.1. Line Breaking Algorithm	58
7.2. Hyphenation	59
7.3. Support for Right-to-Left Writing Systems	59
7.3.1. Bidirectionality	59
7.3.2. Glyph Shaping	60
8. Accessibility Support in XEP	60
8.1. Tagged PDF	61
9. XEP on Java Platform	61
9.1. Software Prerequisites	61
9.2. Contents of the Distribution	62
9.3. XEP Assistant — a GUI shell for XEP	64
9.4. Command-Line Interface to XEP 4.9	64
9.5. Command-Line Interface to XSL-FO Validation	66
9.6. Resolution of External Entities and URIs	67
A. List of Output Generators Options	67
B. Configuration File DTD fragment	68
C. XEP Intermediate Output Format Specification	70
D. List of Warning and Error Messages	86
E. XEP AFP Generator	98
E.1. Introduction	98
E.2. Generated AFP Documents	98
E.3. Starting XEP to generate AFP document	99
E.4. Font mapping	99
E.5. Raster Image Support in XEP AFP Generator	99
E.5.1. Image Clipping	100
E.6. Highlight Color Support In XEP AFP Generator	100
E.7. Graphics Support In XEP AFP Generator	101
E.8. Barcodes Support In XEP AFP Generator	105
E.9. Support for FORMDEF resource	108
E.10. Configuring XEP AFP Generator	112
E.10.1. Configuring Code Pages	112

E.10.2. Configuring fonts	113
E.10.3. Configuring Highlight Color Table	113
E.10.4. Configuring Shading Patterns	114
E.10.5. Other configuration options	116
E.11. Bullets support	118
E.12. International Character Set Support	119
E.13. Limitations of XEP AFP Generator	120
E.14. Frequently Asked Questions	122
E.15. Copyrights	122
F. Additional Components	123
F.1. XEP Connector for jEdit version 2.1	123
F.1.1. Changes since version 1.*	123
F.1.2. Overview	123
F.1.3. Terms of use	123
F.1.4. Installation	124
F.1.5. Copyright notices	124
F.2. XEP ANT Task 2.0 User's Guide	124
F.2.1. Changes since version 1.*	124
F.2.2. Overview	124
F.2.3. Configuration	125
F.2.4. Parameters	125
F.2.5. Parameters specified as nested elements	126
F.2.6. Examples	127

1. Overview

RenderX XEP is a high-quality page layout engine. It takes XML data as input and produces layout descriptions in one of the industry-standard page description languages — PDF, PostScript, or AFP. Input data are expressed in W3C-recommended languages, XSL Formatting Objects and SVG, plus a number of extensions to complement and extend them.

The following topics are covered in this Reference:

- XEP configuration files and parameters;
- supported input and output formats, including RenderX proprietary extensions to standard formats;
- details of support for various graphic formats;

- supported font types, and how to configure them for use with XEP;
- language-related algorithms adopted by XEP.

2. XEP Configuration

2.1. Configuration Structure

The behaviour of XEP formatter is controlled by a single configuration file, that must always be made accessible to the formatter. It stores information about core formatter options, lists fonts available to the formatter, and describes language-specific data. Having a single file that completely defines XEP's behaviour permits easy switching between different XEP configurations, and facilitates environment tuneup.

Methods to locate the configuration file are different on different platforms; please refer to specific platform documentation for details. By default, the formatter looks for file named `xep.xml` in the current directory of the process where it runs.

The configuration file is an XML document in a special namespace: "`http://www.renderx.com/XEP/config`". Its formal grammar (DTD) is given in [Appendix B](#). The root of configuration file should be a `<config>` element. It includes three major subsections:


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

Through the rest of this chapter, we discuss the overall structure of each section in the configuration file. Specific instructions for setting up fonts and output generators are contained in the respective chapters of this Reference.

Some parameters can accept URLs as values. In these cases, the location of 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 the effect of `xml:base` attribute.

Usually, a monolithic configuration file is the most convenient way to store XEP configuration. However, you may need to move parts of configuration into separate files; for example, when font configuration needs to be 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 a `href` attribute.

 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.

XEP core and backend options can also be specified in platform-dependent ways, overriding values set in the configuration file. Please refer to the documentation provided with your version of XEP for details.

2.2. Core Options

The operation of XEP 4.9 is controlled by a number of *options*. An option has a name and an associated value: *name=value*. In the configuration file, an option is defined by an `<option>` element; its `name` attribute defines option's name, and the `value` sets its value. XEP core options are always specified as a direct children of the `<options>` element. The following core options are defined for XEP 4.9:

LICENSE

Location of the license file. XEP looks for a license file at startup, and refuses to run if the signature on the license does not match the public key associated with the specific edition of the formatter. This file is also used as an access key to XEP online update service. The parameter can be specified either as a file name in the local filesystem, or as an URL. In addition to common protocols, `data:` and `resource:` URL schemes are supported.

Default: `license.xml`

VALIDATE

Boolean value (*true/false*). Controls validation of input. In non-validating mode, XEP runs faster and takes less memory; however, less errors are intercepted, and the results of formatting are less predictable for malformed input. We discourage setting this switch to *false* until your stylesheets are thoroughly debugged.

Default: *true*

DISCARD_IF_NOT_VALID

Boolean value (*true/false*). Controls termination of processing upon unsuccessful validation.

Default: *true*

STRICTNESS

Validator strictness level. Valid values are *0* (relaxed), *1* (normal), and *2* (strict).

Default: *1* (normal strictness)


SUPPORT_XSL11

Boolean value (*true/false*). Turns on/off XSL-FO 1.1 support (see [XSL 1.1 Support](#) for details). If the value of this parameter is *false*, XEP runs faster and takes less memory.

Default: *true*

TMPDIR

Path to the directory for temporary files. If set, this parameter must point to a writeable directory, specified either as a path in the local filesystem, or as a file URL. To disable writing temporary files to disk, specify *none* as the value for this option.

 To avoid file name clashes, a separate temporary directory should be specified for each process running XEP.

Default: *none*

BROKENIMAGE

Icon inserted as a replacement for broken or missing images. The parameter can be specified either as a file name in the local filesystem, or as an URL. In addition to the common protocols, *data:* and *resource:* URL schemes are supported.

Default: *images/404.gif*

PAGE_WIDTH

Sets default page width.

Default: *576pt* (8 in)

PAGE_HEIGHT

Sets default page height.

Default: *792pt* (11 in)

KERN

Boolean value (*true/false*). Controls whether the formatter uses or ignores glyph kerning data to determine character positions.

Default: *true*

ENABLE_ACCESSIBILITY

Boolean value (*true/false*). Controls whether the formatter uses a special mode to create accessible PDF documents.

Default: *false*

OMIT_FOOTER_AT_BREAK

Boolean value (*true/false*). Defines whether table footers are omitted at breaks by default.

Default: *false*

SPOT_COLOR_TRANSLATION_TABLE

Path to Spotcolor-to-CMYK translation table file for use in `rgb-icc()` function with `#Spot-Color` pseudoprofile. This table was hardcoded in previous versions of XEP. It is removed from XEP since version 4.6 due to license restrictions. Users are recommended to download this table from [PANTONE® site](http://www.pantone.com/products/products.asp?idArticle=735&idSubArea=0&idArea=1) [http://www.pantone.com/products/products.asp?idArticle=735&idSubArea=0&idArea=1] and specify this option accordingly. The parameter can be specified either as a file name in the local filesystem, or as an URL. In addition to the common protocols, `data:` and `resource:` URL schemes are supported.

Default: *none*. All spot colors will come out black.

2.3. Parameters for Output Generators

Principal output formats for XEP are Adobe Portable Document Format (version 1.3 or higher), or PostScript (Level 2 or 3). Each format is produced by a special module, called *output generator* (sometimes referred to as *backends*). Apart from the three standard generators, there is also a special backend that prints out an XML document corresponding to the stream of drawing events issued by the formatter core. This intermediate XML format is typically used for postprocessing manipulation or debugging through XEP API; it is described in [Appendix C](#).

Each generator has configuration parameters that you can modify to control different aspects of produced documents. PDF generator parameters control PDF security options, PDF stream compression, viewer preferences etc. PostScript generator parameters control PostScript language options and Distiller parameters (via `pdfmark` operators). The only parameter of the XML output generator controls representation of images in the resulting document. All these parameters are thoroughly discussed in the chapter [Output Formats](#).

Default settings for output generators can also be specified inside `<options>` element in the configuration file. To distinguish them from core options, they are wrapped in `<generator-options>` element. Attribute `format` of this element defines target output format for the generator. For example, the following fragment turns on linearization for 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>
```

Configurable parameters for all generators are listed in [Appendix A](#), and discussed in more details in chapter [Output Formats](#).

2.4. Fonts Configuration

Font configuration is specified inside `` 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.

2.4.1. Fonts and Font Families

A *font family* is a group of fonts that share a common design, but differ in stylistic attributes: upright or italic, light or bold, etc. A font family is the basic configuration unit in XEP: individual fonts are bundled into groups by the family they belong to. All data pertinent to one font family is contained in a `<font-family>` element. It bears a required `name` attribute that identifies the font family. Family names should be unique within the configuration file: they are matched against the respective XSL-FO property value. The default font family (the one used when no font family is specified in the input file, or no other family matches) is defined by the `default-family` attribute of the `` element. Its value is a family name that must be present in the file, otherwise a configuration error occurs.

A typical font family descriptor looks like this:

```
<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.


Actual characteristics of the font are specified by a mandatory `<font-data>` element, located inside the ``. Its attributes specify locations of various auxiliary files used by the formatter to retrieve font metric data and outlines, and set additional parameters that may be needed to interpret font data properly. Different attributes are needed for different font types. Supported font types and their configuration requirements are discussed in more details in chapter [Fonts](#).

2.4.1.1. Embedding and Subsetting Fonts

Most fonts can be either *embedded* into the resulting PDF or PostScript document, or specified as external fonts; in the latter case, the resulting file will only be viewable on systems that have the correspondent font configured for use with viewing/printing 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 `embed` attribute of the `` element describing the font.

An embedded font can be *subsetted*: instead of storing the entire font in the document, it is possible to leave 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 `subset` attribute of the `` element.

