
XMLmind XSL-FO Converter v4.2p1 - User's Guide

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

February 22, 2008

Table of Contents

Introduction	1
Features	2
Features	2
Limitations	3
Installation	3
System requirements	3
Installation	3
Contents of the distribution	4
Command-line executables	5
Java™ version	5
.NET version	6
Implementation notes	7
Page references	7
Leaders	8
Special characters	8
Multiple page layouts	8
Expressions	9
XSL-FO extension for Office Open XML	9
What it is	9
How it works	9
Reference Material	11
XSL Utility	17
Modification history	18
Notice	25

Introduction

XMLmind XSL-FO Converter (XFC for short) is a software component designed to convert XSL Formatting Objects (XSL-FO) to various formats compatible with popular word processors. Currently supported formats are RTF, WML, Office Open XML and OpenDocument. The RTF output conforms to version 1.6 of the specification and is compatible with MS-Word 2000+. The WML output is compatible with MS-Word 2003+, while the Office Open XML output is compatible with MS-Word 2007 only. The OpenDocument output is compatible with OpenOffice 2.0.2+. All formats preserve the structure of the original document and most of the presentation information (font attributes, indentation, etc).

XMLmind XSL-FO Converter is available in two versions, targeted at the Java™ Platform and the .NET Framework respectively. Each version comes in two forms:

Personal Edition

Personal Edition is intended for individuals. It may be freely used by natural persons for their personal activities. (See file `license.txt` for conditions of use.) Personal Edition comes in binary form, i.e. it does not provide access to the converter API. Both the Java and .NET versions provide command-line executables, and the Java version also offers a graphical user interface. Personal Edition is available for download [<http://www.xmlmind.com/foconverter/download.shtml>] at no charge.

Professional Edition

Professional Edition is required for corporate or otherwise professional use. In addition to the command-line executables and graphical user interface, it provides access to the converter API, therefore allowing seamless integration of XFC into a Java or .NET application.

Features

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 [<http://www.xmlmind.com/foconverter/conformance.html>].

In addition, XFC supports an XSL-FO extension to generate structured document tags (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.

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 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.

Installation

System requirements

Java™ version	XFC only requires a Java runtime 1.3 or above. All additional libraries (JAXP [http://java.sun.com/xml/jaxp], SAX2 [http://www.saxproject.org], and XP for JAXP [http://www.xmlmind.com/xpforjaxp.html]) are included in the distribution for convenience.
.NET version	XFC requires the .NET Framework Redistributable Package (version 1.1 or 2.0) and the Microsoft Visual J# Redistributable Package (same version) to be installed on the host machine. These packages are available for download [http://msdn.microsoft.com/netframework/downloads/updates/default.aspx] on the MSDN [http://msdn.microsoft.com/] site.

Installation

Unix

The installation procedure on a Unix system is outlined below.

1. Unpack the distribution:

```
$ gunzip -c xfc-42p1.tgz | tar xvf -
```

The above command will create directory `xfc-42p1` in the current directory.

2. Add XFC installation directory (referred to as `$XFC_HOME`) to your `PATH` variable, or create symbolic links to `$XFC_HOME/fo2rtf`, `$XFC_HOME/fo2wml`, `$XFC_HOME/fo2docx` and `$XFC_HOME/fo2odt` in a directory already in your `PATH` variable (e.g. `/usr/local/bin`, assuming you have the required privileges):

```
$ cd /usr/local/bin
$ ln -s $XFC_HOME/fo2rtf fo2rtf
$ ln -s $XFC_HOME/fo2wml fo2wml
$ ln -s $XFC_HOME/fo2docx fo2docx
$ ln -s $XFC_HOME/fo2odt fo2odt
```

XFC may then be invoked with one of the following commands:

```
$ fo2rtf <fo-file> <rtf-file>
$ fo2wml <fo-file> <wml-file>
$ fo2docx <fo-file> <docx-file>
$ fo2odt <fo-file> <odt-file>
```

You may try to convert one of the sample files in the `samples` directory to check the installation.

Windows

The installation procedure on a Windows platform is outlined below.

