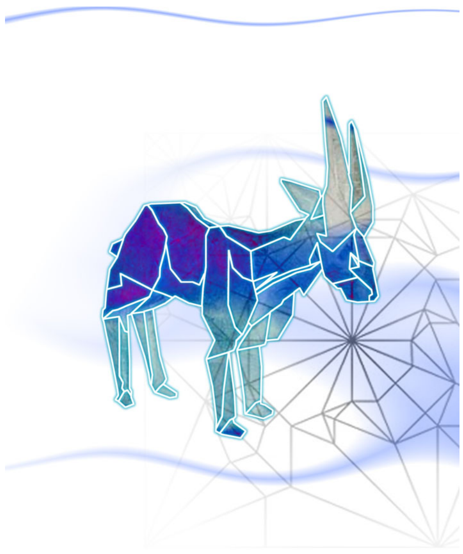
Ibex PDF Creator

.NET Programmers Guide



For Ibex version 3.9.21

Copyright © 2003,2004,2005 Visual Programming Limited

© 2002-2005 Visual Programming Limited. All rights reserved.

NOTICE: All information contained herein is the property of Visual Programming Limited.

No part of this publication (whether in hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

PostScript is a registered trademark of Adobe Systems Incorporated. Adobe, the Adobe logo, Acrobat, the Acrobat logo, Adobe Garamond, Aldus, Distiller, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photoshop, Poetica, and PostScript are trademarks of Adobe Systems Incorporated. Apple, Mac, Macintosh, QuickDraw, and TrueType are trademarks of Apple Computer, Inc., registered in the United States and other countries. ITC Zapf Dingbats is a registered trademark of International Typeface Corporation. Helvetica and Times are registered trademarks of Linotype-Hell AG and/or its subsidiaries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Times New Roman is a trademark of The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be registered in certain other jurisdictions. Unicode is a registered trademark of Unicode, Inc. All other trademarks are the property of their respective owners.

This publication and the information herein are furnished AS IS, are subject to change without notice, and should not be construed as a commitment by Visual Programming Limited. Visual Programming Limited assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third-party rights.

Some parts Copyright (c) 2002-2003, The Sharp Vector Graphics project (http://sourceforge.net/projects/svgdomcsharp) . All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the Sharp Vector Graphics project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Introduction

This manual describes the functionality and use of the Ibex PDF Creator.

What is XSL-FO ?

XSL (eXstensible Stylesheet Language) is a W3C standard which defines an XML vocabulary for use in creating formatted documents from XML. FO stands for Formatting Objects, which is the kind of XML defined in the standard.

The standard defines a set of XML elements and attributes which can be used used to define the layout of a document. The standard defines XML elements for formatting text, tables, lists, images and many other items commonly used in document creation.

How does XSL-FO differ from HTML ?

XSL-FO is oriented towards documents which are formatted in pages. HTML has a very limited concept of pages so does not support required elements such as page numbers or page headers and footers.

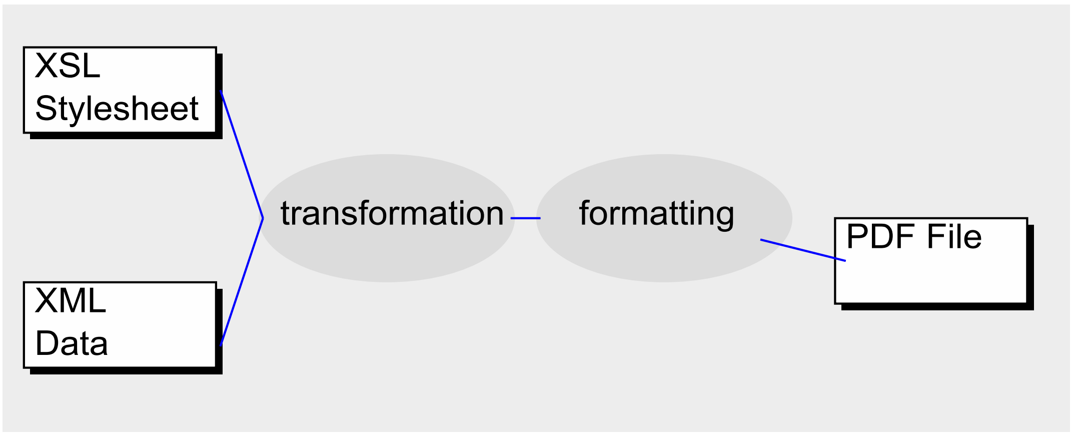
XSL-FO has extensive formatting capabilities for creating page numbers, headers, footers, footnotes and elements typically found in a printed document.

Both XSL-FO and HTML use the common Cascading Style Sheets (CSS) standard for defining attributes such as borders and colors. This standard is widely used and this helps CSS users quickly become proficient with XSL-FO.

How does XSL-FO work ?

XSL-FO is based on the idea of separating presentation from content. The content of a document is presented to the formatter as XML. This XML can be manually edited or retrieved as data from any source. This data XML is transformed using an XSLT stylesheet to get XSL-FO compliant XML which is then converted to PDF format by a formatting program such as Ibex.

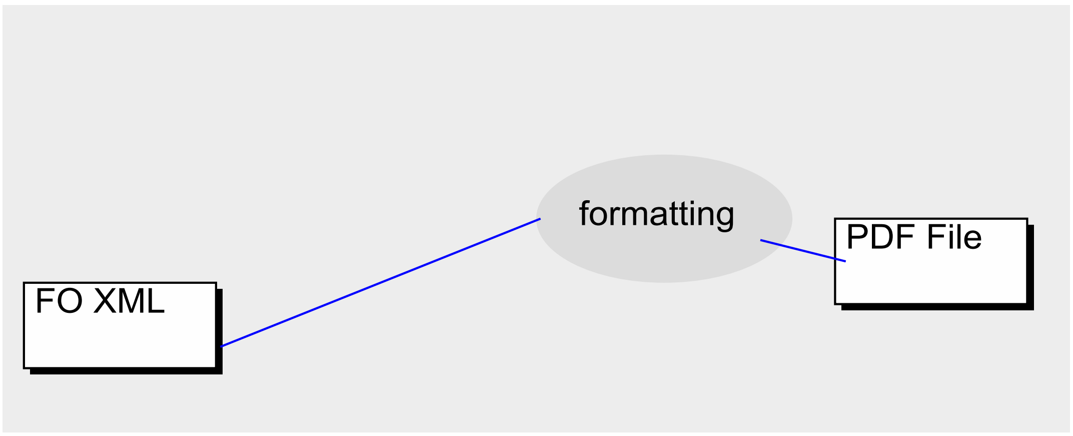
The transformation and formatting process is shown here:



What does Ibex do ?

Ibex performs the transformation and formatting stages show on the diagram above. It takes XML data and an XSLT stylesheet and produces a PDF file. The PDF can be directed to a file on disk or sent as a stream to a web browser.

Ibex will also format a formatting objects file which has been produced from another application such as an XSL editor. The process flow in that case looks like this:



What skills to I need ?

A detailed knowledge of the XSL-FO standard is not required. In fact you should be able to produce the PDF documents you require based solely on the content of this manual. With this in mind this manual focuses on the most frequently used parts of XSL-FO so you can get productive quickly.

About this Manual

Since most developers are familiar with Cascading Style Sheets (CSS) this document concentrates on using those parts of XSL-FO which behave in a way consistent with CSS. This means you can expect the output from Ibex to appear similar to HTML which is produced using the same CSS styles. This particularly applies in the area of tables and border and padding widths where the XSL-FO specification offers a number of formatting options some of which are counter-intuitive and produce different output to that produced by HTML and CSS.

The XSL-FO standard defines an XML namespace "fo". XML elements in this namespace appear in XML files with the namespace prefix, as in "fo:block". To make reading this manual simpler the namespace prefix is normally not shown. It is still required and is shown in all examples [1].

This manual is broken down into chapters convering the following areas: regions, blocks, tables, lists, fonts and error handling.

About Ibex

Ibex is developed entirely in C# and requires the Microsoft .NET Runtime which is available from http://www.microsoft.com. .NET Framework versions 1.0, 1.1 and 2.0 are supported.

This manual was produced with the .NET version of Ibex, release 3.9.21.

[1] If a default namespace is specified in the FO file then the "fo:" prefix is not required. This is done by specifying the default namespace on the root element, like this:

```
    <root xmlns="http://www.w3.org/1999/XSL/Format" ...
```

CHAPTER 1

Installation

The latest version of Ibex can be downloaded from http://www.xmlpdf.com/ibex-downloads-net.html.

The download file is a Windows Installer MSI file can can be installed by double-clicking on it in Explorer.

By default Ibex is installed in the following directory:

c:\program files\visual programming\ibex pdf creator n.n.n

where n.n.n is the version number.

Any number of versions of Ibex can be installed on one machine at the same time.

The installation process installs the .NET assemblies used by Ibex. These are:

ibex10.exe

ibex10.dll

ibex11.exe

ibex11.dll

ibex20.exe

ibex20.dll

ibexshaping11.dll

ibexshaping20.dll

The ibex10.exe and ibex10.dll assemblies are used if you are using version 1.0 of the .NET framework.

The ibex11.exe, ibex11.dll and ibexshaping11.dll assemblies are used if you are using version 1.1 of the .NET framework.

The ibex20.exe, ibex20.dll and ibexshaping20.dll assemblies are used if you are using version 2.0 of the .NET framework.

Each of the DLL assemblies is registered in the Global Assembly Cache (GAC) by the installer.

CHAPTER 2

Getting Started with Ibex

Although primarily intended for use as a part of a larger application Ibex ships with a command line program which can be used to create PDF files from formatting objects (XSL-FO) files. We will use this to demonstrate the basics of PDF creation with Ibex.

The command line programs shipped with Ibex are ibex10.exe (which you use when you have only the 1.0 .NET Framework installed) and ibex11.exe (for use when you have the .NET Framework 1.1 installed).

The command line syntax for both programs is the same. In these examples we use ibex11.exe.

Usage

To create a PDF file from a formatting objects XML file specify the the file names on the command line. For instance to create hello.pdf from hello.fo, you do this:

```
ibex11 hello.fo hello.pdf
```

If the names of the input and output file are the same you can abbreviate this to:

```
ibex11 hello.fo
```

and if the file extension of the input file is "fo" or "xml" you can abbreviate even further to:

```
ibex11 hello
```

Error Logging

Any informational or error messages will be logged to the console. To send any error messages to a file as well use the -logfile option. For example to log errors to the file ibex.log the command becomes:

```
ibex11 -logfile ibex.log hello.fo hello.pdf
```

An Example without XSLT Translation

This example uses the FO file hello.fo which contains the following XML:

```
 <?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <fo:layout-master-set>
    <fo:simple-page-master master-name="page">
      <fo:region-body margin='2.5cm' region-name="body"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="page">
    <fo:flow flow-name="body">
     <fo:block>Hello World</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Each of the elements and attributes used in the file is explained later in the manual, for now we just want to get started with using the Ibex command line program.

using the command

```
ibex11 hello
```

creates the file hello.pdf containing the text 'Hello World'.

Using XSLT Translation

Using Ibex without having Ibex do the XSLT transformation to create the formatting objects XML is useful if you have created the FO using another tool or if you just want to try changing some FO so see what happens without the complexity of editing and testing a stylesheet.

In practice XSLT is almost always part of the PDF creation process because XSL-FO does not have some simple features such as being able to number headings. The designers of XSL-FO presumed that XSLT would be used to this kind of thing and so did not duplicate features already in XSLT.

Ibex gives you the flexiblity of having Ibex do the XSLT translation or having some other tool do it. Internally Ibex uses the XSLT translation classes provided by the .NET Framework .

An Example with XSLT Translation

In this example we will translate some XML with an XSLT stylesheet and produce a PDF from the result of the translation.

We have some weather forecast data in the file weather.xml. This file contains the following XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<forecast>
   <city name='Wellington' temp='20'/>
</forecast>
```

We also have an XSLT stylesheet in the file weather.xsl. This file contains the following:

```
<?xml version='1.0' encoding='utf-8'?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format"
   xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<xsl:strip-space elements='*'/>

<xsl:template match="forecast">

  <fo:root>

    <fo:layout-master-set>
      <fo:simple-page-master master-name="page-layout">
         <fo:region-body margin='2.5cm' region-name="body"/>
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="page-layout">
       <fo:flow flow-name="body">
           <xsl:apply-templates select="city"/>
       </fo:flow>
    </fo:page-sequence>

  </fo:root>

</xsl:template>

<xsl:template match='city'>
   <fo:block>
      <xsl:value-of select="@name"/>

      <xsl:value-of select="@temp"/>
   </fo:block>
</xsl:template>

</xsl:stylesheet>
```

This template outputs the fo:root, fo:layout-master-set and fo:page-sequence elements then for each city record in the data XML outputs an fo:block element using this template:

```
<xsl:template match='city'>
   <fo:block>
      <xsl:value-of select="@name"/>
      &#160;
      <xsl:value-of select="@temp"/>
   </fo:block>
</xsl:template>
```

We can translate and format this example using the command

```
ibex11 -xsl weather.xsl weather.xml weather.pdf
```

The result of this translation is the file weather.pdf

CHAPTER 3

XSL-FO Tutorial

This chapter provides an overview of XSL-FO and provides some suggestions on how to go about the process of creating PDF documents from XML files. We also look at the techniques for using XSLT transformation to create XSL-FO XML.

