
xfront 0.3.3

Armin Sander

Table of Contents

1. Introduction	1
2. Syntax	1
2.1. Elements	1
2.2. Attributes	3
3. xfront command	3
4. Quick help	3
A. Installation	4
B. Required features for version 1.0	4
C. Known problems	5
D. Revision history	5
E. Docbook examples	6

1. Introduction

xfront is a frontend to support writing XML documents in a style similar to C/C++/Java, using curly brackets to specify scopes of elements. **xfront** has been developed to get around writing documents in the rather intricate XML syntax.

xfront has been created to aid developers who are in the need of writing documentation. Developers usually tend to prefer the C/C++/Java style syntax to write their programs, and with time and practice, they get very fast writing source code. **xfront** tries to support this style, and wants to attack prejudices against writing XML documents.

Additionally, XML documents are not really adequate to be used in collaboration with revision control systems. The syntax **xfront** expects does not incorporate redundant and distracting XML closing element names, and so it is easier to read and - in case of revision control - easier to compare visually.

xfront tries to support concise writing using *element name aliases* and *element name combiners*. Both very helpful for writing documents using verbose XML DTDs like DocBook [<http://www.docbook.org>].

This text has been created using **xfront** and DocBook [<http://www.docbook.org>].

2. Syntax

The syntax of **xfront** input files is based on *scoped areas* enclosed by ' { ' and ' } ' curly brackets. The ' \ ' backslash character is used to escape characters which would confuse the lexical analysis.

Each *scoped area* is translated into the appropriate XML element.

2.1. Elements

XML element names are simply placed before a scope. **xfront** uses backtracking to query the element name.

Example 1. Simple element

```
section{ title{Introduction} }
```

produces:

```
<section> <title>Introduction</title> </section>
```

Whitespace is ignored at places where **xfront** scans for element names. All other whitespaces (including line feeds and tabs) are emitted to the resulting XML output.

The whitespace *before* an element name is emitted, to output elements without leading whitespace, you may escape the element identifier by the escape character '\ '.

Example 2. Escaping

```
p{Here I em\em{pha}sized the second syllable.}
```

In this example the em element is escaped using the '\ ' character to produce the output:

Here I *emphasized* the second syllable.

To avoid whitespace prefixed in lines for literal layouts, leading whitespace can be removed by placing the scope introducing character '{ ' into a separate line.

If **xfront** detects a '{ ' in a separate line, it automatically removes the whitespace prefixes which are equal to the one seen before the '{ ' from all the lines in the associated element scope.

Example 3. Literal layout

```
programlisting
    {10 PRINT "HELLO"
    20 GOTO 10}
```

2.1.1. Combining elements

Combining elements is supported using the '/ ' character.

Example 4. Combining elements

writing

```
p/lit{literal paragraph}
```

is equivalent to writing

```
p{lit{literal paragraph}}
```

2.2. Attributes

XML attributes are placed *after* the element name and *before* the scope introducing ' {' character. They are specified similar to the attributes in XML.

Example 5. Simple attribute

```
section id="A Tutorial" { title{Tutorial} }
```

If the attribute value does not contain whitespace, the quotes are optional:

Example 6. Attribute quotes

```
section id=Tutorial { title{Tutorial} }
```

Combined elements and attributes are supported:

Example 7. Combined elements and attributes

```
p id="Examples"/ulink url="http://www.examples.org"{Klick me}
```

3. xfront command

```
xfront [options] [filename...]
```

xfront processes all input files provided at the command line. In case there is no filename given, **xfront** expects input characters to arrive at standard input. If filenames are given, **xfront** generates an XML file for each input file. The output files get the same names like their input files with the extension changed to .xml. If no filename is specified, **xfront** writes the XML file to standard output.

Options

-a, --aliases FILENAME	Specifies the alias file to use. Aliases are specified for <i>element names</i> in the form <i>alias=elementname</i> . Empty lines and comments (beginning with the '#' character) are supported.
------------------------	---

4. Quick help

The most important facts to learn using **xfront**.