1. Unpack the distribution. The Windows distribution comes in the form of a setup program (`setup.exe`). Run this program and follow the instructions to perform the installation.
2. Add XFC installation directory to your PATH variable. XFC may then be invoked with one of the following commands:

```
C:\> fo2rtf <fo-file> <rtf-file>
C:\> fo2wml <fo-file> <wml-file>
C:\> fo2docx <fo-file> <docx-file>
C:\> fo2odt <fo-file> <odt-file>
```

You may try to convert one of the sample files in the `samples` directory to check the installation.

Contents of the distribution

Java™ version

<code>docs/apidoc/</code>	Contains the API documentation.
<code>docs/docbook-xsl/</code> , <code>docs/slides-xsl</code>	Contains Norman Walsh's DocBook and Slides XSL stylesheets documentation.
<code>docs/userguide/</code>	Contains this document in HTML and PDF formats.
<code>docs/xslutility/</code>	Contains XSL Utility user's guide in HTML format.
<code>fo2docx</code>	Unix script to convert an XSL-FO file to Open XML.
<code>fo2docx.bat</code>	Windows script to convert an XSL-FO file to Open XML.
<code>fo2odt</code>	Unix script to convert an XSL-FO file to OpenDocument.
<code>fo2odt.bat</code>	Windows script to convert an XSL-FO file to OpenDocument.
<code>fo2rtf</code>	Unix script to convert an XSL-FO file to RTF.
<code>fo2rtf.bat</code>	Windows script to convert an XSL-FO file to RTF.
<code>fo2wml</code>	Unix script to convert an XSL-FO file to WML.
<code>fo2wml.bat</code>	Windows script to convert an XSL-FO file to WML.
<code>java/</code>	Contains the Java source tree (Professional Edition, Developer and Site License only).
<code>lib/</code>	Contains the additional Java libraries needed to run XFC, as well as general and copyright information regarding these libraries.
<code>license.txt</code>	XFC license agreement (Personal Edition).
<code>license_dev.txt</code> , <code>license_site.txt</code> , <code>license_srv.txt</code>	XFC license agreements (Professional Edition).
<code>samples/</code>	Contains XSL-FO sample files. (Most of these files are simple test cases which are part of the XFC test suite.)
<code>xfc.jar</code>	XFC and XSL Utility binary class library. (Note: the Personal Edition library is obfuscated to prevent access to the converter API.)

xslu/config/	Contains the XML catalog, DTDs and XSL stylesheets bundled with XSL Utility.
xslu/lib/	Contains the additional Java libraries needed to run XSL Utility, as well as general and copyright information regarding these libraries.
xslu/samples/	Contains XML (DocBook) sample files.
xslutility	Unix script to run XSL Utility (graphical user interface).
xslutility.bat	Windows script to run XSL Utility (graphical user interface).
xslutil	Unix script to run XSL Utility (command-line executable).
xslutil.bat	Windows script to run XSL Utility (command-line executable).

.NET version

cs/	Contains the C# source code (Professional Edition, Developer and Site License only).
docs/apidoc/	Contains the API documentation.
docs/userguide/	Contains this document in HTML and PDF formats.
fo2docx.exe	Command-line executable to convert an XSL-FO file to Open XML.
fo2odt.exe	Command-line executable to convert an XSL-FO file to OpenDocument.
fo2rtf.exe	Command-line executable to convert an XSL-FO file to RTF.
fo2wml.exe	Command-line executable to convert an XSL-FO file to WML.
license.txt	XFC license agreement (Personal Edition).
license_dev.txt, license_site.txt, license_srv.txt	XFC license agreements (Professional Edition).
js/	Contains the J# source code (Professional Edition, Unrestricted Use License only).
samples/	Contains XSL-FO sample files. (Most of these files are simple test cases which are part of the XFC test suite.)
xfc.dll	XFC binary class library. (Note: the Personal Edition library is obfuscated to prevent access to the converter API.)

Command-line executables

Java™ version

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. Options are specified as:

```
-<name>=<value>
```

where `<name>` is the option name and `<value>` the option value. Option names and values are described below.