A very simple XSL-FO file looks like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>

</fo:root>
```

This file is logically in three parts:

(a) the fo:root element:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

which contains the whole content of the file and which identifies the XSL-FO namespace. This element is the same for all XSL-FO files.

(b) the fo:layout-master-set element:

```
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>
```

which defines the shape of pages in the document. Within the fo:layout-master-set we have a fo:simple-page-master element which in turn contains the fo:region-body element.

The fo:simple-page-master defines the layout of one type of page, and is uniquely identified by its master-name attribute. The fo:region-body element defines an area of the page where content will be placed. A page can have more than one region so we give the region a unique name 'body' using the region-name attribute.

A document typically contains many fo:simple-page-master elements, each with a unique master-name. In this simple example we have only one. Each fo:simple-page-master element creates a formatting object known as a 'page master'.

(c) the fo:page-sequence element:

```
<fo:page-sequence master-reference="simple">
   <fo:flow flow-name="body">
      <fo:block>Hello World</fo:block>
   </fo:flow>
</fo:page-sequence>
```

The fo:page-sequence element defines a sequence of pages which will appear in the PDF document. The master-reference attribute is used to tie the content of the fo:page-sequence to a particular page layout, in this case one defined previously using a fo:simple-page-master. When Ibex finds a fo:page-sequence element it looks at the list of known fo:simple-page-master and fo:page-sequence-master elements (we have no fo:page-sequence-master elements in this example) and finds one with a master-name attribute which matches the master-reference attribute on the fo:page-sequence. If Ibex does not find a matching page master this is an error. Looking at the XML again, the two underlined elements must match.

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple">
      <fo:region-body margin="2.5cm" region-name="body"
         background-color='#eeeeee'/>
   </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="simple">
   <fo:flow flow-name="body">
      <fo:block>Hello World</fo:block>
   </fo:flow>
</fo:page-sequence>
```

Within the fo:page-sequence element we have a fo:flow element. This holds the content which will appear on the page. A page can have multiple regions. To associate content with the region it will placed in we use the flow-name attribute on the fo:flow element. In order for the content contained in the fo:flow to appear on the page the flow-name of the fo:flow should match a region-name of one of the regions (in this example the fo:region-body) on the page.

If the flow-name of the fo:flow does not match a region-name of one of the regions on the page the content is not displayed on that page. This is not an error, it is a useful feature and we show how to use it later in this tutorial.

Looking at the XML again, the underlined names must match each other, and the black ones should match if you want the content to appear.

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple">
      <fo:region-body margin="2.5cm" region-name="body"
         background-color='#eeeeee'/>
   </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="simple">
   <fo:flow flow-name="body">
      <fo:block>Hello World</fo:block>
      </fo:flow>
    </fo:page-sequence>
```

Within the fo:flow element we can have one or more "block level" elements. These are elements such as fo:list, fo:block and fo:table which define content to appear on the page. In this example we have a single fo:block element containing the text "Hello World".

This produces an page like the one shown in Figure 1. The region created by the fo:region-body element has a shaded background so you can see how big it is:

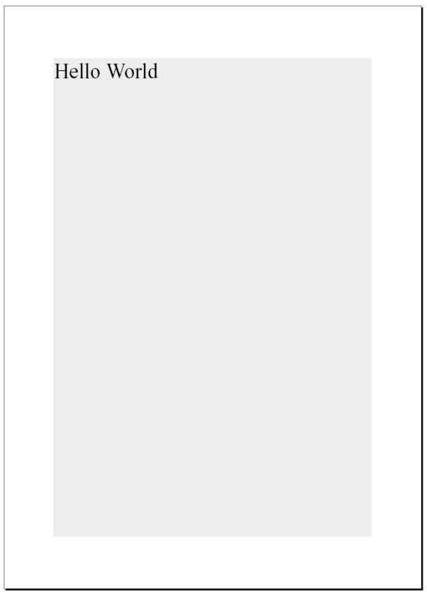


Figure 1: A basic page with a region-body and some text.

3.1. Adding a footer region

All the text contained with the fo:flow element goes into the body region in the center of the page. To add a page footer we need to define a new region on the page and then define some new content to go into that region.

We define a footer region by adding a fo:region-after element into the existing fo:simple-page-master like this:

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple">
     ...
      <fo:region-after extent='1cm' region-name="footer"
        background-color='#dddddd'/>
     ...
     </fo:simple-page-master>
   </fo:layout-master-set>
```

The fo:region-after element defines an area on the page which extends the full width of the page. If we had side regions (fo:region-start and fo:region-end) this might change, but in this example we have no side regions.

The height of the region created by the fo:region-after element is defined by the extent attribute. In this example we have extent='1cm' to the region will be 1cm high and end at the bottom of the page.

Without any content, the footer region is still created and our page now looks like this:

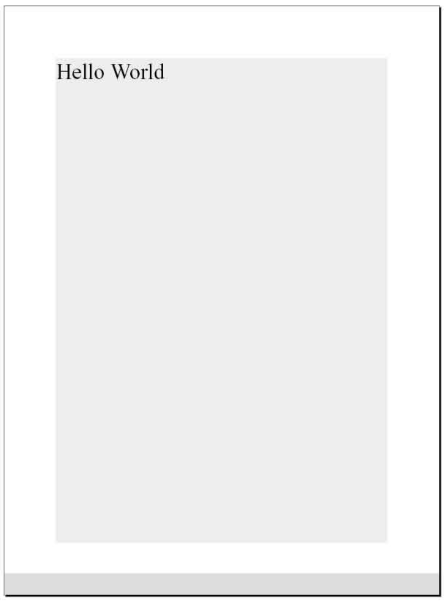


Figure 2: A basic page with a region-body, region-end and some text.

In its current position on the page the footer region will not print on most printers because they do not print right to the edge of the page. We can define a margin around the whole page by setting the margin attribute on the fo:simple-page-master element of the fo:page-sequence like this:

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple"
           margin='2.5cm'>
      <fo:region-body margin="2.5cm" region-name="body"
         background-color='#eeeeee'/>
      <fo:region-after extent='1cm' region-name="footer"
         background-color='#dddddd'/>
   </fo:simple-page-master>
</fo:layout-master-set>
```

This area inside the margins of the fo:simple-page-master is called the page's content area. The area covered by the regions (defined by the fo:region-body and fo:region-end) is measured from the inside of the page's content area, so when we add margins to the fo:simple-page-master we reduce the size of the regions correspondingly.

Our page now looks like this:

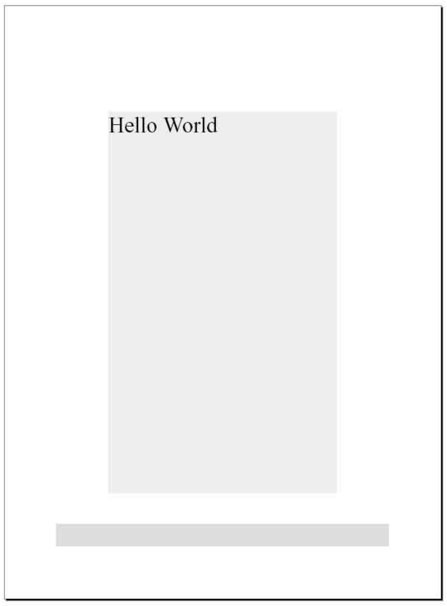


Figure 3: After adding margin to the simple-page-master.

Now that we have some margin space on the sides of the body region, we can remove the side margins from the body by changing the definition from this:

```
<fo:region-body margin="2.5cm" region-name="body"
     background-color='#eeeeee'/>
```

to this:

```
<fo:region-body margin-top="2.5cm" margin-bottom="2.5cm"
   region-name="body"  background-color='#eeeeee'/>
```

giving us a page like this:

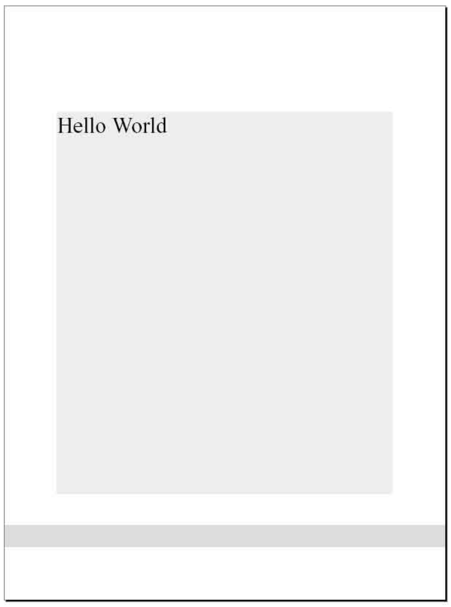


Figure 4: After removing the left and right margins from the region-body.

The last thing we need to do to get a working page layout is to make the footer region narrower by adding side regions. The left side region is created with a fo:region-start element and the right side with a fo:region-end element, as shown below. We can also specify the bottom-margin attribute of the body region to that it ends just where the footer starts, by setting margin-bottom='1cm' on the fo:region-body element.

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple"
          margin='2.5cm'>
      <fo:region-body margin="2.5cm" margin-bottom='1cm'
          region-name="body"
         background-color='#eeeeee'/>
      <fo:region-after extent='1cm' region-name="footer"
         background-color='#dddddd'/>
      <fo:region-start extent='2.5cm'/>
      <fo:region-end extent='2.5cm'/>
   </fo:simple-page-master>
</fo:layout-master-set>
```

By default the side regions take precedence over the top and bottom regions and so the top and bottom regions become narrower. This gives us a page layout like this, to which we can start adding some content.

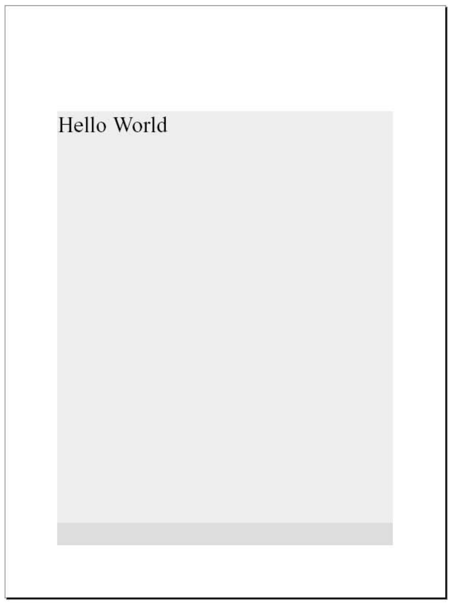


Figure 5: With side regions to reduce the width of the footer.

The XML above also illustrates one of the ways in which XSL-FO handles attributes. We can specify a shorthand attribute such as 'margin', which has the effect of setting the specific values margin-left, margin-right, margin-top and margin-bottom, and then override just the specific value we want (by setting margin-bottom='1cm'). The order in which the attributes are specified has no effect, a more specific setting will always override a more general one. So

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple">
      <fo:region-body margin="2.5cm" margin-bottom='1cm'>
   </fo:simple-page-master>
</fo:layout-master-set>
```

and

```
<fo:layout-master-set>
   <fo:simple-page-master master-name="simple">
      <fo:region-body margin-bottom='1cm' margin="2.5cm">
   </fo:simple-page-master>
</fo:layout-master-set>
```

both produce the same result.

3.2. Adding content to the footer

While content is added to the body of the page using the fo:flow element, content is added to other regions using the fo:static-content element. The 'static' part of the fo:static-content name refers to the fact that the content defined in this element stays within the region specified on this page. If the content exeeds the size of the region it is lost, it does not flow on to other pages like content defined in a fo:flow element.

The content of the fo:static-content is repeated on every page which has a region with a matching flow-name (such as 'header'), and is typically different on every page as the page number changes.

To insert a simple footer with the words 'XSL-FO Example' we add a fo:static-content element like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple"
           margin='2.5cm'>
         <fo:region-body margin="2.5cm" margin-bottom='1cm'
            region-name="body"  background-color='#eeeeee'/>
         <fo:region-after extent='1cm' region-name="footer"
            background-color='#dddddd'/>
         <fo:region-start extent='2.5cm'/>
         <fo:region-end extent='2.5cm'/>
      </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference="simple">
      <fo:static-content flow-name="footer">
          <fo:block text-align='center'>
          XSL-FO Example</fo:block>
      </fo:static-content>
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

Note that the order of the fo:static-content and fo:flow elements is important. All fo:static-content elements must come before the fo:flow element.

This XML produces a page like this:

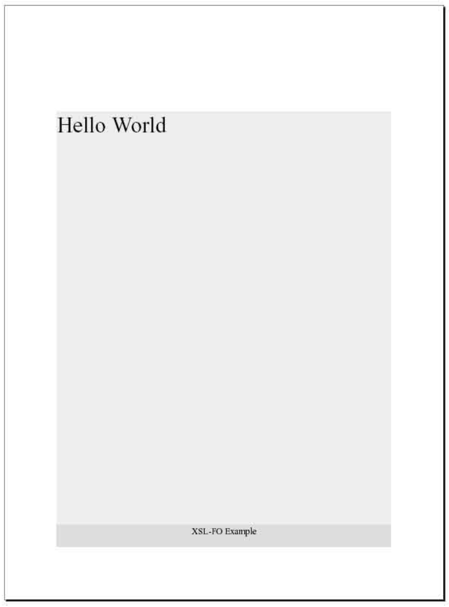


Figure 6: With content in the footer.

Note that the flow-name of the fo:static-content element and the region-name of the fo:region-after element must match for the content to appear.

3.3. Adding the page number to the footer

To insert the current page number into the document use the fo:page-number element inside the fo:static-content element like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple"
           margin='2.5cm'>
         <fo:region-body margin="2.5cm" margin-bottom='1cm'
            region-name="body"  background-color='#eeeeee'/>
         <fo:region-after extent='1cm' region-name="footer"
            background-color='#dddddd'/>
         <fo:region-start extent='2.5cm'/>
         <fo:region-end extent='2.5cm'/>
      </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference="simple">
      <fo:static-content flow-name="footer">
          <fo:block text-align='center'>
          XSL-FO Example, page <fo:page-number/>
          </fo:block>
      </fo:static-content>
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

This XML produces a page like this:

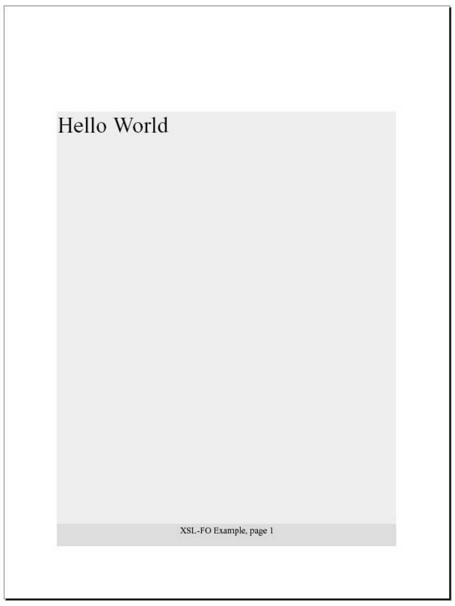


Figure 7: With the page number in the footer.

3.4. Adding the total page count to the footer

Adding the total page count (so we can have 'page 3 of 5') is a two step process, based on the use of the 'id' attribute which uniquely identifies an XML element. We put a block on the last page with the id of 'last-page', and then we use the fo:page-number-citation element to get the number of that page as our total number of pages. Typically the block with the id of 'last-page' is empty so we can move it if required without disturbing the document text.

The XML for the last block in the document looks like this:

```
<fo:block id='last-page'/>
```

And the XML to retrieve the last page number and put it in the footer looks like this:

```
<fo:page-number-citation ref-id='last-page'/>
```

You can see how the id and ref-id values match. This is how XSL-FO associates the two elements and knows which fo:block to retrieve the page number from.

So bringing all these elements together we have this XML:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple"
            margin='2.5cm'>
         <fo:region-body margin="2.5cm" margin-bottom='1cm'
            region-name="body"  background-color='#eeeeee'/>
         <fo:region-after extent='1cm' region-name="footer"
            background-color='#dddddd'/>
         <fo:region-start extent='2.5cm'/>
         <fo:region-end extent='2.5cm'/>
      </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference="simple">
      <fo:static-content flow-name="footer">
          <fo:block text-align='center'>
```

```
                XSL-FO Example, page <fo:page-number/>
                of <fo:page-number-citation ref-id='last-page'/>
                </fo:block>
            </fo:static-content>
            <fo:flow flow-name="body">
               <fo:block>Hello World</fo:block>
               <fo:block id='last-page'/>
            </fo:flow>
         </fo:page-sequence>
      </fo:root>
```

This XML produces a page like this:

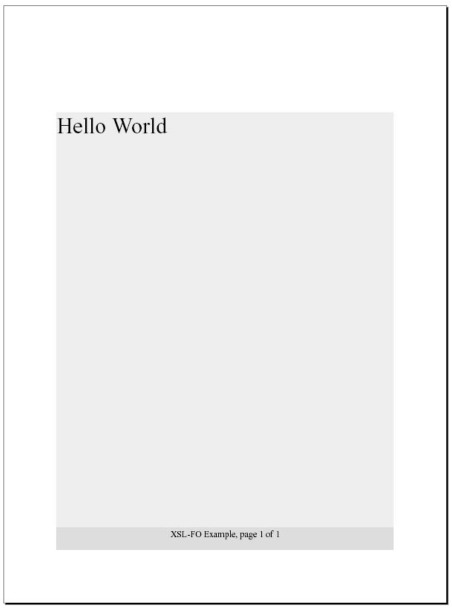


Figure 8: With the page count in the footer.

3.5. Adding text content

Text is added to the body region of the page by using the fo:block element. A fo:block element can contain any amount of text, and has attributes which define how the text will appear. These attributes are described in more detail later in the manual.

A fo:block can contain text like this:

```
            <fo:flow flow-name="body">
               <fo:block>Hello World</fo:block>
            </fo:flow>
```

But it can also contain other fo:block elements which in turn contains text or more nested elements. This XML shows a fo:block which contains another block with a different font, set using the font attribute.

```
            <fo:flow flow-name="body">
               <fo:block>
                    Hello World
                    <fo:block font='16pt arial'>
                       this is a nested block
                    </fo:block>
               </fo:block>
            </fo:flow>
```

There is no limit to the nesting of fo:block elements.

3.6. Using borders and padding

Most XSL-FO elements can have a border around the area they create on the page. If the border around an element is the same on all four sides it can be defined with the border attribute. The space between a border and the content of the block (in this case the text) is controlled using the padding attribute. The XML for a block with border and padding looks like this:

```
<fo:flow flow-name="body">
   <fo:block background-color='#eeeeee'>
      <fo:block>
           Hello World
      </fo:block>
      <fo:block border='1pt solid red' padding='3pt'>
           Hello World
      </fo:block>
   </fo:block>
</fo:flow>
```

This example has two fo:block elements nested inside an outer element, with a background color set on the outer element to emphasis the area created by the outer block. This XML creates this block:

Hello World

Hello World

In keeping with CSS styling most developers are accustomed to the idea that adding border and padding to a block reduces the size of the area for content inside the block by the width of the border plus the width of the padding. This CSS compatible approach is the default one used by Ibex.

To use the alternate approach provided by XSL-FO you can either set xslfo.UserAgent.PreferCSS to false, or explictly define the start-indent attribute for an element. The XML below has the start-indent value set on the fo:block which has a border. This has the effect of setting the indentation of the content (i.e. the text) from the edge of the region, not the edge of the border. After positioning the content, the border and padding is positioned relative to (i.e. *outside*) the content's edge.

The XML to do this is:

```
<fo:flow flow-name="body">
   <fo:block start-indent='2.5cm' background-color='#eeeeee'>
      <fo:block start-indent='2.5cm' >
           Hello World
      </fo:block>
      <fo:block start-indent='2.5cm' border='1pt solid red'
           padding='3pt'>
           Hello World
      </fo:block>
   </fo:block>
</fo:flow>
```

This produces output like this:

Hello World

Hello World

See how text of the block which has a border is aligned with the text in the previous block, and the border is then positioned relative to the content. Because this approach is counter-intuitive to most developers with CSS experience it will not be used in this manual and we will stick to CSS compatible usage of XSL-FO.

3.7. Using margins

In keeping with CSS usage space is inserted to the left and right of a block using the margin-left and margin-right attributes. Also in keeping with CSS usage margins are cumulative so the margin on a block sets its identation relative to any containing block or region. If a block has a left margin of 3cm and this block contains a second block also with a left margin of 3cm, the second block will be indented by 6cm. The XML to do this is:

```
<fo:flow flow-name="body">
   <fo:block margin-left='3cm' background-color='#eeeeee'>
    This is some text in the outer block and we can
    see it is indented by 3cm from the
    left edge of the region.
      <fo:block margin-left='3cm' background-color='#dddddd'>
           This is some text in the inner block and
           we can see it is indented by 6cm from
           the left edge of the region.
      </fo:block>
   </fo:block>
</fo:flow>
```

This XML produces the following:

This is some text in the outer block and we can see it is indented by 3cm from the left edge of the region.

This is some text in the inner block and we can see it is indented by 6cm from the left edge of the region.

3.8. Creating lists

A list is content divided into two columns, called the *label* and the *body*. A list is created with the fo:list-block element. A fo:list-block contains one or more fo:list-item elements, each of which contains exactly one fo:list-item-label element and one fo:list-item-body element.

An example of a simple list is this:

□ this is item one
□ this is item two

This list was created with the following XML:

```
<fo:list-block
     margin-left='3cm' margin-right='3cm' padding='3pt'
     border='.1pt solid blue'
     provisional-distance-between-starts='0.5cm'
     provisional-label-separation='0.1cm'>
     <fo:list-item>
         <fo:list-item-label end-indent='label-end()'>
             <fo:block font='10pt arial'>&#x25A1;</fo:block>
         </fo:list-item-label>
         <fo:list-item-body start-indent='body-start()'>
             <fo:block>this is item one</fo:block>
         </fo:list-item-body>
     </fo:list-item>

     <fo:list-item>
         <fo:list-item-label end-indent='label-end()'>
            <fo:block font='10pt arial'>&#x25A1;</fo:block>
         </fo:list-item-label>
         <fo:list-item-body start-indent='body-start()'>
            <fo:block>this is item two</fo:block>
         </fo:list-item-body>
     </fo:list-item>
</fo:list-block>
```

A list constrains the two columns to the widths specified using the attributes of the fo:list-block elements. The provisional-distance-between-starts attribute specifies the distance between the start of the label column and the start of the body column. The provisional-label-separation attribute sets how much of the label column should be left empty to provide a blank space between the columns.

If we expand the above example and add some more content to the first body we can see that is its constrained to the column. Changing the first fo:list-item-body to this:

```
<fo:list-item-body start-indent='body-start()'>
  <fo:block>
     If your Network Administrator has enabled it,
     Microsoft Windows can examine your network and
     automatically discover network connection settings.
  </fo:block>
</fo:list-item-body>
```

Then the list will appear like this:

□ If your Network Administrator has enabled it, Microsoft Windows can examine your network and automatically discover network connection settings.
□ this is item two

3.9. Creating tables

A table is created using the fo:table element. Within the fo:table element there can be one fo:table-header element, any number of fo:table-body elements and one fo:table-footer element. Each of the fo:table-header, fo:table-body and fo:table-footer elements contains one or more fo:table-row elements, each containing one or more fo:table-cell elements which in turn contain block-level elements such as fo:block, fo:table and fo:list-block.

Since a fo:table-cell can contain any block-level element you can easily create tables which contain text, nested tables or lists. Table headers defined with the fo:table-header element can be optionally repeated at each page break.

A simple table is defined with the following XML:

```
<fo:table>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
         <fo:block>row 1 column 1</fo:block></fo:table-cell>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
         <fo:block>row 1 column 2</fo:block></fo:table-cell>
  </fo:table-row>
  <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
         <fo:block>row 2 column 1</fo:block></fo:table-cell>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
         <fo:block>row 2 column 2</fo:block></fo:table-cell>
  </fo:table-row>
</fo:table-body>
```

This XML produces the following table:

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

The padding and border attributes are not inherited from containing elements so should be defined on the fo:table-cell elements.

3.10. Setting table column widths

The width of a table column is set using the fo:table-column element. A fo:table element contains zero or more fo:table-column elements each of which defines properties such as width and background-color for a column in the table. For instance to make the first column 30% of the table width we would add fo:table-column elements like this:

```
<fo:table>
  <fo:table-column column-width='30%' column-number='1'/>
  <fo:table-column column-width='70%' column-number='2'/>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
         <fo:block>row 1 column 1</fo:block>
      </fo:table-cell>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
            <fo:block>row 1 column 2</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 1</fo:block>
       </fo:table-cell>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 2</fo:block>
       </fo:table-cell>
    </fo:table-row>
</fo:table-body>
```

This XML produces the following table:

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

3.11. Using XSLT

For other than simple documents XSLT transformation is required to create the XSL-FO XML. XSLT is useful in the following circumstances:

When numbering is required:

XSL-FO has no facility for automatically numbering content such as chapters in a document. The numbers must be contained within the XML provided to the formatting engine. XSLT transformation is a simple and effective way of providing automatic numbering of items.

When the document content is already XML:

In many applications the majority of the content which will appear in the PDF file is provided as XML. XSLT is a fast an simple way of creating the XSL-FO XML from the data XML.

When the document is large:

When a document has a lot of content the process of setting similar attributes on paragraphs of text is tedious. The <xsl:attribute-sets> facility of XSLT makes it simple to apply and change styles which apply over the whole document.

CHAPTER 4

Usage

This chapter describes how to call the Ibex API and use the accompanying command line program.

4.1. Ibex command line program

Although primarily intended to be used as a part of a larger application, Ibex ships with a command line program which can be used to create PDF files from XSL-FO XML files.

The command line programs shipped with Ibex are ibex10.exe (which uses .NET Framework 1.0), ibex11.exe (which uses the .NET Framework 1.1) and ibex20.exe (which uses .NET Framework 2.0).

The command line syntax for all programs is the same, in these examples we use ibex11.exe.

To create a PDF file from a XML file specify the the file names on the command line. For instance to create hello.pdf from hello.fo, you do this:

```
ibex11 hello.fo hello.pdf
```

XSLT translation

The command line program will accept XML data and an XSLT stylesheet as inputs. The XML will be translated to XSL-FO by the stylesheet and the results then formatted to PDF. The command line syntax is:

```
ibex11 -xsl book.xsl book.xml hello.pdf
```

XSLT parameters can be passed to the stylesheet by adding them as name-value pairs to the command line. For instance if we wanted to define the paramter called "age" to the value "30" we would use a command like this:

```
ibex11 -xsl book.xsl book.xml hello.pdf "age=30"
```

The use of the double quotes around the name-value pair is necessary on some operating systems to force them to come through as a single parameter to the Ibex program.

