

XMLmind XSL-FO Converter - User's Guide

Jean-Yves Belmonte
Hussein Shafie, Pixware <xfc-support+xmlmind.com>

XMLmind XSL-FO Converter - User's Guide

by Jean-Yves Belmonte and Hussein Shafie

Publication date December 4, 2009

1. Introduction	1
2. Installing XMLmind XSL-FO Converter	2
1. System requirements	2
2. Installation	2
3. Contents of the installation directory	2
3. Command-line executables	3
4. Integrating XMLmind XSL-FO Converter in your application	4
1. Compiling and running the code samples	4
2. Converting an XSL-FO file to RTF	4
3. Converting an XML document to RTF	5
5. Support of the XSL-FO v1.0 standard	7
1. Features	7
2. Limitations	7
3. Conformance statement	8
4. Implementation specificities	17
4.1. Page references	17
4.1.1. RTF/WML/OOXML	17
4.1.2. OpenDocument	17
4.2. Leaders	17
4.3. Multiple page layouts	18
4.4. Expressions	18
6. XSL-FO extension for Office Open XML	19
1. What it is	19
2. How it works	19
2.1. Text field example	19
2.2. Drop-down list example	20
2.3. Specifying a Custom XML Data template	21
2.4. Extracting the Custom XML Data part	21
3. Reference Material	21
3.1. Generic attributes	22
3.2. sdt:text-field	22
3.3. sdt:drop-down-list	23
3.4. sdt:list-entry	24
3.5. sdt:combo-box	24
3.6. sdt:date	25
3.7. sdt:picture	26
3.8. sdt:image-data	26
3.9. sdt:configuration	26
A. Notice	28

Chapter 1. Introduction

XMLmind XSL-FO Converter (XFC for short) is an XSL-FO processor similar to Apache FOP, RenderX XEP or Antenna House XSL Formatter. Unlike the aforementioned processors which all renders XSL-FO as PDF and PostScript®, XMLmind XSL-FO Converter converts XSL-FO v1.0 to the following formats:

- RTF (Word 2000+),
- WordprocessingML (Word 2003+),
- Office Open XML (.docx, Word 2007+),
- OpenOffice (.odt, OpenOffice.org 2+).

That is, XMLmind XSL-FO Converter *translates* one format, XSL-FO v1.0, to the file formats of the two most commonly used word processors, Microsoft Word and OpenOffice.org Writer.

Working at a higher level than the other XSL-FO processors, XMLmind XSL-FO Converter has intrinsic limitations which are detailed in Section 2, “Limitations” [7]. Despite these limitations, XMLmind XSL-FO Converter allows to process very elaborate XSL-FO files. In practice, you should be able to reuse *as is* the XSLT style sheets (which generate XSL-FO) that you have developed to convert your XML documents to PDF.

Note

The target audience of this document is a developer or an integrator, that is, a technical person and not an end user. End users, that is persons who need to convert XML documents to a variety of formats, are more likely to use XMLmind XSL Utility, a handy graphical tool, which is available in a separate, self-contained, distribution.

Chapter 2. Installing XMLmind XSL-FO Converter

1. System requirements

- .NET 2.0 *or above* framework.
- Visual J#® 2.0 Redistributable Package.
- 50Mb of free disk space.

XMLmind XSL-FO Converter is officially supported on Windows XP/Vista/7.

2. Installation

Simply unzip the distribution somewhere. Example:

```
C:\> unzip temp\xfc-4_3_2.zip
C:\> dir xfc-4_3_2
...
bin      <DIR> ...
doc      <DIR> ...
legal    <DIR> ...
samples <DIR> ...
```

This means that uninstalling XMLmind XSL-FO Converter simply consists in deleting the directory created by unzipping its distribution.

3. Contents of the installation directory

bin/fo2rtf.exe, fo2wml.exe, fo2docx.exe, fo2odt.exe
Executable files used to run XMLmind XSL-FO Converter.

bin/xfc.dll
The .NET 2.0 assembly containing the XMLmind XSL-FO Converter engine. Reference it in your project if you are integrating XMLmind XSL-FO Converter in your application.

doc/index.html
Points to copies of this document in HTML, PDF, RTF, WordprocessingML, Office Open XML and OpenOffice formats.

Also points to the reference manual of the API of XMLmind XSL-FO Converter.

legal.txt, legal/
Contains XMLmind XSL-FO Converter licences.

samples/
A few XSL-FO sample files, in case you want to test the installation of XMLmind XSL-FO Converter by running samples/make_samples.bat.

Chapter 3. Command-line executables

Four command-line executables are provided: `fo2rtf`, `fo2wml`, `fo2docx` and `fo2odt`, to convert an XSL-FO file to RTF, WML, Open XML and OpenDocument respectively. The general syntax of a command line is:

```
fo2rtf [<options>] <input> [<output>]
```

where `<input>` is the input XSL-FO file name and `<output>` the output file name. If no output file is specified the conversion output is written to the standard output stream. Available options are described below.

`/b <base>`

Specifies the base URL of relative paths in attribute values (typically the `src` attribute of the `external-graphic` element). By default, paths are taken relative to the input source URL.