- For escaping ' { ' ' } ' or '\ ' use '\{ ' '\}' ' '\ \ ' , respectively.
- To suppress whitespace *before* an element name, escape the element name using '\ ' .
- To emit text directly to XML, use ":xml" as element name. Here, **xfront** will not convert '<', '>'

or ' & ' to their appropriate XML entity representation.

Example 8. Emit text to XML

```
:xml{<?xml version="1.0" encoding="UTF-8"?>}
:xml{<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4/2/docbookx.dtd">}
```

- For using umlauts, you may use ISO-8859-1 encoding instead of UTF-8.
- In case of verbose element names, use alias files and use the ' / ' character to combine multiple element names.

Installation

xfront is part of the Libsmart [<http://www.libsmart.com>] library. The **xfront** project is located at the directory `tools/xfront`. To compile and install **xfront** type `make install` in that directory.

Here is a list of the files installed:

xfront installation files

<code>xfront</code>	Binary directory (usually <code>/usr/local/bin</code>)
<code>docbook.aliases</code>	Data directory (usually <code>/usr/local/share</code>)

Required features for version 1.0

xfront is not finished yet, following things are known to be required for version 1.0:

- Support of (verbatim?) text where the special characters '{' and '}' are common. Parts of C/C++/Java source code are an example. Suggestion: '{ { ' and ' } }' scopes.

Example B.1. Verbatim text

```
{{Herein one can write {, } without escaping them using \. }}
```

- Support for XML comments. Suggestion: ': { ' scope.

Example B.2. XML comments

```
:{This is an XML comment}
```

- Required: **vi** and **emacs** syntax highlighting.
- C/C++ style comments? Not sure about this. Currently they are supported but not part of the specification. Probably **xfront** needs to support two styles of comments, one comment type only appearing in the source, the other emitted as XML comments.
- Support for combined aliases (like lip=li/p for example)
- Includes???
- Support entities for shared text.

Known problems

- Parsing of strings enclosed by ' "' may confuse backtracking and users (inside strings ' { ' and ' } ' need not to be escaped), so ' "' content parsing shall be avoided.
- For some reason XML file line numbers differ from the line numbers of their original .xf files.

Revision history

- 2003 12 08 Version 0.3.3:
 - support for escaping ' / ' characters using the backslash '\ ' character.
- 2003 09 03 Version 0.3.2:
 - implemented dynamic whitespace indent detection and removal
- 2003 09 02 Version 0.3.1:
 - fixed problem with detecting escaped parameter names
- 2003 09 02 Version 0.3:
 - fixed problem with string parsing (was longest match instead of shortest).
 - implemented aliases
 - implemented element combiners

- Included some DocBook examples
- 2003 08 29 Version 0.2: Initial release

Docbook examples

For the impatient ones wanting to write DocBook, here are some useful examples, inspired by the Docbook Crash Course [<http://opensource.bureau-cornavin.com/crash-course/>].

Note

You can read the original markup in `xfront.xf`.

Example E.1. Table

Table E.1. Example table

Month	Week	Feet Traveled
10	20	30
11	23	33
12	25	32
13	27	31
Total	0	many

Use `informatable` to create tables without titles

Example E.2. Segmented list

Name: Fred
Address: Fred's street
Name: Bernd
Address: Bernd's street
Name: Felix
Address: Felix's street

Example E.3. Program listing

```
10 PRINT "HELLO WORLD"  
20 GOTO 10
```

Note

For PDF output, program listings must be aligned at column 1 in **xfront** input documents.

Example E.4. Warnings, Tips and Notes

Caution

Hot!

Important

RTFM!

Note

Big brother is watching you!

Tip

Always take the red pill

Warning

Low Battery

Example E.5. Links

- External link [<http://www.replicator.org>].
- Link to a known id in the same document: see Introduction.
- My email address: <armin@replicator.org>

Example E.6. VarList

VarList Term A	Text for variable list entry 1.
VarList Term B	Text for variable list entry 2.
VarList Term C	Text for variable list entry 3.

Index

A

attributes, 3
 and combined elements, 3
 quotes, 3
 simple, 3

E

elements
 combining, 2
 escaping, 2
 simple, 2
examples, 6

W

whitespace, 2
 in XML output, 2

X

xfront
 command, 3
 options, 3