Logging from the command line

Any informational or error messages will be logged to the console. To send error messages to file as well, use the -logfile option. For example to log errors to the file ibex.log, you would do this:

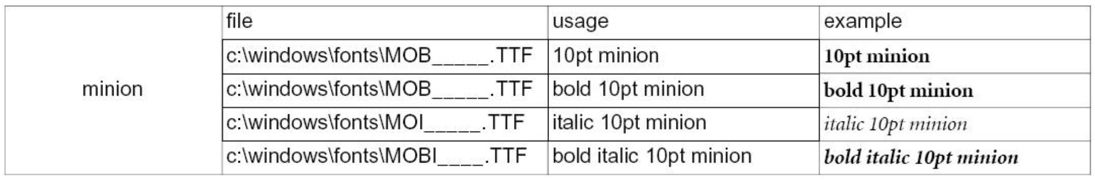
```
ibex11 -logfile ibex.log hello.fo hello.pdf
```

Listing available fonts

You can also list the fonts which are available (based on what fonts are installed on your system) by using the -fonts option like this:

```
ibex11 -fonts
```

The list of fonts is emitted as a XSL-FO file to the standard output. This can be redirected to a file and then used as input to Ibex to create a PDF file containing a table which looks like this:



	file	usage	example
minion	c:\windows\fonts\MOB_____.TTF	10pt minion	10pt minion
	c:\windows\fonts\MOB_____.TTF	bold 10pt minion	**bold 10pt minion**
	c:\windows\fonts\MOI_____.TTF	italic 10pt minion	*italic 10pt minion*
	c:\windows\fonts\MOBI____.TTF	bold italic 10pt minion	***bold italic 10pt minion***

The list of fonts can be limited to fonts which contain a specified string by passing the string on the command line. For instance if we wanted to see what versions of 'arial' are installed, we can use the command:

```
ibex11 -fonts arial
```

This produces output like that shown below. The output includes an example font attribute showing the correct syntax for using this font.

```
font 'arial', c:\windows\fonts\arial.ttf
   usage font="12pt arial"
font 'arial baltic' is an alias for 'arial'
font 'arial black', c:\windows\fonts\ariblk.ttf
   usage font="12pt arial black"
font 'arial black italic', c:\windows\fonts\arbli___.ttf
   usage font="italic 12pt arial black"
font 'arial bold', c:\windows\fonts\arialbd.ttf
   usage font="bold 12pt arial"
font 'arial bold italic', c:\windows\fonts\arialbi.ttf
   usage font="italic bold 12pt arial"
font 'arial ce' is an alias for 'arial'
font 'arial cyr' is an alias for 'arial'
font 'arial greek' is an alias for 'arial'
font 'arial italic', c:\windows\fonts\ariali.ttf
   usage font="italic 12pt arial"
font 'arial narrow', c:\windows\fonts\arialn.ttf
   usage font="12pt arial narrow"
font 'arial narrow bold', c:\windows\fonts\arialnb.ttf
   usage font="bold 12pt arial narrow"
font 'arial narrow bold italic', c:\windows\fonts\arialnbi.ttf
   usage font="italic bold 12pt arial narrow"
font 'arial narrow italic', c:\windows\fonts\arialni.ttf
   usage font="italic 12pt arial narrow"
font 'arial tur' is an alias for 'arial'
font 'arial unicode ms', c:\windows\fonts\arialuni.ttf
   usage font="12pt arial unicode ms"
```

4.2. Ibex API

A PDF document is generated by making a new xslfo.FODocument object and then calling the generate() method on that object. The generate() method takes either (a) two file names (fo and pdf) or (b) two streams, when not using XSLT, or (c) three streams, when using XSLT. A PDF document is generated by making a new xslfo.FODocument object and then calling the generate() method on that object. The generate() method takes either (a) two file names (fo and pdf) or (b) two streams, when not using XSLT, or (c) three streams, when using XSLT.

For example to convert the file 'manual.fo' to 'manual.pdf' the code is like this (in C#):

```
using System;
using xmlpdf.logging;
using xslfo;

FODocument doc = new FODocument();
doc.generate( "manual.fo", "manual.pdf" );
```

or like this (in VB.NET)

```
import System
import xmlpdf.logging
import xslfo

Dim FODocument as New FODocument()
doc.generate( "manual.fo", "manual.pdf" )
```

4.3. Generating to File

```
public void generate( string foFileName, string pdfFileName )
```

This will read the XML contained in foFileName and create the PDF file named as pdfFileName.

4.4. Generating Using Streams

```
public void generate(
   Stream foStream,
   Stream pdfstream,
   bool closeStream )

public void generate( Stream foStream,
      Stream pdfstream )
```

This will read the XML from the System.IO.Stream called foStream and create the PDF file into the stream 'pdfstream'.

If closeStream is true the stream will be closed after the PDF file is generated, if false it will not. By default the stream is closed. Not closing the stream is useful if you are generating to a MemoryStream object.

Typically if you were writing to a FileStream you will want to close the file, like this:

```
FODocument doc = new FODocument();
doc.generate(
      new FileStream(
          "manual.fo", FileMode.Open,
             FileAccess.Read ) );
      new FileStream(
          "manual.pdf", FileMode.Create,
             FileAccess.Write ) );
```

However if you are writing to a MemoryStream you would not close it because you need it open to get the PDF data out of it.

4.5. XSLT Transform and Generate

```
public void generate( Stream xml, Stream xsl, Stream pdf )

public void generate( Stream xml, Stream xsl, Stream pdf,
       bool closeStream )
```

These methods take XML, an XSLT stylesheet, and a stream to write the resulting PDF file to.

Ibex uses the .NET XSLT processor to transform the XML using the specified stylesheet and passes the resulting XSL-FO to the PDF creation routines. The transformation is done using streams in memory, no files are saved to disk.

4.6. XSLT Transform and Generate with parameters

```
public void generate( Stream xml, Stream xsl, Stream pdf,
       bool closeStream, Hashtable params )
```

These methods are similar to the ones in the previous section but take an additional hashtable which (if not null) should contain name-value pairs which are then passed as arguments to the XSLT translation process.

CHAPTER 5

Using Ibex with ASP.NET

This chapter shows how to use Ibex with ASP.NET, including how to create a PDF file and stream it back to a client browser without needing to save it to disk.

Ibex creates a PDF file into a Stream object (from the System.IO namespace). In this example we use a MemoryStream object which derives from Stream and is basically an expandable array of bytes held in memory. Creating the PDF file in a MemoryStream means the file is not saved to disk and the whole process occurs in RAM.

We create the file into a MemoryStream because many versions of Internet Explorer only display a PDF file correctly if they know how long the file is. We need to create the PDF file into a MemoryStream to determine its length and set this length in an HTTP header. This need to know the length of the PDF file prevents us from streaming directly to the Response object.

5.1. The ASP Page

This example uses a page called pdfasp.aspx, which has code behind it in pdfasp.aspx.cs. To use these pages you will need to create a new ASP.NET C# project, add a reference to the Ibex11.dll wherever you have installed it, and then add the pdfgen.aspx page. You will need to remove the '_' characters from the names of the pdfasp.aspx and pdfasp.aspx.cs files.

The pdfasp.aspx file contains the following:

```
<%@ Page language="c#" validateRequest="false" Codebehind="pdfasp.aspx.cs"
  AutoEventWireup="false" Inherits="WebApplication1.pdfasp" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>pdfgen</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
        </form>
    </body>
</HTML>
```

5.2. The ASP Code behind page

The pdfasp.aspx.cs file contains the following (plus some error handling which is not shown for clarity):

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Text;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Xml.Xsl;
using System.Security.Policy;

using xmlpdf;
using xmlpdf.logging;
using xslfo;

namespace WebApplication1 {

  public class pdfasp : System.Web.UI.Page {

  private void Page_Load(object sender, System.EventArgs e) {

     Logger.getLogger().setLevel( Level.FINEST ).clearHandlers();

     try {
       Stream stream = new FileStream(
              Server.MapPath( @"logs\log.txt"), FileMode.Append,
                  FileAccess.Write ) ;

        Logger.getLogger().addHandler( new StreamHandler( stream ) ) ;
      }
      catch( Exception ) {
      }

      FODocument doc = new FODocument();

      string dataFilePath = Server.MapPath("") + "\\hello.fo";

      // if you have a license
      xmlpdf.licensing.Generator.LicenseFileLocation
          = Server.MapPath("") + "\\xmlpdf.lic";

      // stream for output in memory before sending to browser
      MemoryStream pdfStream = new MemoryStream();

      FileStream dataStream = new FileStream( dataFilePath,
             FileMode.Open, FileAccess.Read );

      Logger.getLogger().info( "data file path is " + dataFilePath );

      using( dataStream ) {
          doc.generate( dataStream, pdfStream, false );
      }

      Response.Clear();

      Response.ContentType = "application/pdf";
      Response.AddHeader( "content-length",
            System.Convert.ToString( pdfStream.Length ) );

      Response.BinaryWrite( pdfStream.ToArray() );
      Response.End();
   }
```

```
         override protected void OnInit(EventArgs e)   {
            InitializeComponent();
            base.OnInit(e);
         }

         private void InitializeComponent() {
            this.Load += new System.EventHandler(this.Page_Load);
         }
       }
      }
```

Key things to note in the process of streaming the PDF to the client include:

Logging

We set up logging so if an error occurs we get the error logged to file:

```
       Logger.getLogger().setLevel( Level.FINEST )
           .clearHandlers();

       try {
           Stream stream = new FileStream( Server.MapPath( @"logs\log.txt"),
                  FileMode.Append, FileAccess.Write ) ;

           Logger.getLogger().addHandler( new StreamHandler( stream ) ) ;
       }
           catch( Exception ) {
       }
```

Setting the MIME type

We set the correct MIME type for to cause the browser to invoke the Acrobat plugin to display the PDF:

```
      Response.ContentType = "application/pdf";
```

Setting the content length

We set the content length to IE will correctly handle the content:

```
      Response.AddHeader( "content-length",
          System.Convert.ToString( pdfStream.Length ) );
```

Not closing the MemoryStream

We call the version of generate() which takes a boolean to indicate we do not want the output stream to be closed. If it were closed we would not be able to get the PDF data from the stream back to copy back to the browser. Typically of we were writing the PDF to a file on disk we would close the stream but in this case we need to stream to stay open.

```
      using( dataStream ) {
           doc.generate( dataStream, pdfStream, false );
      }
```

5.3. Using XSLT

If we want to translate our XML data using a stylesheet the ASP code looks like this:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Text;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Xml.Xsl;
using System.Security.Policy;

using xmlpdf;
using xmlpdf.logging;
using xslfo;

namespace WebApplication1 {

  public class pdfasp : System.Web.UI.Page {

  private void Page_Load(object sender, System.EventArgs e) {

     Logger.getLogger().setLevel( Level.FINEST ).clearHandlers();

     try {
       Stream stream = new FileStream(
              Server.MapPath( @"logs\log.txt"), FileMode.Append,
                  FileAccess.Write ) ;

        Logger.getLogger().addHandler( new StreamHandler( stream ) ) ;
      }
      catch( Exception ) {
      }

      FODocument doc = new FODocument();

      string dataFilePath = Server.MapPath("") + "\\data.xml";
      string templateFilePath = Server.MapPath("") + "\\template.xsl";


      // if you have a license
      xmlpdf.licensing.Generator.LicenseFileLocation
          = Server.MapPath("") + "\\xmlpdf.lic";

      // stream for output in memory before sending to browser
      MemoryStream pdfStream = new MemoryStream();

      FileStream dataStream = new FileStream( dataFilePath,
             FileMode.Open, FileAccess.Read );

      FileStream xslStream = new FileStream( templateFilePath,
             FileMode.Open, FileAccess.Read );


      Logger.getLogger().info( "data file path is " + dataFilePath );

      using( dataStream ) {
          using( xslStream ) {
           doc.generate( dataStream, xslStream, pdfStream, false );
      }
      }

      Response.Clear();
```

```
            Response.ContentType = "application/pdf";
            Response.AddHeader( "content-length",
                  System.Convert.ToString( pdfStream.Length ) );

            Response.BinaryWrite( pdfStream.ToArray() );
            Response.End();
         }

         override protected void OnInit(EventArgs e)   {
            InitializeComponent();
            base.OnInit(e);
         }

         private void InitializeComponent() {
            this.Load += new System.EventHandler(this.Page_Load);
         }
       }
      }
```

The changes made were:

(a) we declared a variable for the XSLT template like this:

```
      string templateFilePath =
         Server.MapPath("") + "\\template.xsl";
```

(b) we opened the XSLT template using a Stream:

```
      FileStream xslStream = new FileStream( templateFilePath,
        FileMode.Open, FileAccess.Read );
```

(c) we passed the XSL stream to the generate() method:

```
      using( dataStream ) {
        using( xslStream ) {
           doc.generate( dataStream, xslStream, pdfStream, false );
        }
      }
```

CHAPTER 6

Error Handling

This chapter describes error handling using the Ibex API.

Ibex associates an error handler with the library as a whole. Generally this error handler will log a message and not throw an exception.

The Ibex Logger object is a singleton which is retrieved using a call to the xmlpdf.logging.Logger.getLogger() method. Typically you would import the xmlpdf.logging namespace and then access the logger like this:

```
using xmlpdf.logging;

void sumfunc() {
    Logger.getLogger().clearHandlers();
}
```

The default error handler writes messages to the console. Messages are displayed in various circumstances including:

- when an invalid attribute is found
- when a reference is made to a font or image file which cannot be found
- when an formatting error occurs, such as defining the widths of columns in table which exceed the available width.