`/e <encoding>`

Specifies the output encoding. Supported values depend on the target output format:

- For RTF output supported values are `us-ascii` (ASCII), `windows-1252` (Windows Latin-1), `windows-1250` (Windows Eastern European) and `windows-1251` (Windows Cyrillic). The default value is `windows-1252` (Windows Latin-1).
- For WML and OpenDocument output supported values are `us-ascii` (ASCII), `windows-1252` (Windows Latin-1), `windows-1250` (Windows Eastern European), `windows-1251` (Windows Cyrillic) and `utf-8` (UTF-8). The default value is `windows-1252` (Windows Latin-1) for WML output and `utf-8` for OpenDocument output.
- For Open XML output the only supported value is `utf-8` (UTF-8).

Note: for Open XML and OpenDocument output this option specifies the encoding of XML content in the output document.

`/p`

Disables image prescaling. By default images are prescaled to minimize output document size. If this option is specified the original size of images is preserved and scaling directives are inserted in the output document.

`/r <resolution>`

Specifies the source image resolution in dots per inch. This option value is used to compute the intrinsic size of `external-graphic` elements, which determines the actual content size when the `content-height` and `content-width` attributes are not set to absolute values. The default value is 96.

`/s`

Specifies single-sided page layout. By default RTF, WML and Open XML output documents are given a double-sided page layout regardless of the input document properties. This option may be used to force a single-sided page layout.

`/w`

Specifies MS-Word as target RTF viewer. This option may be needed to circumvent an obscure bug in the RTF loader of MS-Word, which does not handle table cell padding tags correctly. When this option is used, XFC will swap top and left padding values in table cells to work around this bug.

Chapter 4. Integrating XMLmind XSL-FO Converter in your application

1. Compiling and running the code samples

All the code samples used to illustrate this document are found in the `samples/dotnet/` subdirectory.

- Compile the two samples by executing `nmake` in the `samples/dotnet/` directory.
- Run the first sample by executing `nmake tsample1` in the `samples/dotnet/` directory.
- Run the second sample by executing `nmake tsample2` in the `samples/dotnet/` directory.

2. Converting an XSL-FO file to RTF

This first sample consists in a single step: invoke XMLmind XSL-FO Converter to convert the input XSL-FO file to RTF.

Note that converting XSL-FO to other formats is simply a matter of changing the value of the `OutputFormat` property. The possible values for this property are: `OutputFormat.Rtf`, `OutputFormat.Wml`, `OutputFormat.Docx`, `OutputFormat.Odt`.

Excerpts of `samples/dotnet/Sample1.cs`:

```
using XmlMind.FoConverter;

...

    Converter converter = new Converter();1
    converter.OutputFormat = OutputFormat.Rtf;2
    converter.OutputEncoding = "windows-1252";
    converter.ImageResolution = 120;

    String inUri = ToUri(inPath);
    converter.SetInput(inUri);3
    converter.SetOutput(outPath);4
    converter.Convert();5

...
```

- 1** Create a new `Converter` object.
- 2** Parameterize the `Converter` using some of its properties.

Note that specifying property `OutputEncoding` is really useful only in the case of the RTF format. All the other formats are XML-based and thus, the default value of `OutputEncoding`, generally UTF-8, should work fine in all cases.

- 3** Specify the input source of the `Converter` using `Converter.SetInput`.

Here we use the most high-level specification: we specify a (%-escaped) URI. In production, you'll generally specify a `Stream`, `TextReader` or `XMLReader`. Note that when you'll specify a `Stream`, `TextReader` or `XMLReader`, the `Converter` will not automatically close it at the end of the conversion. You'll have to do that yourself. The rule here is: the code which has opened a `Stream`, `TextReader` or `XMLReader` has the responsibility to close it.

Note

`TOURI` is a simple helper function implemented as follows:

```
private static string ToUri(String fileName)
{
```

```
Uri uri = new Uri("file:///\" +
                Path.GetFullPath(fileName).Replace('\\', '/'));
return uri.AbsoluteUri;
}
```

- 4 Specify the output destination of the `Converter` using `Converter.SetOutput`.

Here we use the most high-level specification: we specify a file path. In production, you'll generally specify a `Stream` or a `TextWriter`. As explained before, when you'll specify a `Stream` or a `TextWriter`, the `Converter` will not automatically close it at the end of the conversion.

- 5 Perform the conversion by invoking `Converter.Convert`.

3. Converting an XML document to RTF

This second sample consists in three steps:

1. Compile the XSLT style sheet for all subsequent uses.
2. Invoke the XSLT engine to convert the input XML document to XSL-FO.
3. Invoke XMLmind XSL-FO Converter to convert the temporary XSL-FO file generated by second step to RTF.

Excerpts of `samples/dotnet/Sample2.cs`:

```
using System.Xml.Xsl;
using XmlMind.FoConverter;

...

XslCompiledTransform transform = new XslCompiledTransform();
transform.Load(ToUri(xslPath));1

string xmlUri = ToUri(xmlPath);
foPath = Path.GetTempFileName();
transform.Transform(xmlUri, foPath);2

Converter converter = new Converter();3
converter.OutputFormat = OutputFormat.Rtf;
converter.OutputEncoding = "windows-1252";
converter.ImageResolution = 72;
converter.BaseUrl = xmlUri;4

converter.SetInput(ToUri(foPath));
converter.SetOutput(rtfPath);
converter.Convert();5

...
```

- 1 Compile the XSLT style sheet.