To provide a more compact notation, 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 on <font-family> will affect all fonts in that family; placing them on <font-group> will set the respective parameters for all fonts in all families in the group (unless overridden on some descendant node), etc.

 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.

Fonts support in AFP Generator is slightly different due to AFP implementation. Please refer [Font Mapping in AFP Generator](#) for more details on fonts support in XEP AFP Generator. You may also find useful information on [Configuring fonts in AFP Generator](#).


2.4.1.2. Algorithmic Slanting

In XEP 4.9, there is a possibility to apply algorithmic slanting to fonts, in order to produce oblique or backslanted versions of fonts that don't 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.

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°.

2.4.1.3. Ligaturization

In XEP 4.9, 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 a Unicode ligature codepoints.

 In XEP 4.9, 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.

2.4.2. 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 `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`: it assigns a name to the group. The name serves only for record keeping: no constraints are imposed on it.

2.4.3. Font Aliases

XEP 4.9 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.

 Aliases always resolve to “real” families, and not to other aliases. Chained alias resolution is not possible in XEP 4.9.

2.5. Languages Configuration

Language-specific configuration parameters are stored in the third major section of the configuration file, inside a `<languages>` element. It contains one or more `<language>` elements; each `<language>` stores information pertaining to a single language. The language is identified by two attributes: `name` attribute provides a name of the language, and `codes` attribute contains a list of codes used to refer to the language in the XSL-FO input data. Codes in the list are separated by spaces.

In XEP 4.9 two kinds of data are configurable in this section of the configuration files: hyphenation patterns and language-specific font aliases.

2.5.1. Hyphenation Configuration

XEP 4.9 uses TeX hyphenation patterns for hyphenation data. Details on hyphenation algorithm are given in a separate chapter, [Hyphenation](#).

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 TeX pattern file. An optional `encoding` attribute specifies the encoding of the pattern file; if it is missing, ISO-8859-1 is assumed.

2.5.2. Language-Specific Font Aliases

Language sections may also contain `<font-alias>` elements, described above in chapter [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.

3. XSL FO Support

This section describes the implementation of XSL Formatting Objects in XEP — an XSL Engine for PDF developed by RenderX, Inc, version 4.9. It lists all supported formatting objects and their properties, provides information about fallbacks for unsupported objects, and discusses details of XSL spec interpretation adopted in the engine.

XEP 4.9 implements *Extensible Stylesheet Language version 1.0* as specified in the *XSL 1.0 Recommendation of October 15, 2001* (<http://www.w3.org/TR/2001/REC-xsl-20011015/>).

3.1. Formatting Objects Supported by XEP 4.9

Legend:

Yes

Element/property is supported.

No

Element/property is not supported yet.

—

Element/property is not applicable (e.g. aural properties).

§	Object Name	Implemented
6.4.2	<code><fo:root></code>	Yes
6.4.3	<code><fo:declarations></code>	No
6.4.4	<code><fo:color-profile></code>	No
6.4.5	<code><fo:page-sequence></code>	Yes
6.4.6	<code><fo:layout-master-set></code>	Yes
6.4.7	<code><fo:page-sequence-master></code>	Yes
6.4.8	<code><fo:single-page-master-reference></code>	Yes

§	Object Name	Implemented
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
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

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

² All contents is placed inline.

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

Formatting Objects Supported by XEP 4.9

§	Object Name	Implemented
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
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

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>.

§	Object Name	Implemented
6.11.4	<fo:retrieve-marker>	Yes

3.2. Formatting Properties Supported by XEP 4.9

§	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
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	-

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

Formatting Properties Supported by XEP 4.9

§	Property Name	Implemented
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
7.7.7	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 When the background image is repeated along an axis, its offset on this axis is ignored.

§	Property Name	Implemented
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
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

Formatting Properties Supported by XEP 4.9

§	Property Name	Implemented
7.10.2	margin-bottom	Yes
7.10.3	margin-left	Yes
7.10.4	margin-right	Yes
7.10.5	space-before	Yes
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

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.

§	Property Name	Implemented
7.14.8	min-height	No ¹²
7.14.9	min-width	No ¹³
7.14.10	scaling	Yes
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

¹² Maps to height.

¹³ Maps to width.

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.

Formatting Properties Supported by XEP 4.9

§	Property Name	Implemented
7.16.5	text-shadow	Yes ¹⁷
7.16.6	text-transform	Yes ¹⁸
7.16.7	treat-as-word-space	No
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

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 locale information in your environment properly.

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.

§	Property Name	Implemented
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
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	-

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.

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.

Formatting Properties Supported by XEP 4.9

§	Property Name	Implemented
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
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

§	Property Name	Implemented
7.26.4	border-end-precedence	Yes
7.26.5	border-separation	Yes
7.26.6	border-start-precedence	Yes
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

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 their content being empty; cells not present 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".

Formatting Properties Supported by XEP 4.9

§	Property Name	Implemented
7.28.1	content-type	Yes
7.28.2	id	Yes
7.28.3	provisional-label-separation	Yes
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

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

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

§	Property Name	Implemented
7.29.17	page-break-before	Yes
7.29.18	page-break-inside	Yes
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

3.3. Notes on Formatting Objects Implementation

<fo:block>

By the spec, 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:bidi-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:bidi-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. PDF/PostScript, as well as AFP, are being intrinsically static formats, none of them is applicable.

<fo:float>

The before-float appears on 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 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 take only plain text inside; all formatting will be lost.

<fo:marker>

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

3.4. Supported Expressions

XEP 4.9 implements a subset of XSL expressions algebra. 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:

1. For compound expressions, the result of evaluation of all intermediate subexpressions should have a valid XSL type. For example, expression `(2in * 2in) div 1in` is not supported because its first subexpression yields dimensionality of square inches that is not a valid XSL unit; while `2in * (2in div 1in)` works.
2. Expressions that require knowledge of layout to evaluate (e.g. block widths expressed in percents) 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 also to `body-start()` and `label-end()` functions.
3. 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.

4. Property value functions that take `start-indent/end-indent` as arguments may work incorrectly if the block with indents is placed into another block that has CSS-style `margin-*` attributes. For safety, we recommend using either expressions with indents, or CSS margins; mixing these two coding styles in the same stylesheet may yield unpredictable results.

3.5. Color Specifiers

XEP 4.9 can produce PDF, PostScript, and AFP output using the following color types:

1. *Grayscale*. The following specifiers produce grayscale color at the 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` pseudoprofile. 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 at the output:

- HTML and SVG predefined names and RGB specifiers 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 at the output:

- `rgb-icc()` function with built-in `#CMYK` pseudoprofile. 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 at the output:

- `rgb-icc()` function with built-in `#SpotColor` pseudoprofile. 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).

 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 not found there, 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)
```

Please refer [Highlight Color Support in AFP Generator](#) section for more details on how spot color is treated with AFP Generator.

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

- `rgb-icc()` function with built-in `#Registration` pseudoprofile. 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)
```

3.6. 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.

3.6.1. 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 recognizes 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.

All other values for name are ignored. The 'Producer' field in the PDF file is set to "XEP 4.9"; there is no means to control it from the source file.

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

3.6.2. Document Outline (Bookmarks)

An often requested feature for PDF rendering component. Implementation uses three extension elements: `<rx:outline>`, `<rx:bookmark>`, and `<rx:bookmark-label>`.

<rx:outline>

Top-level element of the document outline tree. Should be located before any <fo:page-sequence> elements, but after <fo:layout-master-set> and <fo:declarations> (if present). 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 subordinated 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>. Contents of this element should be plain text.

3.6.3. Indexes

Building page number lists for indexes is not possible within XSL 1.0. XEP 4.9 provides this functionality via extension elements/properties.

rx:key

This attribute can be specified on any element that can carry an id (and thus be a target to <fo:page-number-citation>). Its content is a term used to group element references in an index entry. Unlike id, rx:key need not be unique across the document.

<rx:begin-index-range>**<rx:end-index-range>**

These atomic inline-level elements are used to create a range in the index. They can be located anywhere in the document, but should always form a pair: each <rx:begin-index-range> has a required id attribute, and <rx:end-index-range> should have a matching ref-id property. Index term is associated with the range by rx:key attribute on <rx:begin-index-range>.

<rx:page-index>

This element creates a list of page numbers for an index entry. It contains one or more <rx:index-item> children. Page numbers/ranges created by <rx:index-item>s are formatted

in a list, sorted in ascending order with duplicates removed. The element accepts the following property:

list-separator

String used to separate page numbers in the list. Default is comma plus space: ", ". The property is inheritable.

Font and style of the page number list are controlled by standard formatting properties that can be either specified directly on the element, or inherited from ancestor nodes.

<rx:index-item>

This element add references to the list of page numbers created by <rx:page-index>. The element accepts the following properties:

ref-key (required)

Selects elements for the index entry. All elements whose rx:key equals the value of this attribute, and only those, will be selected for processing.

range-separator

String used to separate page numbers that form a continuous range. Default is en dash: "–" (U+2013). The property is inheritable.

merge-subsequent-page-numbers

Controls whether sequences of adjacent page numbers should be merged into ranges. Default is *false*. The property is inheritable.

link-back


Specifies whether to create a hotlink from the page number in the index back to the referenced item. Default is *false*. The property is inheritable.

Inline formatting properties can be specified on <rx:index-item>. They affect formatting of each individual number placed to the list by the element.


The following algorithm of merging page numbers is used:

1. Each <rx:index-item> generates a list of page numbers and ranges. This list is filtered as follows:
 - if there are two or more occurrences of the same page number, the one that points to the point located earlier in the document is preserved;
 - if a page number falls inside a range, the page number is suppressed;

- if two ranges overlap, they are replaced by their union;
 - if `merge-subsequent-page-numbers` is set to `"true"` and there is a subsequence of page numbers and/or ranges that fully covers three or more consecutive pages, then the whole subsequence is replaced by a single range that starts at the first item in the subsequence and ends in the last item.
2. Lists produced by individual `<rx:index-item>` elements are merged. If there are full duplicates (repeated page numbers, or ranges with both ends coincident), the entry from the `<rx:index-item>` element that comes first in document order is retained.

 To specify inheritable extension properties on elements from the standard XSL FO namespace, put them into the extension namespace to avoid validation alerts:

```
<fo:root xmlns:rx="http://www.renderx.com/XSL/Extensions"
  rx:link-back="true" rx:merge-subsequent-page-numbers="true">
```

 In XEP 4.9, `<rx:index-item>` only search elements preceding them in the document. No forward references are implemented. This limitation is likely to persist in future versions of XEP.