To change the level of information logged you can set the level on the logging object to one of the values defined in the xmlpdf.logging.Level object. Possible levels of logging which can be set are:

SEVERE WARNING INFO CONFIG FINE FINER FINEST

For example to set the logger to log only messages which are WARNING or worse do this:

```
using System;
using xslfo;
using xmlpdf.logging;

public class Create {

  public static void Main( string[] args ) {

    PDFDocument doc = new PDFDocument();
```

```
          Logger.getLogger().setLevel( Level.WARNING );
```

6.1. Logging to File

To log messages to a file, create an xmlpdf.logging.FileHandler object and then tell the logger to log to this object. This example logs to the file 'log.txt', but any valid file name can be used.

```
using System;
using xslfo;
using xmlpdf.logging;

public class Create {

  public static void Main( string[] args ) {


    Logger.getLogger()
      .setLevel( Level.SEVERE )
      .clearHandlers()
      .addHandler(
        new FileHandler("log.txt") );
```

The FileHandler object synchronises access to the log file.

If you omit the clearHandlers() call shown in the above example, log records will be written to the default console handler and also to the file handler. You will see error messages on the console and they will also be written to the file.

6.2. Logging to a Stream

Ibex can log messages to a stream created by the caller. The stream is any object which implements the System.IO.Stream interface.

To log messages to a stream, create an xmlpdf.logging.StreamHandler object and then tell the logger to log to this object. This example logs to a MemoryStream, but any valid stream can be used.

```
using System;
using System.IO;
using xslfo;
using xmlpdf.logging;

public class Create {

  public static void Main( string[] args ) {

    Logger.getLogger().clearHandlers();
    MemoryStream stream = new MemoryStream();
    StreamHandler h = new StreamHandler( stream );
    Logger.getLogger().addHandler( h )
```

If you omit the clearHandlers() call shown in the above example log records will be written to the default console handler and to the stream handler as well.

Logging to System.Diagnostics.Trace

Ibex can log messages to the System.Diagnostics.Trace object. This means if you are debugging using the Visual Studio IDE you will see the messages in the output window of the IDE.

To log messages to the trace object, create a xmlpdf.logging.TraceHandler object and then tell the logger to log to this object.

```
using System;
using xslfo;
using xslfo.loggging;

public class Create {

  static void Main( string[] args ) {


    Logger.getLogger().clearHandlers();
    Logger.getLogger().addHandler(
        new TraceHandler() );
```

When running in the Visual Studio IDE messages will be logged to the Output window, and when running executables messages will be written to the output debug stream where they can be read using a utility like Debug View.

6.3. Logging to Multiple Destinations

Errors can be logged to any number of handlers. The following example logs to a file called "xslfo.log" and to a memory stream and to the console. Logging to the console is done by not removing the default handler with a call to clearHandlers().

```
using System;
using System.IO;

using xslfo;
using xmlpdf.logging;

public class Create {

  public static void Main( string[] args ) {

    MemoryStream stream = new MemoryStream();

    Logger.getLogger()
      .addHandler( new StreamHandler( stream ) )
      .addHandler( new FileHandler("xslfo.log") );
 }
}
```

CHAPTER 7

Basic Page Layout

The first thing in a simple XSL-FO file is the fo:root element which contains the whole XML tree defining the document and declares the XML namespaces which is used as follows:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

The first XSL-FO element within the fo:root element is the fo:layout-master-set element. This contains one or more fo:simple-page-master elements which define the shape of a page, including width and height. The fo:simple-page-master element contains region elements such as fo:region-body which define an area on the page which can be filled with text or image content.

The following is an example of a fo:layout-master-set:

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="front-page">
    <fo:region-body margin-right="2.5cm"
           margin-left="4cm"
           margin-bottom="4cm"
           margin-top="4cm" region-name="body"
     background-color='#eeeeee'/>
    <fo:region-after extent="3cm" region-name="footer"
       background-color='#dddddd'/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

This shows a fo:layout-master-set which contains a single fo:simple-page-master with a master-name of 'front-page'.

This fo:simple-page-master defines a page which has two regions on which content can be printed. A page defined with this layout appears in the examples at the end of this chapter, on page 205. For the purposes of this example the regions have background-colors defined to show them clearly. More complex layouts showing all five regions appear in the examples on page 205.

Having defined a page layout which has a name, (defined by its master-name attribute) we then use the fo:page-sequence element to define the content of the document. The fo:page-sequence element has a master-name attribute which should match the master-name defined for a fo:simple-page-master (or a fo:page-sequence-master, more of which later).

A fo:page-sequence for printing 'Hello World' would look like this:

```
<fo:page-sequence master-reference="front-page">
  <fo:flow flow-name="body">
    <fo:block>Hello World</fo:block>
  </fo:flow>
</fo:page-sequence>
```

A key thing to note is that the content of the fo:page-sequence is contained in a fo:flow element. For content of the flow to appear on the PDF page *the flow-name attribute of the fo:flow element must match the region-name of a region on the page master specified by the master-reference on the fo:page-sequence*. If the flow-name does not match a region-name, none of the content of this fo:flow will appear in the output document.

It is important to understand this feature. It means that a fo:page-sequence can contain multiple fo:static-content elements each containing a fo:flow element with a different flow-name. Only fo:flow elements whose flow-name attribute matches a region-name defined in the current page sequence will appear. This is how we produce different formats for odd and even pages.

This example shows in matching colors the attributes which should match for content to appear:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="front-page">
      <fo:region-body margin-right="2.5cm"
           margin-left="4cm"
           margin-bottom="4cm"
           margin-top="4cm" region-name="body"/>
      <fo:region-after extent="3cm" region-name="footer"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="front-page">
    <fo:flow flow-name="body">
      <fo:block>Hello World</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

For more complex page layout requirements Ibex supports the following XSL-FO elements:

- fo:page-sequence-master
- fo:repeatable-page-master-alternatives
- fo:conditional-page-master-reference

CHAPTER 8

Fonts

Text is inserted into the document using a fo:block element, like this:

```
<fo:block>hello world</fo:block>
```

The font used for a block is specified either by a single font attribute such as font='12pt arial' or by one or more lower level attributes such as font-size and font-weight.

The font used for a block is inherited from the containing element. If no font is specified on any containing element the default font is used. The default font for Ibex is '12pt times'.

The font attribute can be used to specify a font which can then be changed using more specific attributes. This example sets the font for the block to '12pt arial' and the changes the word 'fox' to 'bold 12pt arial' using just the font-weight attribute.

```
<fo:block font="12pt arial">
  the quick brown <fo:inline font-weight="bold">fox</fo:inline>
  jumped over the lazy dog.
</fo:block>
```

The example produces this output:

the quick brown **fox** jumped over the lazy dog.

8.1. Using the font attribute

The quickest way to get the font you require is to use the `font` attribute, like this:

```
<fo:block font='bold 12pt garamond'>hello world</fo:block>
```

Using the font attribute is simpler than specifying all the individual attributes such as font-weight and font-size, but does need some care. When using the font attribute the order of the words is important. The font style (normal, italic) and the font weight (bold, normal) must come before the font size. This is the CSS standard. The font name must come after the font size. If the font name contains spaces, it must be enclosed in quotes, like this:

```
<fo:block font='bold 12pt "times new roman"'>
```

```
  hello world
</fo:block>
```

The full syntax of the font attribute is:

```
[ [ <font-style> || <font-variant> || <font-weight> ]?
<font-size> [ / <lineheight>]?
<font-family> ]
```

8.2. Using the font-style attribute

The font style can be 'normal' or 'italic'. Other font values such as the font-family are inherited from the current font, as shown in this example:

```
<fo:block font-family='arial'>
  hello <fo:inline font-style='italic'>world</fo:inline>
</fo:block>
```

which produces this:

hello *world*

8.3. Using the font-weight attribute

The font weight can be 'normal' or 'bold'. Other font values such as the font-family are inherited from the current font, as shown in this example:

```
<fo:block font-family='arial'>
  hello <fo:inline font-weight='bold'>world</fo:inline>
</fo:block>
```

which produces this:

hello **world**

8.4. Using the font-size attribute

The font size specifies the size of the font and can be any of the following values:

an absolute size:

attribute value	actual size
xx-small	7.0pt
x-small	8.3pt
small	10.0pt
medium	12.0pt
large	14.4pt

x-large	17.4pt
xx-large	20.7pt

The values defined for these sizes are exposed as static float members on the class xslfo.UserAgent and can be changed by the user. For instance to change the x-large value to 18pt you could to this:

```
xslfo.UserAgent.X_Large = 18.0f;
```

a relative size

smaller	current size / 1.2
larger	current size * 1.2

The values defined for these sizes are exposed as static strings on the class xslfo.UserAgent and can be changed by the user. They are expressed as multiples of em, the current fonts size. For instance to change the meaning of the 'larger' value to be 2 times the current value you could to this:

```
xslfo.UserAgent.Larger = "2.0em";
```

a length	a specific font size in points such as '12pt' or '30pt'
a percentage	a value such as '130%' to make the font 1.3 times the size of the surrounding object.

Other font values such as the font-family are inherited from the current font, as shown in this example:

```
<fo:block font-family='arial'>
  hello <fo:inline font-size='20pt'>world</fo:inline>
</fo:block>
```

which produces this:

hello world

The fixed size font values are shown with this XML:

```
<fo:block font-family='arial' font-size='12pt'>
  12 pt
  <fo:inline font-size='xx-small'>xx-small</fo:inline>
  <fo:inline font-size='x-small'>x-small</fo:inline>
  <fo:inline font-size='small'>small</fo:inline>
  <fo:inline font-size='medium'>medium</fo:inline>
  <fo:inline font-size='large'>large</fo:inline>
  <fo:inline font-size='x-large'>x-large</fo:inline>
  <fo:inline font-size='xx-large'>xx-large</fo:inline>
</fo:block>
```

Which produces this output:

12 pt xx-small x-small small medium large x-large xx-large

This XML shows the larger and smaller values:

```
<fo:block font-family='arial' font-size='12pt'>
12 pt
<fo:inline font-size='larger'>larger</fo:inline>
12 pt
<fo:inline font-size='smaller'>smaller</fo:inline>
</fo:block>
```

And produces this:

12 pt larger 12 pt smaller

8.5. Font Configuration

Ibex reads the registry to see what fonts are available. Specifically the entries under "HKLM\software\microsoft\windows nt\currentversion\fonts" list available fonts, and those under "HKLM\software\microsoft\windows nt\currentversion\fontsubstitutes" list translations from font names to existing fonts. Any of the font names listed in these two places can be used.

Individual components of the font can be specified using attributes described below.

The XSL-FO standard calls for the formatting engine to check each character to which the font-family attribute applies to see if that character is contained in the font. Currently this is not implemented for performance reasons.

For example, the following block specifies fonts which do not exist ('kermit' and 'dog') followed by one which does ('monospace'), so the text will be displayed in the font which does exist.

```
<fo:block font-family='kermit,dog,monospace'>
  hello world
</fo:block>
```

Like this:

```
hello world
```

8.6. Using the font-family attribute

This attribute specifies one or more font family names. These names can be specific font names such as 'times roman' or 'garamond', or generic names such as 'monospace'. Ibex will use the first name in the list which matches a font on your system. Font names are separated by a comma.

CHAPTER 9

Text Formatting

Text is created in the output document using the fo:block element.

The simplest possible fo:block looks like this:

```
<fo:block>hello world</fo:block>
```

This creates a paragraph in the output document which has the default font and the default alignment (which is left).

The sections below detail additional attributes which control the formatting of text.

9.1. Using the text-align attribute

The text-align attribute controls alignment of text in the horizontal direction. Valid values are:

left	text is aligned against the left edge of the block.
right	text is aligned against the right edge of the block.
center	text is centered in the middle of the block.
justify	text is aligned against both the left and right edges of the block. Space is inserted between words and letters to achieve this effect.
start	text is aligned against the start edge, which for an unrotated block is the left edge.
end	text is aligned against the end edge, which for an unrotated block is the right edge.
inside	if the page binding edge is on the start edge, the alignment will be start. If the binding is the end edge, the alignment will be end. If neither, the start alignment will be used.

outside	if the page binding edge is on the start edge, the alignment will be end. If the binding is the end edge, the alignment will be start. If neither, the end alignment will be used.

For text-align values of 'inside' and 'outside' the page number is used to determine the binding edge, which is assumed to be the left hand edge of odd-numbered pages and the right hand edge of even-numbered pages.

This paragraph has no text-align attribute, so by default is aligned to the left, so that the words form a smooth line against the left margin and a ragged edge on the right.

This paragraph has text-align='right' and so is aligned to the right, so that the words form a smooth line against the right margin and have a ragged edge on the left.

This paragraph has text-align='justify', so that the words form a smooth line against both the left and right margins, except for the last line which is aligned independently using the text-align-last attribute.

This paragraph has text-align='center', so that the words are centered in the middle of the block.

9.2. Using the text-align-last attribute

The text-align-last attribute controls the alignment of the last line of a paragraph. Values include:

relative	if text-align is 'justify', align the last line against the start edge (normally the left edge), otherwise use the setting if the text-align attribute.
left	align at left edge.
start	align at start edge, which for an unrotated block of text is the left edge.
right	align at right edge.
end	align at end edge, which for an unrotated block of text is the right edge.
justify	justify last line across whole width of page.

This paragraph is justified with text-align='justify' and text-align-last='left' so the last line is aligned to the left.

This paragraph is justified with text-align='justify' and text-align-last='right' so the last line is aligned to the right.

This paragraph is justified with text-align='justify' and text-align-last='justify' so the last line is fully justified. Not recommended.

9.3. Using margin attributes

The margins of a fo:block are specified using the margin-left and margin-right attributes.

The margin-xxx properties indent the edge of the paragraph by the specified amount from the edge of the containing area.