<code>baseURL</code>	Specifies the base URL of relative paths in attribute values (typically the <code>src</code> attribute of the <code>external-graphic</code> element). By default, paths are taken relative to the input source URL.
<code>imageResolution</code>	Specifies the source image resolution in dots per inch. This option value is used to compute the intrinsic size of <code>external-graphic</code> elements, which determines the actual content size when the <code>content-height</code> and <code>content-width</code> attributes are not set to absolute values. The default value is 96.
<code>outputEncoding</code>	Specifies the output encoding. Supported values depend on the target output format: <ul style="list-style-type: none">• For RTF output supported values are <code>ASCII</code>, <code>Cp1250</code> (Windows Eastern European), <code>Cp1251</code> (Windows Cyrillic) and <code>Cp1252</code> (Windows Latin-1). The default value is <code>Cp1252</code> (Windows Latin-1).• For WML output all encodings available in the current JVM are supported. The option value may be either the encoding name (e.g. <code>ISO8859_1</code>) or the charset name (e.g. <code>ISO-8859-1</code>). The default value is <code>Cp1252</code> (Windows Latin-1).• For Open XML output this option specifies the encoding of XML content in the output document. Supported values are <code>UTF-8</code> and <code>UTF-16</code>. The default value is <code>UTF-8</code>.• For OpenDocument output this option specifies the encoding of XML content (files <code>styles.xml</code> and <code>content.xml</code>) in the output document. All encodings available in the current JVM are supported. The option value may be either the encoding name (e.g. <code>ISO8859_1</code>) or the charset name (e.g. <code>ISO-8859-1</code>). The default value is <code>UTF8</code>.
<code>prescaleImages</code>	Specifies image scaling policy. By default images are prescaled to minimize output document size. If this option is set to <code>false</code> the original size of images is preserved and scaling directives are inserted in the output document.
<code>rtf.target</code>	Specifies the target RTF viewer. Currently the only supported value is <code>MSWord</code> . 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 set to <code>MSWord</code> , XFC will swap top and left padding values in table cells to work around this bug.
<code>singleSidedLayout</code>	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 set to <code>true</code> to force a single-sided page layout.

.NET version

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.

<code>/b <base></code>	Specifies the base URL of relative paths in attribute values (typically the <code>src</code> attribute of the <code>external-graphic</code> element). By default, paths are taken relative to the input source URL.
<code>/e <encoding></code>	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.

Implementation notes

Page references

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.

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.

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"/>
```

Special characters

Java™ version

Handling of characters that cannot be represented in the output encoding depends on the version of the Java runtime in use. With JRE 1.3 no special processing takes place so these characters are replaced with question marks. (This is the standard Java streams behaviour.) With JRE 1.4 XFC uses a character set encoder - an instance of the `java.nio.charset.CharsetEncoder` class - to determine if a given character can be represented in the output encoding. Characters that cannot be encoded are then represented using a Unicode control word (RTF output) or an XML character reference (WML, Open XML and OpenDocument output). However, not all charset encoders are available on all platforms. (For instance, encoders for the `windows-1250` and `windows-1251` charsets are not available in Linux JRE 1.4.0.)

.NET version

XFC uses an instance of the `System.Text.Encoding` class to determine if a given character can be represented in the output encoding. Characters that cannot be encoded are then represented using a Unicode control word (RTF output) or an XML character reference (WML, Open XML and OpenDocument output).

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.

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.

XSL-FO extension for Office Open XML

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 [<http://www.ecma-international.org/>].

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">
```

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 1. Text field (initial display)

Name:

Figure 2. Text field (selected)

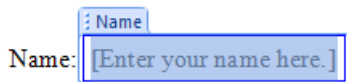
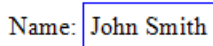


Figure 3. Text field (filled)



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>
```

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 4. Drop-down list (initial display)

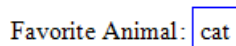
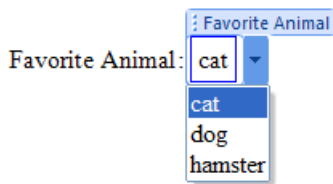


Figure 5. Drop-down list (selecting an entry)



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>
```

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 [<http://www.w3.org/TR/xpath>] 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.

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`.

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.

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 [<http://www.w3.org/TR/xpath>] 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.)

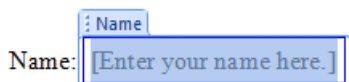
- `title`

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.

`sdt:text-field`

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

Figure 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`).