3.6.4. Flow Sections

Flow sections are a generalization of blocks with `span="all"`. They permit to split the flow into subflows with different column counts in each subflow. The following element does it:

<rx:flow-section>


This element must be a direct child of `<fo:flow>`; it can be mixed with other block-level elements. It explicitly creates a *span-reference-area* with `column-count` and `column-gap` traits taken from the respective properties of `<rx:flow-section>`.

3.6.5. Last Page Number Reference

This extension element retrieves the number of the last page occupied by a particular element. Its syntax and semantics is 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 we 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.

 This element is described in XSL 1.1 Working Draft of 17 December 2003: <http://www.w3.org/TR/2003/WD-xsl11-20031217/>. In subsequent versions of XEP, it is likely to move to the standard XSL FO namespace.


3.6.6. Change Bars

XEP 4.9 has support for change regions, as described in XSL 1.1 Working Draft of December 16, 2004, <http://www.w3.org/TR/2004/WD-xsl11-20041216/>.

<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 6.13.2 and 6.13.3, respectively. In future versions of XEP, when XSL 1.1 will become W3C Recommendation, they will be moved to the standard XSL-FO namespace.

 The content model for these elements is different from that is described in 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 4.9 implementation does not have such limitations, change bar anchors can be placed almost anywhere in the flow.

3.6.7. Background Image Scaling and Content Type

In XSL 1.0, there is no provision to scale/size a background image. XEP 4.9 implements this functionality via 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).

3.6.8. Initial Destination


This extension permits 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 for `internal-destination` property.

3.6.9. Omitted Initial Header in Tables

This extension permits to omit a table header at the beginning of a table. Such a feature can be used to create "continuation headers", 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`.

3.6.10. Base URI Definition: `xml:base`

XEP recognizes and processes `xml:base` attribute, defined in [XML Base Recommendation](http://www.w3.org/TR/xmlbase/) [http://www.w3.org/TR/xmlbase/]. It permits to set the base for resolving relative URIs (link targets, image locations, fonts, hyphenation patterns, etc) for the whole document or a single subtree.

 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.

3.6.11. 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>`); but the spec says they only may accept values of 0 (*sic!*). In XEP, non-zero values of these properties will result in a border around the respective region area, and its content rectangle will be padded by the specified amount.

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

3.6.12. Floats Alignment

It is often needed to make floating figures float towards the different sides of the page depending on its parity. However in XSL 1.0 Recommendation there are no means to achieve such effect. XEP 4.9 supports two additional values for `float` property of `<fo:float>` element. Those values are: `"inside"` and `"outside"`. Their meaning is the same as in `text-align` property as defined by XSL 1.0 Recommendation: `"inside"` value will align floating block to the inner edge of the page (left for odd pages, right for even pages) and `"outside"` will align 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".

3.7. XSL 1.1 Support

XEP 4.9 includes stylesheets to convert XSL 1.1 elements and attributes to elements and attributes from XSL 1.0 and RenderX extensions. Using these stylesheets XSL 1.1 documents can be converted to XSL 1.0 with RenderX extensions. All elements and attributes which exist in both XSL 1.1 and XSL 1.0 will be left without changes. The elements and attributes which exist only in XSL 1.1 will be converted into corresponding elements and attributes from RenderX extensions. To turn on/off XSL 1.1 support, use XEP configuration option `SUPPORT_XSL11` (see [Core Options](#) for details). The stylesheets support conversion for the following features:

1. Document Outline (Bookmarks)
2. Indexes
3. Last Page Number Reference
4. Change Bars

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

- `fo:page-sequence-wrapper`
- `fo:folio-suffix`
- `fo:folio-prefix`
- `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

XEP 4.9 also provides support for the value 'only' of @page-position.

3.7.1. 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

3.7.2. Indexes

§	XSL 1.1 Object/Property Name	RenderX extensions Object/Property Name
6.10.2	<fo:index-page-number-prefix>	No correspondence, will be ignored
6.10.3	<fo:index-page-number-suffix>	No correspondence, will be 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>

Change Bars

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

3.7.3. 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 will be retrieved.

3.7.4. 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> just mapped to themselves.

4. Output Format Settings

Besides extension elements and attributes, certain properties of output documents can be controlled with *XML processing instructions*. They are used to specify information that does not affect formatting, and can be safely ignored by other XSL-FO processors. In most cases, default values for these parameters can also be specified by generator options in the configuration file; however, some features affect only parts of the input document, and can only be expressed with processing instructions.

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

4.1. Unicode Strings in Annotations (PDF, PostScript)

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

These processing instructions enable or disable use of Unicode to represent PDF annotation strings (bookmark text, document info, etc). In PostScript, the information is coded in pdfmark operators, and used for further conversion to PDF. Permitted values are:

true

Enable use of 16-bit Unicode to represent annotation strings. In this mode, XEP uses 8-bit PDFEncoding for strings that can be represented in AdobeStandard character set, and 16-bit Unicode for strings containing characters not in AdobeStandard. **This is the default behaviour.**

false

Disable use of Unicode. Annotations are always represented in 8-bit PDFEncoding; characters outside the AdobeStandard set are replaced by bullet symbols. This option may be used to enforce compatibility with older versions of PDF software that don't support Unicode, e.g. Adobe Acrobat 3.0.

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

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

4.2. Initial Zoom Factor (PDF, PostScript)

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

These processing instructions specify magnification factor when the PDF document is first opened in a PDF viewer. In PostScript, the information is coded in pdfmark operators, and used for further conversion to PDF. Permitted values are:

auto

Page scaling is not specified in the document; it is left to the viewer. **This is the default behaviour.**

fit

The page is scaled to fit completely into the viewport.

fit-width

The page is scaled so that its width matches the width of the viewport.

fit-height

The page is scaled so that its height matches the height of the viewport.

number or percentage

The page is scaled by the specified factor.

The instruction should be placed at the top of the document, before or right after the <fo:root> start tag.

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

4.3. PDF Viewer Preferences (PDF, PostScript)

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

These processing instructions specify the viewer mode to open the PDF document. In PostScript, the information is coded in pdfmark operators, and used for further conversion to PDF. Permitted values are:

auto

Show the bookmarks pane if there are bookmarks in the document; otherwise, hide all auxiliary panes. **This is the default mode.**

show-none

Hide all auxiliary panes.

show-bookmarks

Show the bookmarks pane.

show-thumbnails

Show the thumbnails pane.

full-screen

Show the document in full-screen mode.

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

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

4.4. 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 about named destination is coded in `pdfmark` operators, and used for further conversion to PDF. Permitted values are:

true

Named destinations are created only for objects used as targets in `internal-destination` attributes. **This is the default.**

false

Named destinations are created for all objects having an `id` attribute.

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

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

4.5. ICC Profile (PDF)

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

These processing instructions specify a characterized printing condition. PDF/X 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(.....)`.

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

4.6. PDF/X Support (PDF)

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

This processing instruction sets PDF/X compliance level. Permitted values are:

none

No PDF/X restrictions are applied. **This is the default.**

pdf-x-1a

Sets PDF/X-1a compliance level. The resulting PDF will comply to the PDF/X-1a:2001 spec.

pdf-x-3

Sets PDF/X-3 compliance level. The resulting PDF will comply to the PDF/X-3:2001 spec.

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

4.7. Prepress Support (PDF, PostScript)

These processing instructions define features that support 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 correspondent 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 correspondent 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(.....)`.

To set prepress options for all pages in a document, the respective instruction should appear at the top of the document, before `<fo:root>` element. To control prepress settings for a single page, the instructions should be specified inside the `<fo:simple-page-master>` object used to generate that page.

4.8. PDF Version (PDF)

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

This processing instruction sets target PDF version. Permitted values are 1.3, 1.4, and 1.5. **The default value is 1.4.** When set to 1.3, advanced features of PDF 1.4 are disabled: in SVG images, fill-opacity and stroke-opacity properties will be ignored.

The instruction should be placed at the top of the document, before or right after the <fo:root> start tag.

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

4.9. Compression of PDF Streams (PDF)

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

This processing instruction controls compression of content streams in PDF. Permitted values are:

true

PDF streams are compressed using Flate algorithm. **This is the default.**

false

PDF streams are not compressed. This option may be useful for debugging purposes.

The instruction should be placed at the top of the document, before or right after the <fo:root> start tag.

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

4.10. Linearization (PDF)

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

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

true

PDF is linearized. Use this option to prepare documents for publication on WWW sites.

false

PDF is not linearized. **This is the default.**

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.


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

4.11. Document Security (PDF)

A group of processing instructions controls PDF security settings.

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

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 permissions.