About the thread safety of XMLmind XSL-FO Converter

A `Converter` instance must not be shared between different threads.

- 2 Transform the XML input file to a temporary output file created in the system-dependant temporary file directory.
- 3 Create and parameterize a `Converter` object as explained in Section 2, "Converting an XSL-FO file to RTF" [4].
- 4 Setting the `BaseUrl` property to the URL of the XML input file is really needed in our case:

If the XML input file references graphics files using relative URLs (example: `images/screenshot1.png`), then the generated XSL-FO file is likely to contain `fo:external-graphic` objects referencing the same graphics files using the same relative URLs. The problem is that, in our case, the XSL-FO file is not generated in the same directory as the XML input file. Therefore, without the `BaseUrl` property, these relative URLs would be resolved incorrectly by XMLmind XSL-FO Converter.

An advanced alternative to specifying a `BaseUrl` property, is to specify an `IUriResolver` object using `Converter.SetUriResolver`.

- 5 Perform the conversion by invoking `Converter.Convert`.

Chapter 5. Support of the XSL-FO v1.0 standard

1. Features

XFC preserves the structure of source documents, as well as most of the presentation information. Below is a list of key features of XFC.

- Paragraph attributes

Most paragraph attributes (e.g. indentation) are supported. Vertical spacing is handled reasonably in most cases.

- Font attributes

Most font attributes (family, size, weight, etc) are supported.

- Lists

Both bulleted and numbered lists are supported. XFC tries to infer the numbering style from the label of the first list item. If the numbering style cannot be recognized list items are output as plain paragraphs. Nested lists are supported.

- Tables

XFC supports both the fixed and automatic table layout, as well as the two border models defined in the W3C recommendation. The implementation of the collapsing border model does not strictly conform to the CSS2 specification, but should give the expected result in most cases.

- Images

Image support is achieved through the use of external converter classes. XFC comes with a simple converter class that supports the GIF, JPEG and PNG formats. Image scaling is supported.

- Headers and footers

`static-content` elements associated with the `before` and `after` regions are converted to page headers and footers respectively.

- Page references

Page references (`page-number-citation` elements) are supported.

- Hypertext links

Both internal and external links are supported.

For a complete list of supported objects/properties, see the conformance statement [8].

In addition, XFC supports an XSL-FO extension to generate structured document tags [19] (SDTs) in an Office Open XML document. This extension makes it possible to produce simple forms that can be loaded and filled in MS-Word 2007.

2. Limitations

Though XFC implements the greater part of the W3C recommendation, it does not support all XSL-FO features. Below is a list of the current major limitations of XFC.

- The `leader` element is only partly supported.
- The `instream-foreign-object` element is not supported.
- The `float` and `marker` elements are not supported.
- The `writing-mode` property is not supported (value `lr-tb` is assumed).

The conformance level of XFC [8] may be improved in future versions, however it must be stressed that a full conformance cannot be achieved due to the own limitations of its output formats.

3. Conformance statement

The W3C Extensible Stylesheet Language (XSL) v1.0 Recommendation defines three levels of conformance for an XSL-FO processor: basic, extended and complete. Since XMLmind XSL-FO Converter currently does not conform to any of these levels, this document provides a complete list of supported objects/properties, along with additional information for objects/properties that are not fully supported.

In the following tables, the background color (white, light grey or dark grey) of each entry in the tables below indicates the level of conformance (basic, extended or complete) of that particular object/property, as specified by the Recommendation.

Object	Supported	Comments
Declarations and Pagination and Layout Formatting Objects		
root	yes	
declarations	no	
color-profile	no	
page-sequence	yes	
layout-master-set	yes	
page-sequence-master	yes	
single-page-master-reference	yes	
repeatable-page-master-reference	yes	
repeatable-page-master-alternatives	yes	
conditional-page-master-reference	yes	Limited support. See Section 4.3, "Multiple page layouts" [18] for further information.
simple-page-master	yes	
region-body	yes	
region-before	yes	
region-after	yes	
region-start	no	Output format limitation.
region-end	no	Output format limitation.
flow	yes	
static-content	yes	Supported regions: body, before and after.
title	no	
Block-level Formatting Objects		
block	yes	Not supported inside inline-level objects (output format limitation).
block-container	no	Treated like a <code>block</code> object.

Object	Supported	Comments
Inline-level Formatting Objects		
bidi-override	no	
character	no	
initial-property-set	no	
external-graphic	yes	Supported image formats: GIF, JPEG and PNG.
instream-foreign-object	no	
inline	yes	
inline-container	no	
leader	yes	Limited support (most properties ignored). See Section 4.2, "Leaders" [17] for further information.
page-number	yes	
page-number-citation	yes	
Formatting Objects for Tables		
table-and-caption	yes	Not supported inside inline-level objects (output format limitation).
table	yes	
table-column	yes	
table-caption	yes	
table-header	yes	
table-footer	yes	
table-body	yes	
table-row	yes	
table-cell	yes	
Formatting Objects for Lists		
list-block	yes	Not supported inside inline-level objects (output format limitation).
list-item	yes	
list-item-body	yes	
list-item-label	yes	Multiple block-level descendants not supported.
Link and Multi Formatting Objects		
basic-link	yes	Can only contain text and inline-level objects.
multi-switch	no	
multi-case	no	
multi-toggle	no	
multi-properties	no	
multi-property-set	no	
Out-of-line Formatting Objects		
float	no	
footnote	yes	
footnote-body	yes	