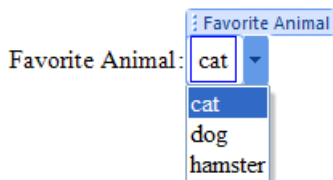
Content model:

EMPTY

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 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)+`

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

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`).

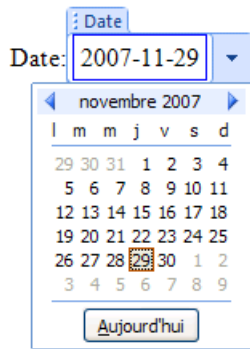
Content model:

`(sdt:list-entry)+`

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 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`).

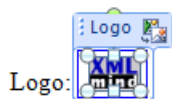
Content model:

EMPTY

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 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)?`

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

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

XSL Utility

XSL Utility is a graphical tool that was primarily designed to make evaluation of XFC easier. It is currently available only with the Java™ version of XFC. XSL Utility can be used to convert XML documents to RTF, WML, Open XML and OpenDocument, as well as additional formats such as PostScript and PDF. (For formats not supported by XFC, XSL Utility relies on the Apache FOP [<http://xml.apache.org/fop/>] FO processor.) XSL Utility includes:

- Norman Walsh's XSL stylesheets [<http://docbook.sourceforge.net/>] for DocBook, Simplified DocBook and Slides
- Michael Kay's Saxon [<http://saxon.sourceforge.net/>] XSLT engine
- Apache FOP [<http://xml.apache.org/fop/>] FO processor

Note: XSL Utility requires a Java runtime 1.4 or above.

To launch XSL Utility from a terminal emulation window, move to the XFC installation directory and enter `xslutility` at the command prompt.

To launch XSL Utility from your filesystem browser, move to the XFC installation directory and open file `xfc.jar`.

If an auto-installable distribution was used for the installation, you can also launch XSL Utility directly from the 'XMLmind XSL-FO Converter' group in the Programs menu.

For details on how to use XSL Utility, refer to the on-line help provided by the application. Sample XML (DocBook) files are included in directory `xslu/samples`.

XSL Utility is also available as a command-line executable (`xslutil`) that allows batch processing of XML files. For usage information enter `xslutil` (with no argument) at the command prompt.

Modification history

Version 4.2p1 (02/22/08)

Bug fixes

- Unsupported XSL-FO extensions were not handled properly, possibly resulting in a `NullPointerException`. (This bug was introduced in version 4.2...)

Bug fixes (RTF)

- Right border on table cells spanning multiple columns was not handled properly, possibly resulting in a missing border.
- Implicit empty cells in a nested table would result in invalid RTF output.

Bug fixes (WML)

- Right border on table cells spanning multiple columns was not handled properly, possibly resulting in a missing border.
- Characters '-' and '.' in `id` property values would possibly result in invalid hyperlink targets. (These characters cannot be used in bookmark names as MS-Word replaces them with character '_' while leaving hyperlink targets unchanged...)
- Footnote body was lost if the `id` property was set on the `inline` child of the `footnote` object.

Bug fixes (OOXML)

- Characters '-' and '.' in `id` property values would possibly result in invalid hyperlink targets. (These characters cannot be used in bookmark names as MS-Word replaces them with character '_' while leaving hyperlink targets unchanged...)

Version 4.2 (12/13/07)

New features

- XSL-FO extension to generate structured document tags (SDTs) in an Office Open XML document.

Bug fixes (WML)

- Character '>' was not escaped in string "]]>", resulting in invalid XML content.

Bug fixes (OOXML)

- Character '>' was not escaped in string "]]>", resulting in invalid XML content.
- Page number format was not properly specified in section properties.

Bug fixes (OpenDocument)

- Character '>' was not escaped in string "]]>", resulting in invalid XML content.
- Images in headers/footers did not show in OpenOffice, due to missing namespace declarations in file `styles.xml`.

XSL Utility

- Use of XML catalogs for resolution of URIs in XSLT stylesheets.

Version 4.1 (08/23/07)

Enhancements

- Support of image viewport.
- New property `imageResolution` to specify source image resolution.
- New property `prescaleImages` to specify image scaling policy.

Version 4.0 (06/05/07)

New license terms (Personal Edition).

New features

- Support of Open XML as alternate output format.

Enhancements

- New property `singleSidedLayout` to force single-sided page layout.