 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?>
```

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 will be authorized.

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

Sets the default privilege list for users accessing the resulting 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 & images from the document into clipboard;

annotate

enables adding annotations to the document and changing form field values.

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

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


If the user password is not empty 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 not empty and the user password is not set, the resulting 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.

By default, neither of the passwords is set (security disabled). Default privilege list is *annotate*.

These instructions should be placed at the top of the document, before or right after the `<fo:root>` start tag.

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

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

4.12. PostScript Language Level (PostScript)

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

This processing instruction sets target PostScript language level. Permitted values are 2 and 3. **The default value is 3.** When language level is set to 2, some advanced features and font flavors are not available.

The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

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

4.13. 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 simply by pasting their contents at each occurrence. Permitted values are:

true

EPS graphics are pasted into the output stream at each occurrence. This method is 100% safe, but it may lead to substantial growth of the resulting file size if an image is used many times.

This is the default mode.

false

EPS graphics are put into a PostScript form. This method saves space, but some EPS images (namely, documents that make use of `currentfile` operator) cannot be processed this way: the resulting PostScript code may be corrupt.


The instruction should be placed at the top of the document, before or right after the `<fo:root>` start tag.

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

4.14. 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.

 XEP does not spellcheck neither the entry name nor the value supplied in this instruction. Wrong code passed with this option can invalidate the whole output file. Be careful!

To set page device options for the whole document, the respective instructions should appear at the top of the document, before `<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.

4.15. Images Treatment in XML Output (XML)

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


This parameter controls if XML output generator should embed external image referenced in the document in the resulting document instance as Base64 strings. Values have the following meaning:

true

All images are stored inside the resulting file, using data: URL scheme.

false

Do not embed images. In the generated XML file, images are referenced by their original URLs. **This is the default mode.**

The instruction should be placed at the top of the document, before or right after the <fo:root> start tag.

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

5. Supported Graphic Formats

5.1. Bitmap Graphics

XEP 4.9 supports the following raster graphics formats: PNG, JPEG, GIF, and TIFF.


When a bitmap graphic has no built-in resolution or dimension data, its resolution defaults to 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 in other image types. The XSL Recommendation suggests, though not mandates, using 0.28 mm as a pixel size in such cases; this corresponds to 90 dpi resolution. In our opinion, a smaller pixel size gives better print results: the proportion between pixel size and page width is similar to that of a computer screen. With lower resolutions, it often happens that 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.

5.1.1. PNG

XEP 4.9 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.

 Combining single-color transparency with 16-bit color is not safe in XEP 4.9 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.

5.1.2. 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".

5.1.3. GIF

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

GIF transparency is supported in PDF output.

5.1.4. 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.

5.2. Vector Graphics

XEP 4.9 supports the following vector graphics formats: SVG, PDF (*PDF generator only*), EPS (*PostScript generator only*).

5.2.1. SVG

XEP supports a subset of [Scalable Vector Graphics](http://www.w3.org/Graphics/SVG) [http://www.w3.org/Graphics/SVG], 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.

In version 4.9, 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 below about limitations);
- text: `<text>`, `<tspan>`, `<tref>`;
- conditional processing: `<switch>`.

The following SVG properties are supported:

- `baseline-shift`
- `clip-path` (see below about 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 4.9:

1. Color treatment for SVG follows the same rules as for XSL FO. In particular, `#CMYK`, `#Grayscale`, `#SpotColor`, and `#Registration` pseudoprofile 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. Otherwise, the image cannot be used with XEP.
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 is 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 PDF generator, and only if 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.

5.2.2. 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 conformant to PDF 1.3 can be embedded as image, provided that it does not mix LZW and non-LZW compression for parts of the same content stream.¹

5.2.3. 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 correspondent area is left blank.

6. Supported Fonts

This chapter lists font types currently supported in XEP 4.9, and describes details of their use in XEP. The overall structure of font configuration is described above, in the chapter [Fonts Configuration](#); here, we give details specific to particular font formats.

6.1. PostScript Type 1 Fonts

To use a Type 1 font with XEP, it is necessary to obtain an AFM file (Adobe Font Metrics) for the font, and specify the URL to it in the `afm` attribute of the `<font-data>` element. If the font is 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.

6.1.1. 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.

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

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 hardwired 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 AGL reach — exotic scripts, custom dingbats, etc.;
- some other 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 is to write an extension to the Adobe Glyph List, and register in the font descriptor: `glyph-list` attribute of a `<font-data>` element will contain 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;
- the rest of the line is treated as a comment.

 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, two versions of the AGL are not compatible between themselves.

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 will be processed according to the AGL 2.0 (hardwired 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 will look 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).

6.1.2. Standard Adobe Fonts

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

All symbols from these fonts should be 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.

6.2. TrueType Fonts

TrueType fonts are supported in XEP 4.9, 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 4.9 can only use Unicode-enabled TrueType fonts, i.e. those with an internal cmap table for mapping glyph IDs to Unicode. Most TrueType fonts 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, like in the example below:

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

6.3. 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 4.9 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, a URL to its font file should be specified in a `otf` attribute to the `<font-data>` element, like in the example below:

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

7. Linguistic Algorithms

7.1. Line Breaking Algorithm

XEP uses the following line breaking algorithm:

1. Line break is forced by explicit linefeed characters: U+000A, U+000D, U+2028, U+2029, unless they are suppressed by linefeed normalization;
2. Line break is permitted at space characters: U+0009, U+0020, U+2000–U+200B, U+3000;
3. If `hyphenate` trait 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. The instance of soft hyphen at the end of line is replaced by text specified in `hyphenation-character` trait; all other instances of U+00AD are suppressed.
4. Unless permitted by the above rules, line break is inhibited in the following conditions:
 - before and after non-breaking spaces and hyphens: U+00A0, U+200C, U+200D;
 - before trailing punctuation characters, closing brackets and quotes, small Katakana and Hiragana characters, superscript characters, etc.;
 - after opening brackets and quotes, Spanish leading punctuations, currency symbols, etc.
5. Unless prohibited by the above rules, line break is permitted before or after CJK ideographic, Katakana, Hiragana, and Hangul characters;
6. Otherwise, line break is prohibited.

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

7.2. Hyphenation

XEP 4.9 uses Unicode soft hyphen characters (U+00AD) to mark possible hyphenation points. These characters can be contained in the source XSL FO document (e.g. come from an external hyphenation software). XEP can also add them inside: it contains a hyphenator class that inserts soft hyphens to all text data before they are passed to the formatter.

The hyphenator implements Liang's algorithm (the same as used in TeX), and uses TeX format for hyphenation patterns. It recognizes sections patterns (for hyphenation patterns) and hyphenation (for exceptions); any other section in the pattern file is ignored. TeX macro definitions are not supported. 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.

XEP's distribution includes patterns for the following languages: English (American and British), French, German, Spanish, and Russian. All patterns were 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; the procedure is described above, in the chapter on [Configuring Hyphenation](#).

7.3. Support for Right-to-Left Writing Systems

XEP supports right-to-left writing and mixed left-to-right/right-to-left text. Glyphs are placed on a line respecting the directionality and nesting of text spans, and glyph mirroring is performed for characters that change their orientation in right-to-left writing.

7.3.1. Bidirectionality

XEP 4.9 provides a limited support for *bidirectionality*. Implicit ordering of glyphs in an inline area that contains only text (no intervening markup) is governed by a simplified version of Unicode bidi algorithm. Only one level of embedding is opened — no nested spans are created. Moreover, bidirectional reordering does not work across markup: any intruding tag splits the sequence into separate pieces that are ordered according to the dominant direction.

XEP 4.9 supports explicit redefinition of writing direction (with `<fo:bidi-override>` element and its properties — `direction` and `unicode-bidi`). Use it wherever possible to avoid dependencies on implicit bidi reordering.

7.3.2. Glyph Shaping

XEP 4.9 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 will not work with Simplified Arabic fonts.

8. Accessibility Support in XEP

XEP 4.9 can create accessible PDF documents that are in compliance with *Section 508 Standards* (<http://www.section508.gov> [<http://www.section508.gov/index.cfm?&FuseAction=Content&ID=12>])

This feature is controlled by the `ENABLE_ACCESSIBILITY` core option in the configuration file.



Accessibility support is applied for 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.

 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 *tagged* PDF.

8.1. Tagged PDF

XEP creates a tagged PDF with a logical structure derived from the structure of the input XSL-FO document.

 By default, Adobe Acrobat reads a tagged PDF according to its logical structure, which will coincide with the order of the input XSL-FO document. To change the reading order to being in accordance with a visual content, choose the 'Left-to-right, top-to-bottom reading order' from '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.

 Adobe Acrobat's 'Accessibility check' plug-in fails if some table cells have not a table row as the parent. Make sure that all table cells in your XSL-FO file are within table rows.

9. XEP on Java Platform

The present chapter describes aspects of XEP usage that are specific to Java platform.

9.1. Software Prerequisites

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.

Charsets.jar is required for proper operation of AFP Backend. This package is installed if "Support for additional languages" option is checked ON upon JRE install.



Although JDK installer has this option ON by default, installer of JRE has it turned OFF. So this option must be manually turned ON upon JRE installation procedure.

9.2. Contents of the Distribution

The package contains the following files:

Root directory

readme.txt

A read-me-first note for XEP.

xep.xml

Main XEP configuration file.

x4u.bat (Windows)

x4u (Unix/Linux)

Script to launch XEP Assistant, a GUI shell to XEP.

xep.bat (Windows)

xep (Unix/Linux)

Command-line script to run the formatter.

validate.bat (Windows)

validate (Unix/Linux)

Command-line script to run the validator.

lib/

Jars used by XEP, including third-party libraries:

images/

Images:

404.gif

Icon inserted as an emergency replacement when an image in the document is missing or invalid.

colorbar.svg

registration.svg

Sample SVG images to produce typographical marks (color bars and registration targets).

fonts/

Font metrics and outlines:

***.afm**

Adobe Font Metric files.

***.pfa, *.pfb**

Outline files for Type 1 fonts.

***.glyphs**

Glyph List files.

hyphen/

Files related to hyphenation:

***.tex**

TeX hyphenation patterns.

doc/

XEP documentation:

intro.pdf

intro.html

Introduction to XEP 4.9. Provides an overview of XEP capabilities and areas of application.

reference.pdf

reference.html

XEP 4.9 Reference for Java (this document). Includes detailed information on XEP configuration, describes supported input and output formats as well as supported graphic and font formats and their configuration.

tutorial.pdf

tutorial.html

XSL Formatting Objects Tutorial. A tutorial of XSL Formatting Objects and some RenderX extensions.

WhatsNew.txt

What's New in XEP 4.9. Tracks the history of changes from previous XEP versions.

examples/

XSL samples:

basic/

Miscellaneous examples of basic formatting features.

charsets/

Samples of font handling in XEP.

hammer/

An example of XSL formatting stylesheet.

xmlspec/

A more complicated stylesheet, used to format W3C documents.

9.3. XEP Assistant — a GUI shell for XEP

XEP 4.9 includes a GUI shell to configure XEP and run formatting. It is invoked as follows:

```
java com.renderx.xep.x4u.as.Assistant
```

For the tool to run, a Java system property `com.renderx.xep.CONFIG` must be set, and point to a valid XEP configuration file. When XEP is installed, the setup creates a launch script that wraps the respective Java call, setting classpath and environment properly: `x4u.bat` for Windows systems, `x4u` for Unix/Linux.

9.4. Command-Line Interface to XEP 4.9

XEP can be used from the command line either as a formatter of XSL-FO files, or as an XSL-FO transformer (provided that a JAXP XSLT transformer factory is present in the classpath). It has the following synopsis:

```
java com.renderx.xep.XSLDriver -help
```

Displays a synopsis reminder and exits.