A block with a 2.5cm left margin is specified like this:

```
<fo:block margin-left='2.5cm'>hello world</fo:block>
```

and looks like this (with a background to show the limits of the block):

hello world

If we nest another block inside this one, the margins are cumulative:

```
<fo:block margin-left='2.5cm'>
  hello
  <fo:block margin-left='2.5cm'>
    world
  </fo:block>
</fo:block>
```

producing output like this:

hello

world

Putting background colors on the blocks shows this more clearly.

```
<fo:block margin-left='2.5cm' background-color='#777777'>
  hello
  <fo:block margin-left='2.5cm'  background-color='#999999'>
    world
  </fo:block>
</fo:block>
```

producing output like this:

hello

world

The approach to indentation defined in the XSL-FO standard is that two nested blocks which do not specify a margin have the same left edge, so this code produces areas with the same left hand edge:

```
<fo:block padding='1cm' background-color='#777777'>
  hello
  <fo:block padding='1cm' background-color='#999999'>
    world
  </fo:block>
</fo:block>
```

like this:

hello

world

In XSL-FO terms, both areas have the same start-indent and hence the same content rectangle, and the outer areas padding extends outside the content rectangle.

This is counter-intuitive to most developers used to the CSS model.

You can invoke the CSS nested areas model in two ways:

- specify a margin-left value, even '0pt'
- set xslfo.UserAgent.preferCSS to "true".

Either specifying a margin-left attribute, and so invoking the CSS nested model, like this:

```
<fo:block padding='1cm' margin-left='0pt'
 background-color='#777777'>
  hello
  <fo:block padding='1cm' margin-left='0pt'
 background-color='#999999'>
    world
  </fo:block>
</fo:block>
```

or setting the static preferCSS value like this:

```
 xslfo.UserAgent.preferCSS = true;
```

will produce this:

hello
world

9.4. Spacing between letters

The spacing between letters is controlled by the letter-spacing attribute. Any value specified using this attribute is *added* to the default spacing specified by the font.

For instance to increase the spacing between letters we can do this:

```
<fo:block letter-spacing="0.2em">WELLINGTON NEW ZEALAND</fo:block>
```

which produces this result:

W E L L I N G T O N   N E W   Z E A L A N D

It is possible to make letters closer than normal using a negative value for letter-spacing like this:

```
<fo:block letter-spacing="-0.1em">WELLINGTON NEW ZEALAND</fo:block>
```

which produces this result:

WELLINGTON NEW ZEALAND

9.5. Spacing before and after words

Spacing before and after text is specified using the space-start and space-end attributes on the inline element.

The space-start attribute specifies space to appear before text, space-end specifies space to appear after the text.
For example to specify a gap between two words we can specify space before the first word like this:

```
<fo:block>
   hello <fo:inline space-start="1cm">world</fo:inline>
</fo:block>
```

This produces a 1cm gap between the words like this:

hello   world

Space between words is collapsed (i.e. merged) by default. If a word has space-end="1cm" and the following word has space-start="0.5cm", the gap between the two words will be the larger of the two spaces, not the sum.
The following example shows this:

```
<fo:block>
   <fo:inline space-end="1cm">hello</fo:inline>
   <fo:inline space-start="0.5cm">world</fo:inline>
</fo:block>
```

This produces a 1cm gap between the words like this:

hello   world

Merging margins between words

If a word has space-end="1cm" and the following word has space-start="0.5cm", the gap between the two words will be the larger of the two spaces, not the sum. This can be changed by either (a) setting the precedence value of the spaces to a numeric value, in which case the highest value will be chosen, or (b) setting one or both precedence values to "force", in which case the one(s) with "force" will be added.
For instance if we have a 1cm gap with precedence="2" and a 0.5cm gap with precedence="3", the smaller gap will be chosen as it has the highest precedence:

```
<fo:block>
   <fo:inline space-end="1cm" space-end.precedence="2">hello</fo:inline>
   <fo:inline space-start="0.5cm"  space-start.precedence="3">world</fo:inline>
</fo:block>
```

This produces a 0.5cm gap between the words like this:

hello   world
To force margins not to be merged, set precedence to "force" like this:

```
<fo:block>
   <fo:inline space-end="1cm" space-end.precedence="force">hello</fo:inline>
   <fo:inline space-start="0.5cm"
space-start.precedence="force">world</fo:inline>
</fo:block>
```

This produces a 1.5cm gap between the words like this:

hello   world

Space at the start of a line

Space specified with the space-start attribute is normally discarded at the start of the line. To force it to be retained use the space-start.conditionality attribute.

This XML shows two blocks, creating two lines. The first block will have no space at the start of the word. The second block has space-start.conditionality="retain" so the space specified by the space-start="1cm" will be retained.

```
<fo:block background-color="#eeeeee">
   <fo:inline space-start="1cm">discard</fo:inline>
</fo:block>
<fo:block background-color="#eeeeee">
   <fo:inline space-start="1cm"
     space-start.conditionality="retain">retain</fo:inline>
</fo:block>
```

This produces the following output:

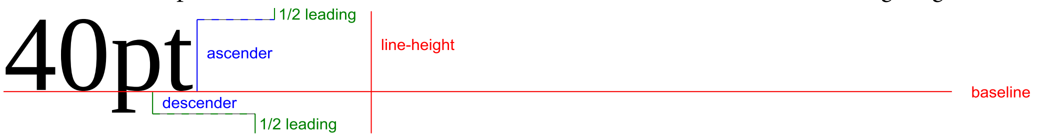
```
discard
      retain
```

9.6. Vertical Alignment

The vertical alignment of blocks of text within a containing fo:flow or fo:block is controlled by the display-align attribute.

The vertical alignment of words on a line is controlled by the vertical-align attribute.

Text on a line is positioned relative to the baseline, which is shown on the following diagram:



By default text sits on the baseline. In the terms of the XSL-FO specification, this is the *alphabetic* baseline.

The height of the font above the baseline is the *ascender*. The height of the font below the baseline is the *descender*. Adding the ascender and descender values for the font (not for individual characters) gives the font size. The *leading* is the space above and below the characters, and is the difference between the line-height and the font-size.

The XSL-FO specification refers to the ascender value as the *text-altitude* and the descender as the *text-depth*. Together these two values add up to the *allocation rectangle* height. In these terms:

leading = (line-height - text-altitude - text-depth)
so

1/2 leading = (line-height - text-altitude - text-depth) / 2

By default the line height is 1.2em. The em unit is proportional to the size of the current font, so as the font size increases so does the line height. This can be changed by setting the xslfo.UserAgent.LineHeightNormal value, for instance to make the line height larger and so space text out more vertically you could do this:

```
UserAgent.LineHeightNormal = 1.4em;
```

Line stacking strategies

XSL-FO uses the line-stacking-strategy attribute to determine how lines are stacked vertically on a page. The default value of this attribute is "max-height". The information which follows assumes that this default value is used. The other values for line-stacking-strategy, namely "font-height" and "line-height" will produce different results.

The leading value is calculated from the line-height and font-size specified for the fo:block element which contains the text. It is constant for the whole block and is not affected by other values specified on fo:inline or other elements contained within the block.

The allocation rectangle for the line is calculated using "largest" characters found on the line, i.e. the sum of the max(ascender) and max(descender) values.

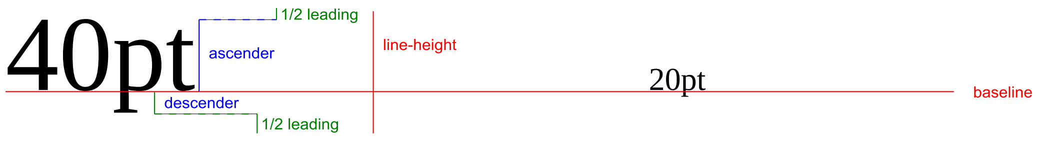
The effect of subscript and superscript text on line spacing

When calculating the largest characters on this line, we really mean those whose ascender and descender values are greatest (i.e. futherest from the baseline). When making this calculation, the value of the line-height-shift-adjustment attribute is considered. If text is a subscript or superscript and so has a baseline-shift value which changes its position vertically, this will also change its effective ascender and descender values and possibly make the allocation rectangle larger. If line-height-shift-adjustment="consider-shifts" (the default value) then the baseline-shift amount is taken into account when working out the greatest ascender and descender. If line-height-shift-adjustment="disregard-shifts" then the effect of the baseline-shift is ignored. Setting line-height-shift-adjustment="disregard-shifts" makes lines stay the same distance apart regardless of subscript and superscript elements.
The effect line-height-shift-adjustment is shown here; the first two lines are in a block which has line-height-shift-adjustment="consider-shifts" and so are further apart than the second two which are in a block which has line-height-shift-adjustment="disregard-shifts":

Specifies a string on which content of cells in a table column will align (see the section, in the CSS2 Recommendation^{2}).

Specifies a string on which content of cells in a table column will align (see the section, in the CSS2 Recommendation^{2}).

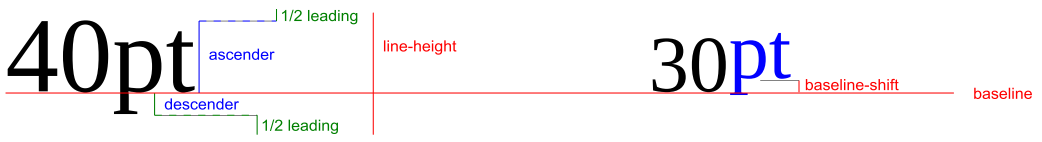
The baseline
The baseline is below the top of the text block a distance equal to 1/2 leading + max(ascender), which places the baseline in the same place for all text elements. This means that normally text rests on the same baseline regardless of the font size, as shown here:



Subscript and superscript

Subscript and superscript text is created by using the baseline-shift attribute on an fo:inline element.

The effect of the baseline shift is shown here:



For instance to move a word above the current baseline by 5 points you would do this:

```
<fo:block
  hello
      <fo:inline color='red' baseline-shift='5pt'>
      super
      </fo:inline>
  </fo:block>
</fo:block>
```

This produces the following content:

hello super

To move a word above the current baseline by the default amount for the current font, you would do this:

```
<fo:block
  hello
      <fo:inline color='red' baseline-shift='super'>
      super
      </fo:inline>
  </fo:block>
</fo:block>
```

This produces the following content:

hello super

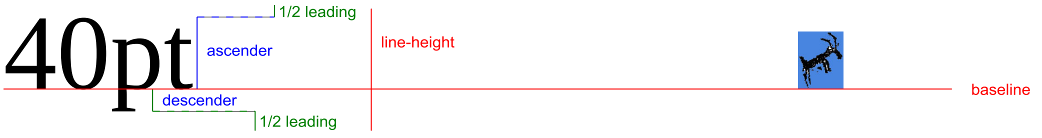
In addition to lengths like '5pt' and '7pt' the baseline-shift attribute can have the following values:

sub	move text below the current baseline by an amount specified in the TrueType font file. If a TrueType font is not used the amount is 0.5em. This default can be changed by setting the UserAgent.BaselineShiftSub value programatically.
super	move text above the current baseline by an amount specified in the TrueType font file. If a TrueType font is not used the amount is 0.5 em. This default can be changed by setting the UserAgent.BaselineShiftSuper value programatically.
baseline	the text is positioned on the current baseline

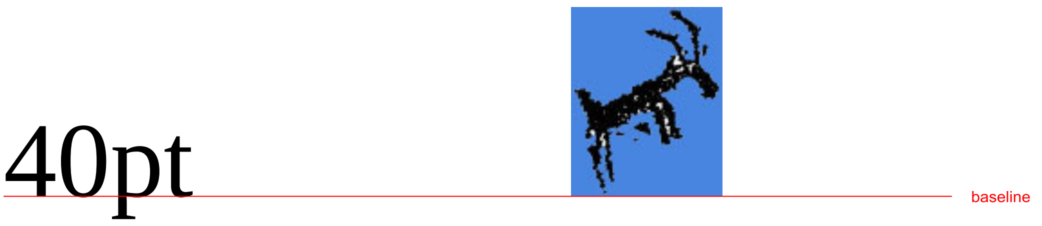
9.7. Aligning images

An inline element such as fo:external-graphic is treated similarly to a text element. The height of the image is used as the ascender value. The descender value is zero.

This means that by default an image will be positioned on the baseline like this:

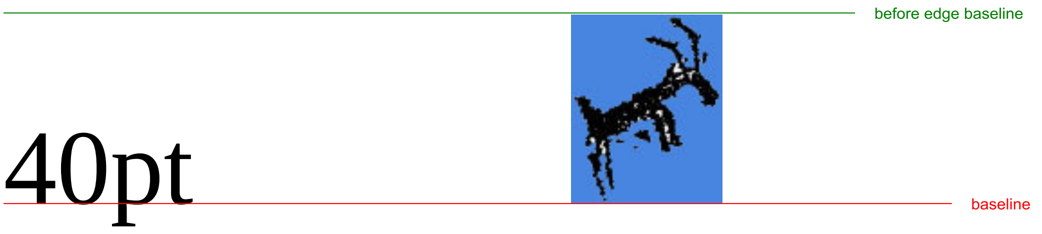


A large image will contribute a large ascender value to the baseline placement calculation, but will still sit on that baseline like this:



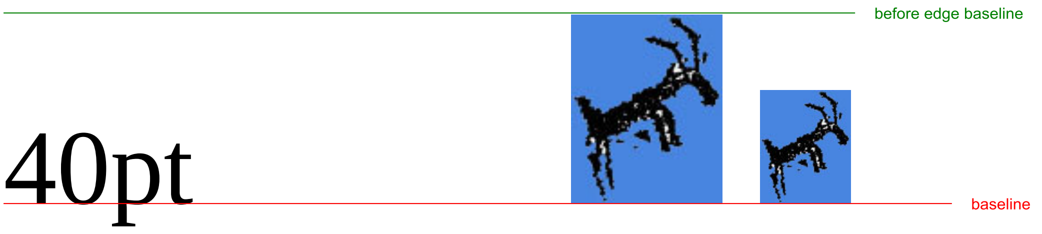
The before-edge baseline

By default an element has a alignment-baseline value of "baseline" and so sits on the baseline shown in the above diagrams. For a given line, the largest thing on that line which has alignment-baseline="baseline" establishes the position of the *before edge baseline*. This is shown here:

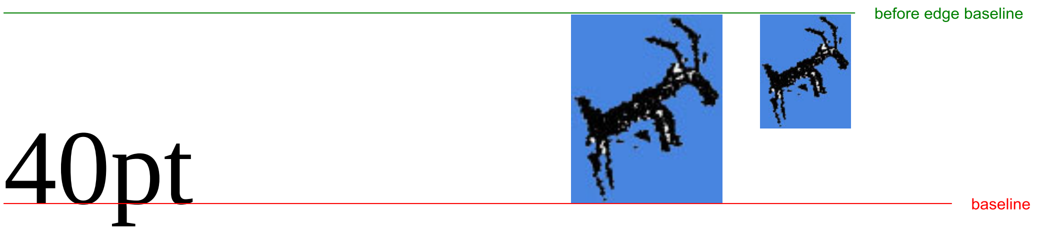


To align another object with the before edge baseline, either set vertical-align="top" or alignment-baseline="before-edge".