XSL Utility

- Upgraded FOP to version 0.93.
- Upgraded DocBook-XSL to version 1.71.1.
- Added DTD for DocBook v4.5.

Version 3.1 (01/18/07)

New licensing scheme (Professional Edition).

Enhancements

- Improved automatic table layout provides better handling of very long words in table cells.

Bug fixes (RTF)

- Images did not show up in MS-Word 2007. (Not a bug actually, but rather a flaw in the RTF loader of Office 2007...)

Bug fixes (RTF/WML)

- All bookmarks supposed to be set on a list item were lost along the way, which would possibly result in broken links in the output document.

Version 3.0p1 (10/18/06)

Bug fixes

- XFC did not handle the `id` property on `wrapper` objects, which would possibly result in broken links in the output document.

Bug fixes (WML)

- Different headers/footers on odd and even pages were not handled properly. (Actually odd and even headers/footers were properly specified in the output document but even headers/footers were meaningless due to a missing element (`w:evenAndOddHeaders`) in the document properties.)

Bug fixes (OpenDocument)

- The automatic table layout would possibly yield an oversized table if the table width was not explicitly specified.
- The page geometry was incorrect if no top/bottom margin was specified.

XSL Utility

- Added a confirmation dialog before transformation removal.

Version 3.0 (09/29/06)

New features

- Support of OpenDocument as alternate output format.

Version 2.3p1 (03/28/06)

Bug fixes

- A table in a list item inside a table cell would not be handled properly, resulting in incorrect RTF/WML output.
- Some particular page sequence definitions would result in missing header/footer on even pages. (This happened for instance when converting a DocBook document with recent versions of DocBook-XSL, unless the `double.sided` parameter was set to '1'.)

XSL Utility

- Fixed a bug in `xslutil`: conversion to RTF/WML would possibly fail when the input file was specified as a relative path name.
- Added DTD for DocBook v4.4.
- Upgraded DocBook-XSL to version 1.69.1.

Version 2.3 (03/02/06)

First .NET version.

Enhancements

- Revised and extended API (Professional Edition).

Bug fixes

- Characters U+00AD (soft hyphen) and U+2011 (non-breaking hyphen) were not handled properly.

Bug fixes (RTF)

- The font table writing did not strictly conform to the RTF specification. This syntax error would make RTF documents unreadable by TextEdit and possibly other Mac OS X applications.

XSL Utility

- Upgrade of Saxon to version 6.5.4.
- New transformation XHTML to RTF (XSLT stylesheet by Antenna House).

Version 2.2p1 (09/20/05)

Enhancements

- Better handling of white space at the beginning/end of a paragraph.
- Fixed issue with characters '-' and '.' in id property values. (The RTF loader of MS-Word replaces these characters with '_' (underscore) in bookmark names.)

Bug fixes

- List item labels containing non-textual objects (e.g. images) were not handled properly. (This bug was introduced with version 2.1.)
- The body of a footnote was ignored if the footnote element was a child of another inline-level element.
- Bad shorthand property values (e.g. border="1") would possibly result in a `ClassCastException`.

Version 2.2 (05/25/05)

Enhancements

- Use of expressions for property values specification is now supported.
- The header and footer offsets (RTF `\headery` and `\footery` control words) are now set according to the page master `margin-top` and `margin-bottom` property values.

Bug fixes (RTF)

- Failure to access the URL (`src` property value) of an `external-graphic` object would possibly result in a `NullPointerException`. (This bug affects version 2.1 only.)
- Setting the `space-before` property on a nested table would result in invalid RTF output.

XSL Utility

- Upgrade of DocBook-XSL to version 1.68.1.

Version 2.1 (03/18/05)

Enhancements

- Support of nested tables.

- Automatic switch to the fixed table layout when all column widths are specified.

Bug fixes (RTF)

- The charset encoder used to determine if a given character can be represented in the output encoding was a class variable, which could possibly cause an `IllegalStateException` in a multi-threaded environment.

Bug fixes (WML)

- XML special characters (e.g. '&') were not escaped in the `w:dest` attribute values.

Version 2.0 (10/20/04)

New features

- Support of WML as alternate output format.

Bug fixes

- When a block contained character data followed by a table, the character data before the table would end up inside the first cell of the table.
- Values of some compound properties (e.g. `border-separation`) were not properly evaluated.