```
java com.renderx.xep.XSLDriver
    {option} {-quiet | -version | -valid | -hosted}
    ( [-xml] <infile> [-xsl <stylesheet>] {-param <name=value>}
    | -fo <infile>
```



```
| -xep <infile> )  
[[-out | -pdf | -ps | -afp | -at] <outfile>]  
[-format <output format>]
```

An option is a pair *name=value* preceded by `-D` prefix. For a full list of valid XEP options, please refer to [Configuration](#) above. Each option can be also set through a Java system property. For option *X*, the respective property is `com.renderx.xep.X`.

To run XEP, it is necessary to specify the location of its configuration file. By default, the formatter looks for a file names `xep.xml` in the current directory of the process; you can override this default by setting a `CONFIG` option in the command-line parameters, or through a system variable `com.renderx.xep.CONFIG`. The location can be specified either as a file name in the local filesystem, or as an URL.

Other switches have the following meaning:

-version

prints the current version number

-valid

turns off validation

-quiet

enables quiet mode - only warnings and errors are displayed

-hosted

disables system exit calls; used to prevent the JVM from stopping after `XSLDriver.main()`.

Input:

-fo

next token is a source XSL FO file name

-xep

next token is a XEP-generated formatted document representation in XML format (produced with `-at` output switch);

-xml

next token is a source XML file name

-xsl

next token is an XSL stylesheet file name

-param

next token is an XSL transformation parameter

Output:

-out

silently ignored (present for compatibility with previous versions)

-format

next token is an output file format specifier

-<output format>

sets output format to the name of the switch

A single dash ("-") as an input file name denotes standard input. Similarly, a dash as an output file name denotes standard output. Omitting input file specification also means reading from standard input.

The default output format is PDF. If the output file name is not specified, it is constructed from the source file name by adding an extension corresponding to the selected format.

Program exit codes are zero for successful termination and non-zero for failure. If `-hosted` switch is specified, no exit code is ever returned; failures result in an exception being thrown.

If you haven't set `CLASSPATH` system variable, you will need to supply an additional `-classpath` parameter in your Java call, to include required library files into class search path. You may need to add an `-Xmx` switch to the Java call mentioned above in order to increase the amount of memory available to Java. Recommended value of `-Xmx` is about 80% of physical RAM available on the computer that runs XEP.

XEP installation includes a launch script that wraps a respective Java call: `xep.bat` for Windows systems, `xep` for Unix/Linux. This script is preconfigured during XEP installation.

9.5. Command-Line Interface to XSL-FO Validation

There is also a command-line interface to the validator component of XEP that lets you check the structure of your XSL FO file without formatting them:

```
java com.renderx.xep.Validator arguments
```

List of Output Generators Options

Arguments are one or more XSL FO file names, validated one after another. There is a preconfigured launch script, too: `validate.bat` for Windows systems, `validate` for Unix/Linux.

9.6. 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>.

A. List of Output Generators Options

Feature	Option Name	Default Value	Output format
Document Security	OWNERPASSWORD	null	PDF
Document Security	USERPASSWORD	null	PDF
Document Security	USERPRIVILEGES	annotate	PDF
PDF Version	PDF_VERSION	1.4	PDF
Compression of PDF Streams	COMPRESS	true	PDF
Linearization	LINEARIZE	false	PDF

Feature	Option Name	Default Value	Output format
Unicode Strings in Annotations	UNICODE_ANNOTATIONS	true	PDF, PostScript
Treatment of Unused Destinations	DROP_UNUSED_DESTINATIONS	true	PDF, PostScript
Initial Zoom Factor	INITIAL_ZOOM	auto	PDF, PostScript
PDF Viewer Preferences	VIEW_MODE	auto	PDF, PostScript
PostScript Language Level	LANGUAGE_LEVEL	3	PostScript
EPS Graphics Treatment	CLONE_EPS	auto	PostScript
Images Treatment in XML Output	EMBED_IMAGES	false	XML

B. Configuration File DTD fragment

This DTD fragment describes the format of XEP configuration file. Namespace nodes and prefixes are omitted for clarity.

```

<!ELEMENT config (options?, fonts, languages?)>
<!ATTLIST config
    xml:base CDATA #IMPLIED>

<!ELEMENT options (option | generator-options)+>
<!ATTLIST options
    href CDATA #IMPLIED>

<!ELEMENT generator-options (option+)>
<!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
    default-family CDATA #IMPLIED
    embed CDATA #IMPLIED

```

Configuration File DTD fragment

```
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
    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
    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
    xml:base CDATA #IMPLIED>

<!ELEMENT transform EMPTY>
<!-- ATTTLIST transform
    slant-angle CDATA #IMPLIED>

<!ELEMENT font-alias EMPTY>
<!-- ATTTLIST font-alias
    name CDATA #REQUIRED
    value CDATA #REQUIRED>

<!ELEMENT languages (language+)>
<!-- ATTTLIST languages
    href CDATA #IMPLIED
    xml:base CDATA #IMPLIED>

<!ELEMENT language (hyphenation?, font-alias*)>
<!-- ATTTLIST language
    name CDATA #IMPLIED
    codes NMTOKENS #REQUIRED
    xml:base CDATA #IMPLIED>

<!ELEMENT hyphenation
    pattern CDATA #REQUIRED
    encoding CDATA #IMPLIED
    xml:base CDATA #IMPLIED>
```

C. XEP Intermediate Output Format Specification

This chapter describes XEP intermediate output format — an XML based representation of the layout that is passed from the generator to 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>
```

Let's consider the output elements in more details:

<document>

Root element; initialization and finalization are usually performed at its beginning and end.

Attributes:

creator

information about document creator application;

initial-destination

(optional) destination that should be moved into focus when the document is first opened.

There are several other attributes (such as `author` or `title`) that transferred 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.

Attributes:

height

height of the page;

width

width of the page.

page-id

page number label

page-number

page number as an ordinal

<rotate>

Rotates the coordinate system.

Attributes:

phi

rotation angle, in degrees. May take values in multiples of 90: 0, 90, 180, 270 degrees.

<translate>

Shifts the origin of the coordinate system.

Attributes:

x

horizontal shift distance;

y

vertical shift distance.

<word-spacing>

Sets word spacing (additional spacing between words).

Attributes:

value

the value of word spacing.

<letter-spacing>

Sets letter spacing (additional spacing between non-space characters).

Attributes:

value

the value of letter spacing.

<font-stretch>

Sets font-stretch factor.

Attributes:

value

the value of font-stretch factor.

Changes the current font.

Attributes:

family

font family;

weight

font weight (100 to 900);

style

font style (normal, italic, oblique, or backslant);

variant

font variant (normal or small-caps);

size

font size;

<text>

Prints a character string.

Attributes:

x

horizontal position of the initial point on the baseline;

y

vertical position of the initial point on the baseline;

value

the text string to print;

width

text width.

<line>

Draws a line.

Attributes:

x-from

horizontal position of initial point

y-from

vertical position of initial point

x-till

horizontal position of final point

y-till

vertical position of final point

thickness

style

<image>

Embeds an external image.

Attributes:

src

the source URL of the image;

base

the base directory to resolve hrefs in the image;

type

the image MIME type;

x-from

horizontal position of the lower left corner of the image;

y-from

vertical position of the lower left corner of the image;

scale-x

horizontal scaling factor;

scale-y

vertical scaling factor;

role

alternate description for accessible PDFs.

<gray-color>

Sets current color for stroking and filling. Color is chosen in grayscale color space.

Attributes:

gray

gray color value, 0 to 1;

<rgb-color>

Sets current color for stroking and filling. Color is chosen in RGB color space (additive).

Attributes:

red

red color value, 0 to 1;

green

green color value, 0 to 1;

blue

blue color value, 0 to 1.

<cmym-color>

Sets current color for stroking and filling. Color is chosen in CMYK color space (subtractive).

Attributes:

cyan

cyan color value, 0 to 1;

magenta

magenta color value, 0 to 1;

yellow

yellow color value, 0 to 1.

black

black color value, 0 to 1.

<spot-color>

Sets current color for stroking and filling. Color is chosen in a spot color space (subtractive).

Attributes:

colorant

colorant name;

tint

color intensity, 0 to 1;

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.

To describe alternate color, exactly one of the following attribute sets must be present:

- alt-gray — fallback color is grayscale;
- alt-red, alt-green, alt-blue — fallback color is RGB;
- alt-cyan, alt-magenta, alt-yellow, alt-black — fallback color is CMYK.

<registration-color>

Sets current color for stroking and filling. Color is chosen in registration color space (subtractive): it appears on all separations present in the document.

Attributes:

tint

color intensity, 0 to 1.

<rectangle>

Draws a filled rectangle.

Attributes:

x-from

horizontal position of lower left corner;

y-from

vertical position of lower left corner;

x-till

horizontal position of upper right corner;

y-till

vertical position of upper right corner.

<clip>

Sets clipping area.

Attributes:

x-from

horizontal position of lower left corner;

y-from

vertical position of lower left corner;

x-till

horizontal position of upper right corner;

y-till

vertical position of upper right corner.

<polygon>

Draws a filled polygon. All vertices but the first one are specified in the contained `<point>` elements.

Attributes:

x-from

horizontal position of the first vertex

y-from

vertical position of the first vertex

<point>

Adds a vertex to a polygon.

Attributes:

x-till

horizontal position

y-till

vertical position

<target>

Sets endpoint for an internal destination

Attributes:

name

internal destination name (id of the element that created the target)

x

horizontal position

y

vertical position

<internal-link>

Specifies an internal link destination.

Attributes:

destination-id

name of the target endpoint; should match name attribute of a <target> element somewhere in the document;

destination

page number to point the link to;

x-destination

horizontal position of the destination point;

y-destination

vertical position of the destination point;

<external-link>

Specifies an external link destination.

Attributes:

destination

an URL

show-destination

controls whether to create a new window when jumping to the link target.

<internal-bookmark>

Specifies an internal bookmark destination.

Attributes:

label

bookmark text;

id

bookmark ID — a positive integer;

parent-id

ID of the parent bookmark, or 0 if the bookmark is a top-level one;

destination-id

name of the target endpoint; should match name attribute of a <target> element somewhere in the document;

destination

page number to point the link to;

x-destination

horizontal position of the destination point;

y-destination

vertical position of the destination point;

collapse-subtree

initial state (collapsed or expanded).