Object	Supported	Comments
Other Formatting Objects		
wrapper	yes	
marker	no	
retrieve-marker	no	

Property	Supported	Comments
Common Accessibility Properties		
source-document	no	
role	no	
Common Absolute Position Properties		
absolute-position	no	
top	no	
right	no	
bottom	no	
top	no	
Common Aural Properties		
azimuth	n/a	
cue-after	n/a	
cue-before	n/a	
elevation	n/a	
pause-after	n/a	
pause-before	n/a	
pitch	n/a	
pitch-range	n/a	
play-during	n/a	
richness	n/a	
speak	n/a	
speak-header	n/a	
speak-numeral	n/a	
speak-punctuation	n/a	
speech-rate	n/a	
stress	n/a	
voice-family	n/a	
volume	n/a	
Common Border, Padding and Background Properties		
background-attachment	no	
background-color	yes	
background-image	no	
background-repeat	no	
background-position-horizontal	no	
background-position-vertical	no	

Property	Supported	Comments
border-before-color	yes	<ul style="list-style-type: none"> Not supported on block-level objects that contain other block-level objects (output format limitation). Not supported on inline objects that contain other objects (output format limitation).
border-before-style	yes	
border-before-width	yes	
border-after-color	yes	
border-after-style	yes	
border-after-width	yes	
border-start-color	yes	
border-start-style	yes	
border-start-width	yes	
border-end-color	yes	
border-end-style	yes	
border-end-width	yes	
border-top-color	yes	<ul style="list-style-type: none"> Not supported on block-level objects that contain other block-level objects (output format limitation). Not supported on inline objects that contain other objects (output format limitation).
border-top-style	yes	
border-top-width	yes	
border-bottom-color	yes	
border-bottom-style	yes	
border-bottom-width	yes	
border-left-color	yes	
border-left-style	yes	
border-left-width	yes	
border-right-color	yes	
border-right-style	yes	
border-right-width	yes	
padding-before	yes	<ul style="list-style-type: none"> Not supported on block-level objects that contain other block-level objects (output format limitation). Not supported together with border-*-style="none" or border-*-style="hidden" (output format limitation).
padding-after	yes	
padding-start	yes	
padding-end	yes	
padding-top	yes	<ul style="list-style-type: none"> Not supported on block-level objects that contain other block-level objects (output format limitation). Not supported together with border-*-style="none" or border-*-style="hidden" (output format limitation).
padding-bottom	yes	
padding-left	yes	
padding-right	yes	
Common Font Properties		
font-family	yes	
font-selection-strategy	no	
font-size	yes	

Property	Supported	Comments
font-stretch	no	
font-size-adjust	no	
font-style	yes	Value <code>backslant</code> not supported (output format limitation).
font-variant	yes	
font-weight	yes	
Common Hyphenation Properties		
country	no	
language	no	
script	no	
hyphenate	no	
hyphenation-character	no	
hyphenation-push-character-count	no	
hyphenation-remain-character-count	no	
Common Margin Properties - Block		
margin-top	yes	Percentages and value <code>auto</code> not supported.
margin-bottom	yes	
margin-left	yes	
margin-right	yes	
space-before	yes	Conditionality not supported.
space-after	yes	
start-indent	yes	Percentages not supported.
end-indent	yes	
Common Margin Properties - Inline		
space-end	no	
space-start	no	
Common Relative Position Properties		
relative-position	no	
Area Alignment Properties		
alignment-adjust	no	
alignment-baseline	no	Values <code>middle</code> , <code>before-edge</code> and <code>after-edge</code> supported on external-graphic objects.
baseline-shift	yes	
display-align	no	Supported on <code>table-cell</code> and external-graphic objects.
dominant-baseline	no	
relative-align	no	
Area Dimension Properties		
block-progression-dimension	no	Supported on <code>table-row</code> and external-graphic objects.
content-height	yes	
content-width	yes	

Property	Supported	Comments
height	no	Supported on <code>table-row</code> and <code>external-graphic</code> objects.
inline-progression-dimension	no	Supported on <code>table</code> and <code>external-graphic</code> objects.
max-height	no	
max-width	no	
min-height	no	
min-width	no	
scaling	yes	
scaling-method	no	
width	no	Supported on <code>table</code> and <code>external-graphic</code> objects.
Block and Line-related Properties		
hyphenation-keep	no	
hyphenation-ladder-count	no	
last-line-end-indent	no	Output format limitation.
line-height	yes	Value type <code>space</code> not supported.
line-height-shift-adjustment	no	
line-stacking-strategy	no	
linefeed-treatment	yes	
text-align	yes	Values <code>inside</code> and <code>outside</code> and value type <code>string</code> not supported.
text-align-last	no	Output format limitation.
text-indent	yes	Percentages not supported.
white-space-collapse	yes	
white-space-treatment	yes	
wrap-option	no	
Character Properties		
character	no	
letter-spacing	no	
suppress-at-line-break	no	
text-decoration	yes	
text-shadow	no	
text-transform	no	
treat-as-word-space	no	
word-spacing	no	
Color-related Properties		
color	yes	
color-profile-name	no	
rendering-intent	no	
Float-related Properties		
clear	no	