The following shows a second smaller image with default alignment, which positions it on the baseline:



By specifying vertical-align="top" on the external-graphic for the second image, we can align this image to the before edge baseline and get this effect:



If all the elements on the line have vertical-align="top", then the *before edge baseline* cannot be calculated, so the *text before edge baseline* is used. This is the top of the ascender for the font specified for the block which contains the elements.

before
start
end
after

CHAPTER 10

Text Flow

10.1. The fo:float element

The fo:float element can be used to position an image or other elements to the side of the page and cause text to flow around that image.

In Ibex the implementation of fo:float is limited to situations where the float attribute is "before" or "start".

The paragraph which follows uses a fo:float element to make the image appear to the left and the text to flow around the image and below it.

This text should appear to the right of the image until we pass the bottom of the image and then appear below the image as well.

We have lots of text here just to show that it will be formatted in the correct way and eventually there will be enough text to go past the image and appear below it on the page. Then we will have some XML which shows how to acheive this effect. We have lots of text here just to show that it will be formatted in the correct way and eventually there will be enough text to go past the image and appear below it on the page. Then we will have some XML which shows how to acheive this effect. We have lots of text here just to show that it will be formatted in the correct way and eventually there will be enough text to go past the image and appear below it on the page. Then we will have some XML which shows how to acheive this effect.

This effect is achieved by having an outer fo:block which contains a fo:float element and some fo:block elements. The fo:float element in turn contains a block-container element which has a inline-progression-dimension attribute defining the width of the float area. Any elements inside the block-container will be in the float area.

The XML for creating the above example is:

```
<fo:block>
  <fo:float float="start">
       <fo:block-container inline-progression-dimension="3cm" padding="5mm">
            <fo:block padding="2mm" space-before.conditionality="retain"
border="1pt solid white" width="2cm">
                 <fo:block padding-left="2mm">
                        <fo:external-graphic src="url(ibexorange.jpg)"
content-width='50%'/>
                 </fo:block>
             </fo:block>
        </fo:block-container>
```

```
    </fo:float>
    <fo:block padding-top="2mm" padding-bottom='2mm' space-before="9pt"
  font="10pt 'minion regular'">
    This text should appear to the right of the image until we pass the bottom of
   the image
    and then appear below the image as well.
    </fo:block>
    <fo:block font="10pt 'minion regular'">
    We have lots of text here just to show that it will be formatted in the
  correct way and eventually
    there will be enough text to go past the image and appear below it on the
  page.  Then we will have some
    XML which shows how to acheive this effect.
    We have lots of text here just to show that it will be formatted in the
  correct way and eventually
    there will be enough text to go past the image and appear below it on the
  page.  Then we will have some
    XML which shows how to acheive this effect.
    We have lots of text here just to show that it will be formatted in the
  correct way and eventually
    there will be enough text to go past the image and appear below it on the
  page.  Then we will have some
    XML which shows how to acheive this effect.
    </fo:block>
  </fo:block>
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin vitae neque ac pede hendrerit tempor. Nunc in magna et diam aliquam ultricies. In sed ipsum a arcu auctor porttitor. Quisque facilisis, augue sed condimentum molestie, quam velit mattis metus, quis consectetuer magna lectus laoreet lectus. Sed suscipit, nulla vitae lobortis auctor, sapien augue dictum est, ac commodo lacus dui nec odio.

Pellentesque dolor a a a a mauris, dapibus id, blandit ac, luctus mollis, lacus. Donec quis leo. Vivamus adipiscing pretium tortor. In tristique dui id sem. Praesent pulvinar tristique mauris. Cras vitae dui. Nulla rutrum nisl in sem. Donec sodales arcu nec leo. Suspendisse et est. Duis ut est. Etiam lobortis commodo neque.

CHAPTER 11

Space Handling

XSL-FO defines various attributes for managing space handling in XML.

By default linefeeds and whitespace preceeding and following newlines are removed during formatting.

This example shows some input XML which has linefeeds and spaces before and after text.

```
<fo:block margin='2cm'>        To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
</fo:block>
```

The resulting output has neither linefeeds nor spaces around the text. This is the default situation.

To be, or not to be: that is the question: Whether 'tis nobler in
the mind to suffer The slings and arrows of outrageous
fortune, Or to take arms against a sea of troubles,

11.1. Using the linefeed-treatment attribute

We can retain the linefeeds by setting the linefeed-treatment attribute to 'preserve', so the example is now:

```
<fo:block linefeed-treatment='preserve''>
        To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
</fo:block>
```

And the resulting output is:

To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,

11.2. Using white-space-treatment and white-space-collapse

Suppose we want to put some code in our document, we might have XML like this:

```
<fo:block linefeed-treatment='preserve'>
private void swap_byte( ref byte x, ref byte y ) {
  byte t = x;
  x = y;
  y = t;
}
</fo:block>
```

with linefeed-treatment='preserve' we get this output. We have preserved the linefeeds but all formatting spaces have gone.

```
private void swap_byte( ref byte x, ref byte y ) {
byte t = x;
x = y;
y = t;
}
```

The white-space-collapse attribute controls whether the formatter compresses adjacent white space characters into a single character. Setting white-space-treatment='preserve' makes the formatter retain white space which appears adjacent to linefeeds.

If we set white-space-treatment to 'preserve', and white-space-collapse to 'false' we will retain the white spaces around the linefeeds, like this:

```
<fo:block
   linefeed-treatment='preserve'
   white-space-treatment='preserve'
   white-space-collapse='false'>
private void swap_byte( ref byte x, ref byte y ) {
  byte t = x;
  x = y;
  y = t;
}
</fo:block>
```

producing the properly formatted code example:

```
private void swap_byte( ref byte x, ref byte y ) {
  byte t = x;
  x = y;
  y = t;
}
```

11.3. Non-breaking spaces

Unicode defines the character U+00A0 called NO-BREAK SPACE. This can be used to insert a space between words without allowing a line break to occur between the words. Ibex treats two words seperated by a U+00A0 as a single word.

For example the following paragraph has no no-break spaces and so breaks between the words 'break' and 'anywhere'.

This is a list of words with no no-break spaces so can break
anywhere between two words

If we insert a no-break space between the words 'break' and 'anywhere' Ibex will treat them as a single large word which will be moved to the next line, like this:

This is a list of words with no no-break spaces so can
break anywhere between two words

The no-break space can be inserted into XML using the &#160; entity like this:

```
<fo:block>
 This is a list of words with no no-break
 spaces so can break&#160;anywhere
 between two words
</fo:block>
```

Alternatively the Unicode &#x00A0; entity can be used like this:

```
<fo:block>
 This is a list of words with no no-break spaces
 so can break&#x00A0;anywhere between two words
</fo:block>
```

CHAPTER 12

Colors

XSL-FO defines various attributes for managing color.

By default text is displayed with the foreground color (ie. the text) being black and the background color being white.

12.1. Text color

The color of text is defined using the color attribute, so to make some text blue we do this:

```
<fo:block color='blue'>
        To be, or not to be: that is the question:
</fo:block>
```

The resulting text will be blue like this

12.2. Background color

The background color of any element is defined using the background-color attribute, so to make a block have a gray background we would do this:

```
<fo:block background-color='gray'>
        To be, or not to be: that is the question:
</fo:block>
```

The resulting text will have a gray background like this

The color and background color attributes can be combined like this:

```
<fo:block background-color='black' color='white'>
        To be, or not to be: that is the question:
</fo:block>
```

The resulting text will be white on black

12.3. Available colors

The value used for the color and background-color attributes can be a predefined color such as 'red', an RGB color defined using a hex value such as '#eeffdd' or a CMYK color.

12.4. Predefined colors

XSL-FO uses the list of colors defined for HTML 4.0, which contains these values:

aqua	ibex
black	ibex
blue	ibex
fuchsia	ibex
gray	ibex
green	ibex
lime	ibex
maroon	ibex
navy	ibex
olive	ibex
purple	ibex
red	ibex
silver	ibex
teal	ibex
white	
yellow	ibex

12.5. Hex RGB colors

A color can be defined as a string of six digits preceeded by a '#' character. The first two digits define the red component of the color, in a range from 0 to 255. The second two digits define the green component and the last two define the blue component.

12.6. CMYK colors

A CMYK color can be defined using the rgb-icc function. This takes eight parameters. The first three define the red, green and blue components of a fallback RGB color, the fourth defines the color profile name, and the last four define the four parts of the CMYK color. The color profile must have been declared in the fo:declarations formatting object using an fo:color-profile element.

An example of this function is:

```
<fo:block color='rgb-icc( 0, 0, 0, cmyk, 0.7,0.3,0.3,0.4 )'>
   in cmyk .5,.5,.5,0
</fo:block>
```

In this example, the three components of the fallback RGB color are zero and the color profile name is 'cmyk'. Ibex requires that the color profile name be 'cmyk' when creating a CMYK color.

A complete document using the CMYK color space looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 <fo:layout-master-set>
  <fo:simple-page-master master-name="page">
   <fo:region-body margin="1in"
       region-name="body"/>
  </fo:simple-page-master>
 </fo:layout-master-set>

 <fo:declarations>
     <fo:color-profile src='src'
            color-profile-name='cmyk'/>
 </fo:declarations>

 <fo:page-sequence master-reference="page">
  <fo:flow flow-name="body">
   <fo:block color='rgb-icc( 0, 0, 0, cmyk, 0.7,0.3,0.3,0.4 )'>
   in cmyk .5,.5,.5,0
   </fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

This shows how to use the fo:declarations and fo:color-profile elements to define a color profile.

CHAPTER 13

Lists

A fo:list-block in XSL-FO is an area of content divided into two columns, the first being called the *label* and the second the *body*.

A simple fo:list-block looks like this:

```
<fo:list-block provisional-distance-between-starts=".5cm"
    provisional-label-separation="0.1cm">
   <fo:list-item>
      <fo:list-item-label end-indent="label-end()">
         <fo:block font='8pt arial'>&#x25CF;</fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">
           <fo:block>
               item one
           </fo:block>
       </fo:list-item-body>
  </fo:list-item>
   <fo:list-item>
      <fo:list-item-label end-indent="label-end()">
         <fo:block font='8pt arial'>&#x25CF;</fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">
           <fo:block>
               item two
           </fo:block>
       </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```

This example produces this list:

- item one
- item two

Looking at one thing at a time:

- the fo:list-block is a block-level element which contains the whole list.
- the provisional-distance-between-starts attribute on the fo:list-block defines the distance between the start of the first column (the label) and the start of the second column (the body).
- the provisional-label-separation attribute on the fo:list-block defines the size of the gap between the two columns. The gap between the two columns is created by reducing the size of the content in the first column. For example if provisional-distance-between-starts is 5cm and the

provisional-label-separation is 1cm, then the start edge of the two columns will be 5cm apart, and the first column will be 4cm (5cm - 1cm) wide.

- each item in the list is contained in a fo:list-item element.
- each fo:list-item must contain both a fo:list-item-label and a fo:list-item-body. The fo:list-item-label must come first.
- the fo:list-item-label should have the end-indent attribute set to 'label-end()'. This is a special function which returns a value derived from provisional-distance-between-starts and provisional-label-separation.
- the fo:list-item-body should have the start-indent attribute set to 'body-start()'. This is a special function which returns a value derived from provisional-distance-between-starts and provisional-label-separation.
- both the fo:list-item-label and fo:list-item-body contain one or more block-level elements, so a fo:list-item-label or fo:list-item-body can contain other block-level element including such as fo:block, fo:table and fo:list-block.

The widths of the label and body columns should be managed using the provisional-distance-between-starts and provisional-label-separation attributes.

13.1. Bulleted lists

The example above also shows how to insert a Unicode character into the XML, using the syntax &#x25CF;.

Here are some common bullet types for lists:

Unicode	Result
&#x2022;	•
&#x2023;	‣
&#x25CF;	●
&#x25CB;	○
&#x25A0;	■
&#x25A1;	□
&#x25C6;	◆
&#x25C7;	◇

The list of bullet types looks like a table but is actually created with a fo:list-block by setting the border attributes on the fo:list-block and the border-bottom attribute on the first fo:list-item.