<external-bookmark>

Specifies an external bookmark destination.

Attributes:

label

bookmark text;

id

bookmark ID — a positive integer;

parent-id

ID of the parent bookmark, or 0 if the bookmark is a top-level one;

destination

an URL;

collapse-subtree

initial state (collapsed or expanded);

show-destination

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 obeys the following rules:

- instructions placed before `<fo:root>` element in the source file will be reproduced at the very top, before the root `<document>` element;
- instructions placed inside a `<fo:simple-page-master>` element will be 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 the above, ordering of instructions is not preserved.

D. List of Warning and Error Messages

The following table represents warning (W) and error (E) messages issued by XEP. <Message> pattern holds the place where extra information is printed, generally from Java exceptions.

Type	Warnigns and Errors
E	'{' expected.
E	'format' descriptor for MIME type <mimetype> has no 'parser' attribute.
E	'format' descriptor has an empty 'mime-type' attribute.
E	'format' descriptor has no 'mime-type' attribute.
E	'hyphenation' element is not a child of 'language' element; element ignored.
E	'namespace' descriptor has an empty 'mime-type' attribute.
E	'namespace' descriptor has no 'mime-type' attribute.
E	'namespace' descriptor has no 'uri' attribute.
W	'span' attribute on <name> ignored because the element is not a direct child of a flow.
W	(starts ends)-row inside a row.
E	[Line <linecounter>]: spot color <name> has an invalid CMYK equivalent.
E	[Line <lineno>]: glyph <name> has an invalid Unicode number.
E	<arg> could not be parsed due to structure errors.
E	<BAD TOKEN TYPE - INTERNAL ERROR>.
E	<Count> error[s] found during validation.
E	<ERROR! INCORRECT BASE FOR LENGTH>.
E	<ERROR: UNINITIALIZED TRANSFORM>.
E	<Internal error: wrong color type>.
E	<Message>; option <option> ignored.
W	Alias <name> contains reference to an undefined font family <FontFamily>.
W	Alias <name> redefined; the first definition is kept.
E	An LZW-encoded strip does not end with EOI pattern (100000001) - data format error in PDF file <Message>.
E	An LZW-encoded strip does not start with ClearCode pattern (100000000) - data format error in PDF image <Message>.

List of Warning and Error Messages

Type	Warnigns and Errors
E	An LZW-encoded strip is too short, only <length> bytes long - data format error in PDF file <Message>.
W	Annotations inside page area are not allowed in <PDFXVersion>; external link <url> removed.
W	Annotations inside page area are not allowed in <PDFXVersion>; local link <dest> removed.
W	Annotations inside page area are not allowed in <PDFXVersion>; PDF link <filename> removed.
W	Annotations inside page area are not allowed in <PDFXVersion>; text annotation <title> removed.
W	Attachment defined twice in <name> shorthand: second definition ignored.
W	Attribute ignored: <Message>.
E	Attribute ignored: <name>=<value>; <Message>.
W	Bad attribute <name>: <Message>.
E	Bad image file URL, <Message>.
W	Bad table structure: overlapping cells may be truncated or skipped.
E	Bad value for slant angle: <slantAttr>; property ignored.
E	Bad value for smallcap size ratio: <smallcapAttr>; using default value of 0.8.
E	bad value for style switch: <style>.
E	Bookmark with id <ID> already exists.
E	bounding box missing.
E	break with context 'auto'.
E	Broken font file <fontfile>. <Message>.
E	Cannot constructs serializer from an output stream: <Message>.
E	Cannot create temporary file <tempFile>, <Message>.
E	Cannot create XML reader: <Message>.
W	Cannot draw printer marks <Message>.
W	Cannot draw printer marks: <Message>.
E	cannot find image parser class <parser> for MIME type <mimetype>.
E	Cannot insert image <image>. <Message>.

Type	Warnigns and Errors
W	Cannot insert image: <Message>.
E	Cannot load the stat DLL for Java. Make sure that PATH points to the directory with the stat DLL. <Message>.
E	Cannot make base directory from system ID: <sysID>.
E	Cannot parse a document: <Message>.
E	Cannot parse image file, <Message>.
E	Cannot read font file <fontfile>, <Message>.
W	Cannot reconnect to <URL>. A previously cached copy will be used.
E	Cannot store attribute <name> in attribute list: <Message>.
E	cannot use neither raw nor expanded bitmap data in file <file>.
E	Cannot use TrueType/OpenType font: no Unicode mapping table found.
E	Cannot use TrueType/OpenType font: unknown format of Unicode 'cmap' table.
E	CFF font data doesn't contain supplied font name.
E	CFF font embedding is a feature of Postscript Level 3; font <fontName> not embed.
W	Character <U> is not recognized as a ligature; character ignored.
W	clip-path' property is not supported on SVG 'clipPath' element and its descendants.
E	codes' attribute on 'language' element contains no meaningful tokens: <codesAttr>.
W	Color defined twice in <name> shorthand: second definition ignored.
W	could not find any font family matching <FontFamily>; replaced by <DefaultFont-Family>.
W	Could not retrieve image from <url>: <Message>.
W	CSS parser: <Message>.
E	Directory <tmpdir> does not exist; disk caching disabled.
E	Directory <tmpdir> does not exist; disk caching in PDF generator disabled.
E	Directory <tmpdir> does not exist; disk caching in PostScript generator disabled.
E	Directory <tmpdir> is not writable; disk caching disabled.
E	Directory <tmpdir> is not writable; disk caching in PDF generator disabled.
E	Directory <tmpdir> is not writable; disk caching in PostScript generator disabled.
W	display-align=<valign> not supported for area-containers.

List of Warning and Error Messages

Type	Warnigns and Errors
E	Document <context> does not contain element with id=<id>.
E	Document already started!
E	Element with id <id> does not exists.
E	Empty file.
E	Empty list of codes for language <langName>.
E	Error during parsing test data! File: <file> <Message>.
E	Error during setup! <Message>.
E	Error in a function.
E	Error in a hexcolor operand.
E	Error in an operand.
E	Error in image factory configuration file: <Message>.
E	Error in OpenType/CFF font: Glyph ID <gid> exceeds max number of glyphs <numGlyphs>.
E	Error in TrueType font: Glyph ID <glyphid> exceeds max number of glyphs <numGlyphs>.
E	Error in TrueType font: Glyph ID for <U> not found in cmap table.
E	error initializing com.renderx.util.Magic: <Message>.
E	error initializing ImageFactory: <Message>.
E	Error parsing image, <Message>.
E	error parsing location, <Message>.
E	error processing <path>: <Message>.
E	Error processing attribute <name> on element 'font-data'; attribute ignored, <MalformedURLException>.
E	Error processing image, <Message>.
E	Error processing license file <license>, <Message>.
E	error signing the license: <Message>.
E	Error while closing element <name>: <Message>.
E	Error while opening element <name>: <Message>.
E	Error while parsing <Message>.

Type	Warnigns and Errors
E	Error while parsing CFF metric: STRING INDEX out of boundary.;
E	Error while processing text node: <Message>.
E	Error while reading configuration file: <Message>.
E	Error while reading source: <Message>.
E	Error! <fo:name> doesn't supported markers:
E	Error! Can't find properties! <Message>.
E	Error! Element <fo:name> doesn't support markers:
E	error: cannot open file <out> for writing: <Message>.
E	error: duplicate output format.
E	error: duplicate output.
E	error: formatting failed, <Message>.
E	error: problem accessing file <out>, <Message>.
E	error: too few command line arguments.
E	error: unexpected command line argument #<i> (input parameter): <arg>.
E	error: unexpected command line argument #<i> (output parameter): <arg>.
E	Expect at least two tokens of the line.
E	Expected a hex number in angle brackets.
E	Expected identifier.
E	Failed to read image file, <Message>.
E	failed to recover, truncating.
E	fatal error at element: <name>: <Message>.
E	fatal error at the end of document <Message>.
E	Fatal parse error: <Message>.
E	File <tmpdir> is not a directory; disk caching disabled.
E	File <tmpdir> is not a directory; disk caching in PDF generator disabled.
E	File <tmpdir> is not a directory; disk caching in PostScript generator disabled.
E	File could not be retrieved from <SystemId>; skipped.
E	File does not appear to be a Cmap.
E	Font <fontName> cannot be embedded because of license restrictions.

List of Warning and Error Messages

Type	Warnigns and Errors
E	Font <fontName> cannot be embedded because of license restrictions; PDF/X status reverted to 'none'.
E	Font <fontName> cannot be subsetted because of license restrictions.
E	font' element outside of 'font-family' ignored.
E	Font family <familyName> defined twice; first definition preserved.
E	Font is not a TrueType Collection; cannot extract multiple fonts.
E	font name missing.
E	font-data' element outside of 'font'; element ignored.
E	Formatter initialization failed, <Message>.
E	I/O error: <Message>.
W	Ignored unexpected token in <name> shorthand: <token>.
W	Ignored unexpected token in <name>: <token>.
E	illegal region-body@reference-orientation: <phi>.
E	Image has Indexed color space with RGB colorants that is not allowed in PDF/X; PDF/X status reverted to 'none'.
E	Image has RGB color space that is not allowed in PDF/X; PDF/X status reverted to 'none'.
E	Image is a PDF image that is not allowed in PDF/X; PDF/X status reverted to 'none'.
W	Image URL defined twice in <name> shorthand: second definition ignored.
W	Incorrect number of tokens in <option>; setting ignored.
W	Incorrect or unsupported LANGUAGE_LEVEL value: <value>; setting ignored.
W	Incorrect syntax in <option>; should be: url('<sorce name>').
W	Initial destination <InitialDestination> not found; parameter ignored.
W	Initial destination <initialDestination> not found; parameter ignored.
E	Internal error: bad value for style switch: <style>.
E	Internal error: cannot create a serializer for an embedded image: <Message>.
E	Internal error: cannot lazily copy data from TIFF image <Message>.
E	Internal error: empty encoding.