Property	Supported	Comments
float	no	
intrusion-displace	no	
Keeps and Breaks Properties		
break-after	yes	
break-before	yes	
keep-together	yes	Not supported on block-level objects that contain other block-level objects.
keep-with-next	yes	Not supported on block-level objects that contain other block-level objects.
keep-with-previous	no	
orphans	no	
widows	no	
Layout-related Properties		
clip	no	
overflow	no	
reference-orientation	no	
span	no	
Leader and Rule Properties		
leader-alignment	no	
leader-pattern	yes	Value <code>use-content</code> not supported.
leader-pattern-width	no	
leader-length	no	
rule-style	yes	Supported values: <code>none</code> , <code>dotted</code> and <code>solid</code> .
rule-thickness	no	
Properties for Dynamic Effects Formatting Objects		
active-state	no	
auto-restore	no	
case-name	no	
case-title	no	
destination-placement-offset	no	
external-destination	yes	
indicate-destination	no	
internal-destination	yes	
show-destination	no	
starting-state	no	
switch-to	no	
target-presentation-context	no	
target-processing-context	no	
target-stylesheet	no	
Properties for Markers		
marker-class-name	no	

Property	Supported	Comments
retrieve-class-name	no	
retrieve-position	no	
retrieve-boundary	no	
Properties for Number to String Conversion		
format	yes	
grouping-separator	no	
grouping-size	no	
letter-value	no	
Pagination and Layout Properties		
blank-or-not-blank	no	
column-count	yes	
column-gap	yes	
extent	no	
flow-name	yes	Values <code>xsl-before-float-separator</code> and <code>xsl-footnote-separator</code> not supported.
force-page-count	no	
initial-page-number	yes	
master-name	yes	
master-reference	yes	
maximum-repeats	no	
media-usage	no	
odd-or-even	yes	
page-height	yes	
page-position	yes	Value <code>last</code> not supported.
page-width	yes	
precedence	no	
region-name	yes	
Table Properties		
border-after-precedence	no	
border-before-precedence	no	
border-collapse	yes	Value <code>collapse-with-precedence</code> not supported.
border-end-precedence	no	
border-separation	yes	
border-start-precedence	no	
caption-side	yes	Values <code>start</code> , <code>end</code> , <code>left</code> and <code>right</code> not supported (output format limitation).
column-number	yes	
column-width	yes	
empty-cells	no	
ends-row	yes	

Property	Supported	Comments
number-columns-repeated	yes	
number-columns-spanned	yes	
number-rows-spanned	yes	
starts-row	yes	
table-layout	yes	
table-omit-footer-at-break	no	
table-omit-header-at-break	no	
Writing-mode-related Properties		
direction	no	Value <code>ltr</code> assumed.
glyph-orientation-horizontal	no	
glyph-orientation-vertical	no	
text-altitude	no	
text-depth	no	
unicode-bidi	no	
writing-mode	no	Value <code>lr-tb</code> assumed.
Miscellaneous Properties		
content-type	yes	
id	yes	
provisional-label-separation	yes	
provisional-distance-between-starts	yes	
ref-id	yes	
score-spaces	no	
src	yes	
visibility	no	
z-index	no	
Shorthand Properties		
background	no	Background color specification supported.
background-position	no	
border	yes	See restrictions on individual properties.
border-bottom	yes	
border-left	yes	
border-right	yes	
border-top	yes	
border-color	yes	
border-style	yes	
border-width	yes	
border-spacing	yes	
cue	n/a	
font	yes	
margin	yes	See restrictions on individual properties.

Property	Supported	Comments
padding	yes	See restrictions on individual properties.
page-break-after	yes	See restrictions on individual properties.
page-break-before	yes	
page-break-inside	yes	
pause	n/a	
position	no	
size	no	Value type <code>length</code> supported.
vertical-align	no	
white-space	yes	
xml:lang	no	

4. Implementation specificities

4.1. Page references

4.1.1. RTF/WML/OOXML

Page references - i.e. `page-number-citation` objects - are converted to `PageRef` fields. The values of these fields are not automatically updated when loading an RTF/WML/OOXML document in MS-Word. The easiest way to update all field values is to force a repagination of the document, for instance by switching to the Page Layout view. This will work fine for fields in the body of the document, but not for those in the header/footer. To update fields in the header or footer of a document, proceed as follows:

1. Switch to the Page Layout view.
2. Double-click on an odd page header/footer outline.
3. Type `Ctrl-A` (*Select all*) and `F9` (*Update fields*).
4. Double-click on an even page header/footer outline and repeat step #3.
5. If applicable, double-click on the title page header/footer outline and repeat step #3.

4.1.2. OpenDocument

Page references - i.e. `page-number-citation` objects - are converted to reference fields. The values of these fields are not automatically updated when loading an OpenDocument file in OpenOffice. Select `Update->Fields` in the `Tools` menu to update the field values.