Version 1.3 (05/28/04)

New features

- Support of multiple page layouts (e.g. different headers and/or footers on left and right pages).

Enhancements

- Support of justified text in the body of list items.

Bug fixes

- XFC would occasionally hang while processing an `external-graphic` object in some particular environments (e.g. on Windows platforms with JRE 1.3).
- The `margin-left` and `margin-right` properties were not considered in the computation of text indents.

XSL Utility

- Upgrade of DocBook-XSL to version 1.65.1.
- The Jimi image library is now included in the distribution.

Version 1.2 (12/03/03)

Enhancements

- Support of the `proportional-column-width` function.
- Support of the `text-align` property on the `table-and-caption` element.
- Support of the `keep-with-next` property on table rows (MS-Word compatible implementation).
- Handling of vertical space before a table. (Implemented by means of an empty paragraph.)

- Use of the RTF Unicode control word (`\u`) for characters that cannot be represented in the output encoding. (Requires JRE 1.4+.)

Bug fixes

- A bookmark (`id` property) attached to the last block of a document was lost if the block contained no character data.
- Particular values of the `border`, `border-top`, `border-bottom`, `border-left` and `border-right` property (e.g. `border-top="solid"`) would cause a `NullPointerException`. (Was supposed to be fixed since version 1.0p1...)
- A `list-item-body` element with a `list-block` as its first child was not handled properly, resulting in weird paragraph layout.

XSL Utility

- Upgrade of DocBook-XSL to version 1.62.0.
- Drag & Drop support enhancement.
- Command-line utility for batch processing.

Version 1.1 (03/19/03)

New features

- Support of page references (`page-number-citation` element).
- Partial support of leaders (`leader` element).
- Support of hypertext links (`external-destination` and `internal-destination` properties).
- New image converter based on the Java Image I/O library.
- New Java property `rtf.target`.

Bug fixes

- Paragraph attributes were not reset inside an empty table cell, resulting in bad rendering by MS-Word whenever the very first cell of a table was empty.

XSL Utility

- Upgrade to FOP 0.20.4 and DocBook-XSL 1.60.1.

Version 1.0p1 (10/31/02)

Enhancements

- Revised and extended API (Professional Edition).

Bug fixes

- Particular values of some shorthand properties would cause a `NullPointerException`. This bug would show up for instance when the value of the `border`, `border-top`, `border-bottom`, `border-left` or `border-right` property did not specify the border color, e.g. `border-top="solid"`.
- A bad property value would possibly result in additional properties of the current object not being evaluated.

- The background color of an outer block would not propagate to inner blocks.
- SAX exceptions were not handled properly, causing a `NullPointerException` when trying to convert an ill-formed document.

Version 1.0 (07/26/02)

First commercial version.

Enhancements

- Support of keeps and breaks properties in tables.
- Support of the `scale-to-fit` value of the `content-width/content-height` properties.

Bug fixes

- The `space-after` and `break-after` properties were not handled properly.
- Image scaling was inaccurate when the `content-width` or `content-height` property was specified as a `<length>` value.

Version 1.0b1 (06/05/02)

New features

- Automatic table layout.
- Collapsing border model.
- Support of the `static-content` (before and after regions), `footnote` and `page-number` elements.

Enhancements

- Support of lists in table cells.
- Full implementation of the separated border model.
- Partial support of keeps and breaks properties.
- Support of the `baseline-shift` property.
- Support of the background color and border attributes of inline-level elements.
- Better selection of a page master among alternatives.
- Use of the bullet character (`\u2022`) as default label in bulleted lists.

Bug fixes

- The font attributes of `basic-link` elements were ignored.
- The `monospace` generic family was not bound to an actual monospaced font.
- The list numbering type specification was not compatible with Word 97.
- White space between two inline-level elements was entirely discarded.
- The initial value of the `column-number` property did not consider row spans, possibly causing an incorrect table layout in some situations.

- The initial value of the `border-*-color` properties was not properly set, possibly causing a `NullPointerException` in some situations.
- XSL-FO extensions - e.g. `fox` - were not handled properly, resulting in an `ArrayIndexOutOfBoundsException`.

Version 0.9 (03/04/02)

Initial release.

Notice

This product 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>.