Type	Warnigns and Errors
E	Internal error: incorrect base in a length value.
E	Internal error: incorrect transformation step <type>.
E	Internal error: invalid color type in SVGAttrValue.Color.
E	Internal error: invalid command in parsed path data.
E	Internal error: invalid compression type in TIFF image <Message>.
E	Internal error: invalid rule style index <style>.
E	Internal error: SVG parser misconfigured.
E	Internal error: unexpected bit count <bitCount> in Base64OutputStream.
E	Internal error: unrecognized separator character.
E	Internal misconfiguration: text-shadow incorrectly parsed.
E	Invalid boolean value <key>=<value>; reverting to the default value of <defvalue>.
E	invalid border-<id>-color: <color>.
E	invalid border-<id>-style: <style>.
E	Invalid CFF font data: Charset encoding type is incorrect.
E	Invalid CFF font data: Charsets data are missing.
E	Invalid CFF font data: FD Array operator is missing.
E	Invalid CFF font data: font name table is empty.
E	Invalid CFF font data: offset element size is <offsetElementSize>; it is out of range 1-4.;
E	Invalid change-bar-color: <color>.
W	Invalid change-bar-placement: <place>, using 'start'.
W	invalid change-bar-style: <style>, using 'solid'.
W	Invalid CLONE_EPS value: <value>; setting ignored.
W	Invalid COMPRESS value: <value>; setting ignored.
E	Invalid double value <key>=<value>; reverting to the default value of <defvalue>.
W	Invalid element in anonymous namespace: <name>; element skipped.
W	Invalid EMBED_IMAGES value: <value>; setting ignored.
E	Invalid font number <subno> in TrueType collection: should be in the range from 1 to <subfontsTotal>.

List of Warning and Error Messages

Type	Warnigns and Errors
E	Invalid format name for generator options: <backend>.
E	invalid image parser class <imgclass> for MIME type <mimetype>.
E	invalid input: flow may de defined only once.
E	invalid input: title may de defined only once.
E	Invalid integer value <key>=<value>; reverting to the default value of <defvalue>.
E	invalid leader-pattern <leader_pattern>.
E	Invalid length specifier <key>=<value>; reverting to the default value of <defvalue>.
W	Invalid LINEARIZE value: <value>; setting ignored.
E	invalid MIME type specifier <mimetype>: <Message>.
E	invalid MIME type specifier <typename>: <Message>.
E	Invalid name of the configuration system property: <key>.
E	Invalid option name in the argument list: <key>.
W	Invalid RenderX extension element: <name>; element skipped.
E	invalid rule color: <color>.
E	invalid rule-style: <style> (not applicable to leaders).
E	invalid rule-style: <style>.
W	Invalid structure of <name> attribute: too few consecutive lengths.
W	Invalid token in USERPRIVILEGES: <token>; token ignored.
E	Invalid TrueType/OpenType file structure: no <name> table found.
W	Invalid UNICODE_ANNOTATIONS value: <value>; setting ignored.
W	Invalid URL or non-existent file: <URL>; setting ignored.
W	Invalid URL or non-existent file: <url>; setting ignored.
E	Invalid URL or non-existent file: <url>; setting ignored; PDF/X status revreted to 'none'.
E	Invalid value <value> for a boolean property ignored.
W	Invalid value for DROP_UNUSED_DESTINATIONS: <value>; setting ignored.
E	Invalid value in the configuration file: <key>=<value>; setting ignored.
W	Invalid value number-columns-repeated=<colrepeat>; defaulted to 1.
W	Invalid value number-columns-spanned=<colspan>; defaulted to 1.

Type	Warnigns and Errors
W	Invalid value of <name> attribute: <value>; shadow set to none.
W	Invalid xml:base: cannot convert <value> to an URL: <Message>.
W	last page's content does not fit into the last page, reverting to 'rest'.
E	Lexer internal error: unhandled single-char token <token>.
E	License check failed: <Message>. If you keep getting this error after the activation procedure, please contact support@renderx.com attaching your license.xml file.
E	Magic number in the font header does not match the spec: file probably corrupt.
E	Mandatory attribute <att> is missing on element <name>; element skipped.
W	Mismatched change-bar-end element of class <cbclass> (ignored).
W	Misplaced token in <name> shorthand: <token>.
E	Named destination cannot be added, page is not initialized!
E	negative font stretch is not allowed <value>.
E	Nested 'font-family' elements are not permitted: inner element ignored.
E	No 'clipPath' element found with id=<clipPath>.
E	no entries for index key <id>.
E	No meaningful family names in the value of alias: <name>=<value>.
E	No options are defined for Java back-end.
E	No outline file for font metric <metricfile> -- font <fontName> cannot be embedded; PDF/X status reverted to 'none'.
E	No outline file for font metric <metricfile>; font cannot be embedded.
E	No PFA/PFB file found for font <fontName>.
W	no place for a footnote, some data will be lost.
E	No region <flowname> on last page #<page.id>.
E	No region <flowname> on page #<page.id>.
E	No source specified for an image.
W	no space for a float, some data will be lost.
E	no space for an element, trying to recover.
W	No value specified for PAGE_DEVICE parameter <pdname>; setting ignored.
E	Not a OpenType/CFF file.

List of Warning and Error Messages

Type	Warnigns and Errors
E	Not a TrueType/OpenType file, or unsupported TrueType version: <major.minor>.
W	Null System ID: relative URLs may not work properly.
W	Opacity is not supported on SVG 'tspan' element.
E	OpenType/CFF (CID) font has no glyph names.
E	Output color profile must be present in <pdfxConformanceStatus>; PDF/X status re- vreted to 'none'.
W	overflow in <absolute fixed> block-container.
W	overflow in a float.
W	overflow in a table-cell or block-container.
E	Parse error: <Message>.
W	Parse warning: <SAXParserMessage>.
W	PDF version <PDFVersion> does not support opacity; feature ignored.
W	PDF version <Version> does not support Tagged PDF; reverted to the lowest possible value: 1.4.
W	PDF Version <Version> is not allowed in <PDFXVersion>; reverted to maximum possible value: 1.3.
W	PDF/X of version <PDFXVersion> should not use an encryption; settings ignored.
E	PDF/X of version <version> should not use an encryption.
E	Pdflib internal error. Broken xref. Please report to manufacturer!.
E	problem parsing input.
W	Processing instruction ignored: <Message>.
W	Repeated color specification in <name> ignored.
W	RGB colors are not allowed in <PDFXVersion>; RGB color converted to CMYK.
E	Serializer error: <Message>.
W	Some characters from <FontName> font cannot be represented if the font is unembed- ded.
W	Some images lack alternate desription; "No alternate description specified" will be used for these images.
W	Some table cells do not have table row parent; PDF may not be fully accessible.
W	Some text elements have no language specified; 'en' will be used.

Type	Warnigns and Errors
W	Stray change-bar-end element of class <cbclass> (ignored).
E	Structure error in configuration file: <Message>.
E	Subsetting for Type 1 OpenType/CFF fonts not implemented yet.
E	Support for OpenType/CFF fonts with predefined charset not implemented yet.
E	SVG element <name> ignored: <Message>.
E	SVG property <name>=<value> ignored: <Message>.
E	table cell has unknown parent element.
E	table cells overlap, skipping a cell.
W	table cells overlap.
E	text-decoration=<decor> is not supported.
E	The code for hashes of lengths other than 16 is not in place yet.
W	Tiling defined twice in <name> shorthand: second definition ignored.
E	Too few numbers in the array: expected <length>.
W	Too many length specifiers in <name>: extra tokens ignored.
E	Too many numbers in the array: expected <length>.
W	trailing transformation parameters are deprecated, use -param instead.
E	transform' element outside of 'font'; element ignored.
E	Trying to write data of negative length; CFF subsetting is broken.
W	unbalanced end key <id> on page <page>.
W	Unclosed change-bar-begin element of class <cbclass>.
E	undefined label <id>.
W	Unembedded fonts are not allowed in PDF/X; font <FontName> will be embedded.
W	Unexpected element in the clipPath <ClipClass> skipping:
W	Unexpected element in the clipPath: <ClipClass> skipping:
E	Unexpected number of elements in the clipPath - expected zero or one, found: <length>.
W	Unexpected token in <name>: <tokentype>.
W	Unexpected token type in <name> shorthand: <tokentype>, token ignored.
E	Unknown FO element name: <name>.

List of Warning and Error Messages

Type	Warnigns and Errors
E	Unknown MIME type: <mimetype>.
W	Unknown SVG element: <Name>; element skipped.
W	Unknown SVG element: <name>; element skipped.
W	Unknown XSL-FO element: <name>; element skipped.
E	Unparsed a OpenType file.
W	Unrecognized PDF option <option>: setting ignored.
W	Unrecognized PDF Version number: <Version>; reverted to default value: <DefaultVersion>.
W	Unrecognized PostScript option <option>: setting ignored.
E	Unrecognized style value <style> in font family <familyName>.
W	Unrecognized SVG generator option <option>: setting ignored.
E	Unrecognized variant value <variant> in font family <familyName>.
E	Unrecognized weight value <weight> in font family <familyName>.
W	Unrecognized XEP output option <option>: setting ignored.
E	Unresolved image reference: id-ref=<idref>.
W	Unresolved initial destination: <initialDestination>; parameter ignored.
W	unresolved internal destination: <refid>.
W	unresolved page-number-citation: <refid>.
E	Unsupported CFF CID font type: <CIDFontType>.
E	Unsupported CFF font: the font contains more than one subfont.
E	Unsupported encoding: <encoding>.
E	Unsupported file format or broken file found reading font metric from <metricfile>: <Message>.
E	Unsupported font type in PSDocument.setfont() for font <fontName>.
W	Unsupported PDF Version: <Version>; reverted to the lowest possible value: 1.3.
W	Unsupported PDF/X version: <PDFXVersion>; reverted to default PDF/X status: <DefaultPDFXVersion>.
E	Unsupported type of CFF Charstring encoding: <charstringType>.
E	Unsupported version of TrueType Collection header: <major.minor>.

Type	Warnigns and Errors
W	writing mode <writingmode> is not supported.
E	Wrong document: <URI>. <Message>.
W	Wrong font stretch: <value>!
E	Wrong Format Type into post table.
W	Wrong line cap style: <style>.
W	Wrong line width: <width>.
E	Wrong link: <link>.
W	Wrong value for view mode: <viewMode>; parameter ignored.
W	Wrong value for zoom factor: <zoom>; parameter ignored.
E	Wrong xml:base in the <name> element: <xml:base>.
E	Wrong xml:base in the root svg element: <xml:base>.
E	xml:base value ignored: cannot convert <value> to a URL: <Message>.