4.2. Leaders

For lack of a corresponding element in the output formats, `leader` objects are implemented by means of tab stops. This is not very convenient given the `leader` object specification, since there is no way for XFC to derive the tab position from the property values. Though XFC will usually set the tab position to a reasonable value by default, this arbitrary position is unlikely to result in the intended layout.

However, the actual tab position may be specified to XFC by setting an additional property on the `leader` object. This property is named `tab-position` and must be defined in the XFC namespace (<http://www.xmlmind.com/fo-converter/xsl/extensions>). The property value is a `<length>` as defined in section 5.11 of the Recommendation. A positive value specifies the tab position relative to the left margin, whereas a negative value specifies the position relative to the right margin.

The code samples below are excerpts from file `$XFC_HOME/xsl/docbook/fo/autotoc.xsl`. They illustrate a typical use of the `tab-position` property in an XSL stylesheet.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:xfc="http://www.xmlmind.com/foconverter/xsl/extensions"
  version='1.0'>
```

```
<fo:leader leader-pattern="dots"
  leader-pattern-width="3pt"
  leader-alignment="reference-area"
  xfc:tab-position="-5mm"
  keep-with-next.within-line="always" />
```

4.3. Multiple page layouts

XFC supports all `conditional-page-master-reference` element combinations that can be accommodated by a single RTF section. This means the following page sequence layouts are supported:

- Single-sided layout.
- Header page + single-sided layout.
- Double-sided layout.
- Header page + double-sided layout.

This applies to all output formats. Also, note that a single RTF section can handle different headers/footers on left/right/first pages, but does not allow page geometry changes, except for switching left and right margins on facing pages. This restriction does not apply to OpenDocument output.

Note: By default RTF, WML and Open XML output documents are given a double-sided page layout regardless of the input document properties. This results in all sections having separate headers/footers for odd and even pages, even though the content of both headers/footers may be identical. It may also result in blank pages being inserted in the document in order for every section to start on an odd page.

4.4. Expressions

Use of expressions for property values specification is supported, subject to the following restrictions:

- The `proportional-column-width` function may not be part of an arithmetic expression, i.e. it must be used as a single primary expression.
- The `rgb-icc`, `system-color`, `system-font` and `merge-property-values` are not supported.

Chapter 6. XSL-FO extension for Office Open XML

1. What it is

XFC supports an XSL-FO extension to generate structured document tags (SDTs) in an Office Open XML document. Structured document tags are WordprocessingML elements that may be used to include form fields - such as text fields and drop-down lists - in an OOXML document and store form data in a dedicated part - called a Custom XML Data part - of the document. In other words, the SDT technology makes it possible to produce simple forms that can be loaded and filled in MS-Word 2007. As Custom XML Data parts are simple XML files the form data can then be easily extracted and processed. For further information regarding structured document tags refer to section 2.5.2 of part 4 (Markup Language Reference) of the Office Open XML specification, available from Ecma International.

2. How it works

To include form fields in an OOXML document one must embed custom elements in the XSL-FO tree. These elements must be in a separate namespace specified by XMLmind. This namespace - referred to by prefix `sdt` in this document - must be declared in the opening tag of the root element of the XSL-FO tree, as shown below.

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:sdt="http://www.xmlmind.com/foconverter/xsl/extensions/docx/sdt">
```

2.1. Text field example

Consider the XSL-FO snippet below:

```
<fo:block margin-left="1cm" margin-right="1cm">Name: <fo:inline
border="solid 1pt blue" padding="1mm"><sdt:text-field binding="name"
prompt="[Enter your name here.]" title="Name" /></fo:inline></fo:block>
```

The `sdt:text-field` element will be converted by XFC to a plain text SDT, which provides the functionality of a basic text field. The `prompt` attribute specifies placeholder text to be initially displayed in the field. The `sdt:text-field` element is wrapped in an `fo:inline` object that carries presentation properties. The initial display of the whole block in MS-Word 2007 is shown below. The next image shows the appearance of the field when selected, and the last one shows the field once filled.

Figure 6.1. Text field (initial display)

Name:

Figure 6.2. Text field (selected)

Name:

Figure 6.3. Text field (filled)

Name:

The `binding` attribute of the `sdt:text-field` element establishes the mapping between the field and an XML element in the Custom XML Data part. In the simplest case the value of this attribute is an XML element name. The Custom XML Data part will be automatically generated by XFC, in the form of a simple XML instance where all elements associated with form fields are children of the root element. Assuming the document contains no other field, XFC will therefore generate the XML instance below:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name></name>
</root>
```

When saving the document after an editing session MS-Word will store the current value of the field as the content of the `name` element in the Custom XML Data part, as shown below.