E. XEP AFP Generator

E.1. Introduction

Since version 4.6 XEP engine has a built-in support for generating AFP documents (available with a special license). AFP is an architecture standard for High Volume Transaction Output supported by such vendors of printing equipment like IBM, Kodak, Xerox and others. AFP has built-in support for text and raster graphic output, vector graphic, vector and raster fonts and many other features. All document structure of AFP document is organized by means of higher level protocol called MO:DCA which links all printable objects together and builds the whole document.

E.2. Generated AFP Documents

XEP AFP generator generates documents which are MODCA-P streams that contain PT:OCA, I:OCA, BC:OCA, and G:OCA objects. XEP AFP generator can also generate an additional stream that can contain some of raster images as IOCA objects. This stream is a MO:DCA-L stream and it is called resource file.

For purpose of interoperability each MO:DCA structured field is prefixed with byte X'5A'. For details on X'5A' prefix usage, see Advanced Function Presentation, Programming Guide and Line Data

Reference, Chapter 4. Mixed Documents: Adding MO:DCA Structured Fields to Line Data, Section "X'5A' Carriage Control Character".

E.3. Starting XEP to generate AFP document

New command line parameters have been added to allow generation of AFP documents and AFP resource files.

- to instruct XEP to generate AFP document use parameter `-afp`:

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

- to instruct XEP to generate AFP resource file use parameter `-DH4AFP.RESOURCE`:

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

Note that as `-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 configuration file variable. For more details, refer [Configuring XEP AFP Generator](#).

E.4. Font mapping

XEP does not support native AFP fonts so far. Instead, XEP uses concept of mapping fonts supported by XEP to AFP native fonts. When an XSL FO document is being rendered, XEP uses metrics from supported fonts listed in XEP fonts configuration files and required in XSL FO document. When AFP file is generated, AFP generator substitutes each font mentioned in XSL FO document for a corresponding AFP font. This is possible as most of AFP fonts are simple conversion of commonly used fonts to internal AFP format (FOCA).

Correspondence between FO fonts and AFP native fonts is configured in AFP generator configuration section of XEP configuration file.

E.5. Raster Image Support in XEP AFP Generator

As AFP does not support many of commonly used raster image formats and color images are rarely supported by AFP printing systems, all raster images are converted to bilevel or grayscale images depending on the configuration options.

- Bilevel images have 1 bit per pixel and are compressed

- Grayscale images have 8 bits per pixel and are uncompressed

 In AFP, bilevel images are mixed with their background. This means that white points appear transparent.

E.5.1. Image Clipping

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


E.6. Highlight Color Support In XEP AFP Generator

Highlight color is a special case of color encoding, when a solid colorant used (contrary to other schemes above). 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 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.

 When the Highlight Color space is specified in a target repeating group, the percent coverage parameter is normally only supported 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.

E.7. Graphics Support In XEP AFP Generator

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

AFP Backend 4.9 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.



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 a "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 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.

E.8. Barcodes Support In XEP AFP Generator

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.

Following types of Barcodes are supported by IBM Bar Code Object Content Architecture (BC:OCA):

- Code 39 (3-of-9 Code), AIM USS-39
- MSI (modified Plessey code)
- UPC/CGPC Version A
- UPC/CGPC Version E
- UPC - Two-digit Supplemental
- UPC - Five-digit Supplemental
- EAN 8 (includes JAN-short)
- EAN 13 (includes JAN-standard)
- Industrial 2-of-5
- Matrix 2-of-5
- Interleaved 2-of-5, AIM USS-I 2/5
- Codabar, 2-of-7, AIM USS-Codabar
- Code 128, AIM USS-128
- EAN Two-digit Supplemental
- EAN Five-digit Supplemental

- POSTNET
- RM4SCC
- Japan Postal Bar Code
- Data Matrix (2D bar code)
- MaxiCode (2D bar code)
- PDF417 (2D bar code)
- Australia Post Bar Code
- QR Code
- Code 93

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 [Graphics support in XEP AFP Generator](#) section for more information about G:OCA implementation.

✎ 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 [Limitations of 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

 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.

E.9. Support for FORMDEF resource

FORMDEF is a special type of AFP resource which is attached to a document (or multiple documents) in a resource section of a printfile and allows controlling of

- paper size and rotation
- simplex or duplex printing mode
- printing of several logical document pages on same sheet
- number of copies
- paper cutting, punching etc

- paper source selection
- overlays for physical and logical pages
- finishing documents

For more details how to instruct XEP to generate AFP resource file, refer [Starting XEP to generate AFP document](#).

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

As **FORMDEF** is a MODCA resource, to generate a document with **FORMDEF** you must specify resource file in XEP command line and form print file by concatenating resource file and document file or files. If a resource file is not specified, **FORMDEF** instructions are processed but not generated in the resulting AFP file.

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

FORMDEF processing instructions are divided into two main groups, the first is non-repeating instructions and the second is repeating instructions.

- Non-repeating instructions must have only one instance in a document, if they are needed. These are:
 - xep-afp-form-definition
- Repeating instructions may appear in a document several times. These are:
 - xep-afp-page-definition
 - xep-afp-copy-group

Repeating instructions contain pseudo-element "id" which uniquely identifies each of instruction instances of same type within a document. When XEP processes repeating instructions, only the first instruction instance with the same id is processed, other instructions are ignored. If an instructions is

in a page master, it is reproduced on each page generated by this page master, but XEP processes this instruction only first time it meets this instruction, as instruction id-s are same.

Syntax and semantics of FORMDEF processing instructions are following.

- **xep-afp-form-definition** instruction defines FORMDEF resource.

Format is:

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

sheet-height and **sheet-width** are mandatory parameters. Values may contain simple size expressions like "3in", "10pt" etc.

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

Format is:

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

- **id** - instruction identifier. Must be a decimal number.
- **x0, y0** - coordinates of logical page presentation space origin on physical sheet. May contain simple size expressions.
- **orientation** - orientation of logical page on physical sheet. May be one of 0, 90, 180, 270.
- **type** - type of the logical page which defines common page layout. See AFP documentation for details. May be one of 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** instruction defines copy group and its attributes and keywords. See AFP documentation, MODCA structured fields MCC and MMC description for details on copy groups concepts.

Format is:

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

- **id** - instruction identifier. Must be a decimal number.
- **copy-count** - number of copies in the copy group. Must be a decimal number.
- **mode** - printing mode, may be one of the following: "simplex", "duplex" or "tumble-duplex".

id, **copy-count** and **mode** are mandatory attributes. Besides these attributes xep-afp-copy-group instruction 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' | 'E' | 'F' ) ,
        ( '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 MODCA MMC structured field content.

Note that printing mode like simplex, duplex or tumble duplex is controlled separately by **mode** attribute which effectively generate X'F4' 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 two times. You must not use X'F4' MMC keyword explicitly.

E.10. Configuring XEP AFP Generator

Configuration of 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"/>`.

 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*.

E.10.1. Configuring Code Pages

Since version of 4.9, XEP has automatic processing of international character sets and does not require configuring. More details in [International Character Set Support](#) section.

E.10.2. 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)



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

E.10.3. 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" />
```

E.10.4. 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 grayscale 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*.

E.10.5. 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"/>
```

Other configuration options

- 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 [Graphics Support in XEP AFP Generator](#) 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 [Barcodes Support in XEP AFP Generator](#) for more details on supported barcode types and barcodes implementation notes in XEP AFP Generator.

E.11. 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>
```



Unicode character used for text bullets must belong to any of supported Character Sets. The most common Unicode characters `0x2022` and `0x2023` used for circle and triangle bullets belong to General Punctuation Unicode Character Set. Since this Character Set has no direct mapping to particular Codepage, these characters are not supported yet and cannot be used for implementing bullets. Please refer [Limitations of XEP AFP Generator](#) for more details.

- 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 [Limitations of XEP AFP Generator](#) for more details.

E.12. 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 supported character sets list). After that, it uses conversion table to convert the character to the CodePage that AFP device is capable to process.

AFP Generator supports the following International character sets:

Name	AFP CodePage	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
Cyrillic	T1000866	Cp866	0x0400-0x04FF	Contains characters for Cyrillic languages

Currently, each Character Set has single CodePage and AFP CodePage assigned, and this cannot be configured.

E.13. Limitations of 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 [Graphics support in XEP AFP Generator](#).
- 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. Please refer [Appendix B](#) for more details on how to configure XEP fonts.
- 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 CharacterSet 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.

E.14. 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.



Actual checkbox name may vary for different versions of JRE.

- **Q: I cannot configure XEP to produce text in International character set**

A: Current version of AFP Generator has several hardcoded Character Sets supported. Please refer [International Character Set Support](#) section to find details on which CharacterSets and CodePages are supported.

E.15. Copyrights

This software partly uses some software from Sun Microsystems.

Copyright (c) 2001 Sun Microsystems, Inc. All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistribution in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind.

ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You acknowledge that Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

F. Additional Components

F.1. XEP Connector for jEdit version 2.1

F.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.

F.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.

F.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.

F.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.

F.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>].

F.2. XEP ANT Task 2.0 User's Guide

F.2.1. Changes since version 1.*


XEP Ant Task now uses the new RenderX XEP API, introduced in XEP 3.7.

F.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.

Parameters

 Availability of output formats depends on XEP edition. In some editions, PostScript generator is not available.

F.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>
```

F.2.4. Parameters

Attribute	Description	Required
in	Specifies a single XML document to process.	Either in or a nested <code><file-set></code> element must be available
out	Used with in, specifies the file name for the formatted document.	No

Attribute	Description	Required
destDir	Used with nested <fileset> 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 <fileset>) 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	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. 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. By default, all documents are reformatted.	No
style	XSLT stylesheet is to apply to source XML documents, either a path-name or a URL. If it is not available, input files are assumed to be XSL FO documents.	No
format	Output format. Possible values:PDF, PostScript, XEP. ! Format identifiers are case-sensitive!	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.

F.2.5. Parameters specified as nested elements

The following nested elements can appear inside of XEP task entry.

Examples

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
name	XSL parameter name	Yes
expression	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

F.2.6. Examples

 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>
```