```
<?xml version="1.0" encoding="UTF-8"?><root>
  <name>John Smith</name>
</root>
```

2.2. Drop-down list example

Consider the XSL-FO snippet below:

```
<fo:block margin-left="1cm" margin-right="1cm">Favorite Animal:
<fo:inline border="solid 1pt blue" padding="1mm"><sdt:drop-down-list
  binding="favorite-animal" initial-value="cat"
  title="Favorite Animal">
  <sdt:list-entry value="cat" />

  <sdt:list-entry value="dog" />

  <sdt:list-entry value="hamster" />
</sdt:drop-down-list></fo:inline></fo:block>
```

The `sdt:drop-down-list` element will be converted by XFC to a drop-down list SDT, which provides the ability to select a single value from a predefined list. The list entries are specified by the `sdt:list-entry` children. The `initial-value` attribute of the `sdt:drop-down-list` element specifies the initial value of the field. The initial display of the whole block in MS-Word 2007 is shown below. The next image shows the appearance of the field while selecting an entry in the list.

Figure 6.4. Drop-down list (initial display)

Favorite Animal:

Figure 6.5. Drop-down list (selecting an entry)

Favorite Animal:

The `initial-value` attribute differs from the `prompt` attribute in that the specified value is initially stored in the Custom XML Data part. Assuming the document contains no other field, XFC will therefore generate the Custom XML Data part below:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <favorite-animal>cat</favorite-animal>
</root>
```

2.3. Specifying a Custom XML Data template

Sometimes it may be desirable to have form data stored in an XML instance more complex than the default instance generated by XFC. In this case a Custom XML Data template may be specified by inserting an `sdt:configuration` element before the first `fo:page-sequence` object in the XSL-FO tree, e.g.:

```
<sdt:configuration custom-xml-template="custom.xml" />
```

The `custom-xml-template` attribute specifies the URL of an XML template to be used as the initial content of the Custom XML Data part. This XML template must be encoded in UTF-8 or UTF-16.

When a Custom XML Data template is specified, the `binding` attribute of a form field associated with an XML element in the Custom XML Data part references that particular element by means of an XPath 1.0 expression. For instance, consider the XML template below:

```
<?xml version="1.0" encoding="UTF-8"?>
<order>
  <product>
    <reference />
    <quantity />
  </product>
  <product>
    <reference />
    <quantity />
  </product>
</order>
```

To associate the `reference` child of the first `product` element with a form field one would set the `binding` attribute value of that field to `/order/product[1]/reference`. Moreover, when a Custom XML Data template is specified the `initial-value` attribute of form fields is ignored. If a field is to be initialized the initial value must be stored in the Custom XML Data template as the content of the XML element associated with that field.

2.4. Extracting the Custom XML Data part

Office Open XML documents are basically ZIP archives, so the Custom XML Data part can be easily extracted. In accordance with MS-Word's naming scheme XFC stores the Custom XML Data part in ZIP entry `custom-xml/item1.xml`.

3. Reference Material

This section provides a comprehensive description of the custom elements that make up the XSL-FO extension for Office Open XML. These elements must be in a separate namespace specified by XMLmind. This namespace - referred to by prefix `sdt` in this document - must be declared in the opening tag of the root element of the XSL-FO tree, as shown below.

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
        xmlns:sdt="http://www.xmlmind.com/foconverter/xsl/extensions/docx/sdt">
```

There are five elements that translate into a form field:

- `sdt:text-field`
- `sdt:drop-down-list`
- `sdt:combo-box`
- `sdt:date`
- `sdt:picture`

These are inline-level elements that may appear anywhere inline-level Formatting Objects are allowed.

3.1. Generic attributes

The attributes described below apply to all form fields, except for the `initial-value` and `prompt` attributes that do not apply to the `sdt:picture` element.

- `binding`

This attribute establishes the mapping between a field and an XML element in the Custom XML Data part. In the simplest case the value of this attribute is an XML element name. The Custom XML Data part will be automatically generated by XFC, in the form of a simple XML instance where all elements associated with form fields are children of the root element. When a Custom XML Data template is specified the attribute value is an XPath 1.0 expression that identifies the XML element associated with the field. If this attribute is omitted no mapping is established.

- `editable`

This attribute specifies whether or not the field content is editable. Possible values are `true` (default) and `false`.

- `initial-value`

This attribute specifies the initial value of the field. The specified value will be stored in the Custom XML Data part, unless a Custom XML Data template is in use. (This attribute has no effect if a Custom XML Data template has been specified. In this case the initial value must be stored in the Custom XML Data template as the content of the XML element associated with the field.)

- `locked`

This attribute specifies whether or not the field is locked. Possible values are `true` (default) and `false`. (The feature of a locked field is that it cannot be deleted from the document.)

- `prompt`

This attribute specifies placeholder text to be initially displayed in the field if no initial value is provided. (If both the `prompt` and `initial-value` attributes are specified the latter will take precedence.)

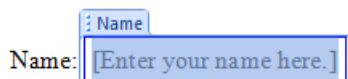
- `title`

This attribute specifies the field title. This title is displayed as part of the field outline when the field is selected. The default value is specific to each field type.

3.2. `sdt:text-field`

This element is converted to a plain text SDT, which provides the functionality of a basic text field.

Figure 6.6. Text field



Attributes:

- `binding`

See generic attributes.

- `editable`

See generic attributes.

- initial-value

See generic attributes.

- locked

See generic attributes.

- multi-line

This attribute specifies whether or not line breaks are allowed in the field value. Possible values are `true` and `false` (default).

- prompt

See generic attributes.

- title

See generic attributes. (The default value is `Text Field`).

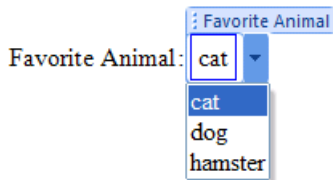
Content model:

EMPTY

3.3. sdt:drop-down-list

This element is converted to a drop-down list SDT, which provides the ability to select a single value from a pre-defined list.

Figure 6.7. Drop-down list



Attributes:

- binding

See generic attributes.

- editable

See generic attributes.

- initial-value

See generic attributes.

- locked

See generic attributes.

- prompt

See generic attributes.

- title

See generic attributes. (The default value is `Drop-Down List`).

Content model:

```
(sdt:list-entry)+
```

3.4. sdt:list-entry

This element specifies an entry in the list of possible values of a drop-down list or combo box SDT.

Attributes:

- display-text

This attribute specifies alternative text to be displayed when this entry is selected. (By default the actual entry value is displayed.)

- value

This attribute specifies the actual entry value. This is the value that will be stored in the Custom XML Data part when this entry is selected. This attribute is required. (The `sdt:list-entry` element is ignored if this attribute is omitted.)

Content model:

```
EMPTY
```

3.5. sdt:combo-box

This element is converted to a combo box SDT, which combines a text field and a drop-down list.

Attributes:

- binding

See generic attributes.

- editable

See generic attributes.

- initial-value

See generic attributes.

- locked

See generic attributes.

- prompt

See generic attributes.

- title

See generic attributes. (The default value is `Combo Box`).

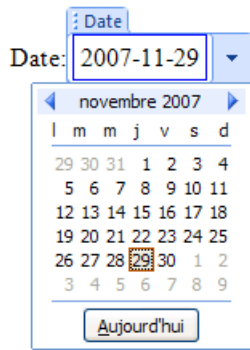
Content model:

```
(sdt:list-entry)+
```

3.6. sdt:date

This element is converted to a date SDT, which is a text field with date semantics. This SDT provides a date picker for fast and secure input, though a date value may be typed in as well.

Figure 6.8. Date



Attributes:

- binding

See generic attributes.

- editable

See generic attributes.

- format

This attribute specifies the date format. (This format is used by the date picker but is not enforced when a value is typed in directly.) The attribute value is a character string in which the following variables are recognized:

Variable	Expanded Value
%D	day of month (01-31)
%M	month (01-12)
%Y	year (4 digits)
%y	year (last 2 digits)

The default value is %Y-%M-%D.

- initial-value

See generic attributes.

- locked

See generic attributes.

- prompt

See generic attributes.

- title

See generic attributes. (The default value is `Date`).

Content model:

EMPTY

3.7. sdt:picture

This element is converted to a picture SDT, which provides the ability to select, display and edit images. The value of this field - stored as the content of the associated XML element in the Custom XML Data part - is the Base64-encoded image data.

Figure 6.9. Picture



Attributes:

- binding
See generic attributes.
- editable
See generic attributes.
- locked
See generic attributes.
- title
See generic attributes. (The default value is `Picture`).

Content model:

(`sdt:image-data`)?

3.8. sdt:image-data

This element specifies the initial value of an `sdt:picture` element. It contains the Base64-encoded image data to be initially displayed in the picture SDT. If this element is omitted an image placeholder will be displayed. This placeholder includes a button to open an image selection dialog.

Attributes:

- format
This attribute specifies the image data format, in the form of a MIME type. Supported formats are GIF (`image/gif`), JPEG (`image/jpeg`) and PNG (`image/png`). This attribute is required. (The `sdt:image-data` element is ignored if this attribute is omitted.)

Content model:

#PCDATA

3.9. sdt:configuration

This element specifies optional parameters related to the Custom XML Data part. If this element is present in the XSL-FO tree it must occur before the first `fo:page-sequence` object.

Attributes:

- custom-xml-template

This attribute specifies the URL of an XML template to be used as the initial content of the Custom XML Data part. This XML template must be encoded in UTF-8 or UTF-16. The URL is resolved by XFC using its current URI resolver.

- prefix-mappings

This attribute specifies the mapping of namespace prefixes used in XPath expressions that identify an element in a Custom XML Data template. The attribute value is a list of namespace declarations separated by white space. This attribute is required if the Custom XML Data template makes use of namespaces. For instance, consider the XML template below:

```
<?xml version="1.0" encoding="UTF-8"?>
<order xmlns="http://www.xmlmind.com/ns/order">
  <product>
    <reference />
    <quantity />
  </product>
</order>
```

As this template contains a namespace declaration, names in XPath expressions that identify an element in the template should be qualified. For this purpose one would set the `prefix-mappings` attribute and use the so declared namespace prefix to qualify element names in XPath expressions, as shown below.

```
<sdt:configuration
  custom-xml-template="custom.xml"
  prefix-mappings="xmlns:ns="http://www.xmlmind.com/ns/order" />

<sdt:text-field binding="/ns:order/ns:product/ns:reference"
  prompt="[Enter product reference.]" title="Reference" />
```

Content model:

EMPTY

Appendix A. Notice

The .NET version of XMLmind XSL-FO Converter may incorporate intellectual property owned by Microsoft Corporation. The terms and conditions upon which Microsoft is licensing such intellectual property may be found at <http://msdn.microsoft.com/library/en-us/odcXMLRef/html/odcXMLRefLegalNotice.asp>.