Note that what actually gets displayed in the document depends on whether the font you are using contains the specified character. If the font does not contain the specified character you will see a warning message like this:

```
    warning:380 No glyph index found for character code 2023 in font ArialMT
```

The example bullets above are displayed using the mighty Arial Unicode MS font which contains just about every character and is 23 MB in size. Ibex will load this file and output a PDF file which contains only the subset of the font which you actually used.

13.2. Nested lists

Lists can be nested. This example shows a lists nested 3 deep. The nested lists have their border set to make this more obvious.

&#x2022; •

&#x2023;

1 I'm a nested list

2 I'm a nested list

3

1 I'm a doubly nested list

2 I'm a doubly nested list

3 I'm a doubly nested list

&#x25CF; ●

Chapter 14

Columns

XSL-FO allows us to define a page which has multiple columns, just like this one.

This can only be done for whole pages, not for partial pages. However if we are in a region which has multiple columns we can treat it as a single-column region and place output across the whole width of the multi-column page by setting span='all' on block-level elements which appear immediately below the fo:flow element.

Columns are defined by setting the `column-count` attribute of a region element (such as fo:region-start, fo:region-end, fo:region-before etc.) to more than 1, and optionally setting the `column-gap` attribute to define a gap between the columns.

The page master for this page is:

```
<fo:simple-page-master  master-name="chapter-2col-odd">
    <fo:region-start extent='2.5cm'/>
    <fo:region-end extent='2.5cm'/>
    <fo:region-body column-count='2'
       region-name="body" margin='2.5cm'/>
    <fo:region-after region-name="footer-odd" extent="2.5cm"/>
    <fo:region-before region-name="header-odd" extent="2.5cm"/>
</fo:simple-page-master>
```

All the blocks above, including this one, have span='all' set so that they span the whole page.

This block does not have span='all', so it will be fitted into the first column in the page. Text will flow to the bottom of this page and then start at the top of the next column.

If there are blocks above this one on the page which have span='all' (as there are) then they will remain in place and the text which is in only one column will be placed in the next column, below the span='all' blocks.

We deliberately repeat the paragraph to demonstrate this wrapping. This block does not have span='all', so it will be fitted into the first column in the page. Text will flow to the bottom of this page and then start at the top of the next column. If there are blocks above this one on the page (as there are) which have span='all' then then they will remain in place and the text which is in only one column will be placed in the next column, below the span='all' blocks.

It is also possible to have a page start with content in two columns (like this).

When a block-level object is encountered which has span='all' the content already on the page is pushed up to the top, and the block with span='all' is spread over the two columns.

This block has span='all'.

After the content with span='all' the two-column format is resumed.

CHAPTER 15

Tables

A table in XSL-FO is an area of content divided into rows and columns. A table is created with the fo:table element.

A simple table is defined like this:

```
<fo:table>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
            <fo:block>row 1 column 1</fo:block>
      </fo:table-cell>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
            <fo:block>row 1 column 2</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 1</fo:block>
       </fo:table-cell>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 2</fo:block>
       </fo:table-cell>
    </fo:table-row>
</fo:table-body>
```

This XML produces the following table:

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

The padding and border attributes are not inherited from containing elements, so are best defined on the fo:table-cell elements.

15.1. Cell padding

Padding is the amount of space which appears between the inside edge of the border of a cell and the outside edge of the content of the cell. Padding is specified by the padding attribute. The default amount of padding is '0pt'. The following example shows a table with two cells, the first cell has padding='1pt' and the second has padding='5pt'.

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has padding set to '5pt' so the text is not so close to the edges of the cell

The padding attribute sets padding for all four sides of the cell. Individual sides can be set using the padding-left, padding-right, padding-top and padding-bottom attributes.

The padding attribute also supports a shorthand format where

(a) if one value is specified (padding='2pt') the same value will apply to all four sides

(b) if two values are specified (padding='2pt 3pt') the first value will apply to top and bottom, the second value to left and right

(c) if three values are specified (padding='2pt 3pt 1pt') the first value will apply to top, the second to left and right, and the third to bottom

(d) if four values are specified (padding='2pt 3pt 1pt 0pt') the first value will apply to top, right, bottom and left in that order

15.2. Cell background color

The background color of a cell is specified using the background-color attribute. This supports the same predefined colors as CSS and the use of hex values such as '#33ffcc'. The background color of the cell extends to the inside edge of the border, which means that the area specified by the padding attribute is colored by the background color. This is shown here with the second cell having the attribute background-color='#dddddd'.

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has padding set to '5pt' so the text is not so close to the edges of the cell. The background color covers the padding.

If you do not want the background to extend to the edge of the padding, specify the background-color attribute on the contents of the cell (i.e. the fo:block elements) rather than on the fo:table-cell like this:

```
<fo:table>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='1pt'>
            <fo:block>
            this cell has padding set to '1pt' so the text is close to the
edges of the cell
            </fo:block>
      </fo:table-cell>
      <fo:table-cell border='1pt solid blue' padding='5pt'
            background-color='#dddddd'>
            <fo:block background-color='#dddddd'>
            this cell has padding set to '5pt' so the text is not so close to
the edges of the cell
            </fo:block>
      </fo:table-cell>
    </fo:table-row>
</fo:table-body>
```

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has padding set to '5pt' so the text is not so close to the edges of the cell

15.3. Cell background images

An image can be used as the background to a cell by specifying the background-image element, like this:

```
<fo:table>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='1pt'>
            <fo:block>
            this cell has padding set to '1pt' so the text is close to the
edges of the cell
            </fo:block>
      </fo:table-cell>
      <fo:table-cell border='1pt solid blue' padding='5pt'
            background-image='url(ibex.jpg)'>
            <fo:block>
            this cell has a background image
            </fo:block>
      </fo:table-cell>
    </fo:table-row>
</fo:table-body>
```

This produces this result:

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has a background image

As the above example shows, the image will by default be repeated if it is less than the width of the cell. This can be changed using the background-repeat attribute. If this is set to 'no-repeat' the output changes to this:

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has a background image

The background image can be positioned in the cell using the background-position-horizontal and background-position-vertical attributes. This example has background-position-horizontal set to '50%'.

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has a background image

15.4. Implicit and explicit rows

This XML uses the fo:table-row element to define which cells are in which rows.

```
<fo:table>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'>
            <fo:block>row 1 column 1</fo:block>
      </fo:table-cell>
```

```
            <fo:table-cell border='1pt solid blue' padding='2pt'>
                  <fo:block>row 1 column 2</fo:block>
            </fo:table-cell>
          </fo:table-row>
          <fo:table-row>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 1</fo:block>
             </fo:table-cell>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 2</fo:block>
             </fo:table-cell>
          </fo:table-row>
      </fo:table-body>
```

It is possible to dispense with the fo:table-row element and have a fo:table-body contain fo:table-cell elements directly. In this case any cell can have the ends-row attribute set to 'true', which causes a new row to be started containing the next cell. This approach is sometimes easier to use when generating the XSL-FO XML using XSLT.

If we change the above XML to use implicit rows it looks like this:

```
      <fo:table>
        <fo:table-body>
            <fo:table-cell border='1pt solid blue' padding='2pt'>
                  <fo:block>row 1 column 1</fo:block>
            </fo:table-cell>
            <fo:table-cell border='1pt solid blue' padding='2pt'
                  ends-row='true'>
                  <fo:block>row 1 column 2</fo:block>
            </fo:table-cell>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 1</fo:block>
             </fo:table-cell>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 2</fo:block>
             </fo:table-cell>
      </fo:table-body>
```

And the resulting table looks like this:

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

15.5. Table columns

A fo:table can contain a number of fo:table-column elements which define column characteristics such as background-color and column width. A fo:table-column element has an associated column number which determines which column the fo:table-column element refers to. This column number is either implied (with the first fo:table-column element applying to the first column, the second to the next etc., or explitly set using the column-number attribute.

A single fo:table-column element can be made to define the style of multiple columns by using the number-columns-spanned attribute.

This XML shows a table with two fo:table-column elements, which implicitly apply to the first and second columns. In this case they set the column widths (to 30% and 70%), and the give the second column a shaded background.

```
      <fo:table>
          <fo:table-column column-width='30%'/>
          <fo:table-column column-width='70%'
          <fo:table-body>
               background-color='#dddddd'/>
          <fo:table-row>
```

```
            <fo:table-cell border='1pt solid blue' padding='2pt'>
                  <fo:block>row 1 column 1</fo:block>
            </fo:table-cell>
            <fo:table-cell border='1pt solid blue' padding='2pt'>
                  <fo:block>row 1 column 2</fo:block>
            </fo:table-cell>
          </fo:table-row>
          <fo:table-row>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 1</fo:block>
             </fo:table-cell>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 2</fo:block>
             </fo:table-cell>
          </fo:table-row>
      </fo:table-body>
```

This XML produces the following table:

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

The order of precedence in determining cell characteristics such as background-color is fo:table-cell, fo:table-row, fo:table-body, fo:table-column and finally fo:table.

15.6. Proportional column widths

Columns can be allocated widths which are proportional to the widths of other columns. For example if we have two columns and want to give the first column twice the width of the second, we can specifiy column widths using the proportional-column-width() function like this:

```
      <fo:table>
        <fo:table-column
               column-width='proportional-column-width(2)'/>
        <fo:table-column
               column-width='proportional-column-width(1)'
               background-color='#dddddd'/>
        <fo:table-body>
          <fo:table-row>
            <fo:table-cell border='1pt solid blue' padding='2pt'>
                  <fo:block>row 1 column 1</fo:block>
            </fo:table-cell>
            <fo:table-cell border='1pt solid blue' padding='2pt'>
                  <fo:block>row 1 column 2</fo:block>
            </fo:table-cell>
          </fo:table-row>
          <fo:table-row>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 1</fo:block>
             </fo:table-cell>
             <fo:table-cell border='1pt solid blue' padding='2pt'>
                   <fo:block>row 2 column 2</fo:block>
             </fo:table-cell>
          </fo:table-row>
      </fo:table-body>
```

The total of the values used in the proportional-column-width() functions is 3 (2+1), so the first column will gave 2/3 of the width and the second 1/3.

This XML produces the following table:

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

15.7. Colspan and rowspan

The number of columns which a cell spans is set by the number-columns-spanned attribute. This XML shows the first cell of the first row spanning two columns:

```
<fo:table>
  <fo:table-column column-width='30%'/>
  <fo:table-column column-width='70%'
         background-color='#dddddd'/>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'
          number-columns-spanned='2'>
            <fo:block>row 1 column 1</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 1</fo:block>
       </fo:table-cell>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 2</fo:block>
       </fo:table-cell>
    </fo:table-row>
</fo:table-body>
```

This XML produces the following table:

row 1 column 1	
row 2 column 1	row 2 column 2

The number of rows which a cell spans is set by the number-rows-spanned attribute. This XML shows the first cell of the first row spanning two rows:

```
<fo:table>
    <fo:table-column column-width='30%'/>
    <fo:table-column column-width='70%'
         background-color='#dddddd'/>
    <fo:table-body>
    <fo:table-row>
      <fo:table-cell border='1pt solid blue' padding='2pt'
          number-rows-spanned='2'>
            <fo:block>row 1 column 1</fo:block>
      </fo:table-cell>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 1 column 2</fo:block>
       </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
       <fo:table-cell border='1pt solid blue' padding='2pt'>
             <fo:block>row 2 column 1</fo:block>
       </fo:table-cell>
    </fo:table-row>
</fo:table-body>
```

This XML produces the following table:

row 1 column 1	row 1 column 2
	row 2 column 1

This example show a 4 cell x 4 cell table with the middle cell having rowspan and colspan equal to '2'.

row 1 cell 1	row 1 cell 2	row 1 cell 3	row 1 cell 4
row 2 cell 1	row 2 cell 2		row 2 cell 3
row 3 cell 1			row 3 cell 2
row 4 cell 1	row 4 cell 2	row 4 cell 3	row 4 cell 4

15.8. Cell Separation

XSL-FO has two different ways of handling the borders of adjacent cells depending on the value of the border-collapse attribute on the table.

If border-collapse='collapse', which is the default, there is no gap between cells and the borders of adjacent cells are merged (or "collapsed") to get a single border shared by both cells. The rules for combining borders are explained in the XSL-FO specification. Broadly speaking the widest border will be used. This is called the collapsed border model.

If border-collapse='separate' adjacent borders are not merged. A gap can be inserted between adjacent borders using the border-spacing attribute. The border-spacing atrtibute can have one or two values. If one value is specified (for instance border-spacing='1mm') the vertical and horizontal spacing between cells is set to this value. If two values are specified separated by a space (for instance border-spacing='1mm 3mm') the horizontal separation is set to the first value and the vertical separation is set to the second. This is called the separated border model.

The following examples use a table with one row containing two cells. The first cell has a bottom border, the second does not. The table also has a bottom border.

In the separate border model the border from the first cell will be drawn before the border of the table like this:

separate border model

this cell has a bottom border	this cell does not have a bottom border

In the collapsed border model the border from the first cell will combined with the border of the table an a single border will be drawn like this:

collapsed border model

this cell has a bottom border	this cell does not have a bottom border

If we add a inner border to each cell we can see this with the separate model:

separate border model

this cell has a bottom border	this cell does not have a bottom border

With the collapsed border model the border between the two cells will be half the with it is in the separate model, like this:

collapsed border model

this cell has a bottom border	this cell does not have a bottom border

This is an example of the separate table:

separate border model

cell one	cell two
cell three	cell four

See how the border-spacing on the previous table only sets the space between cells, not the space between the cell and the table border. This space can be set using padding. If we add padding='2mm' to the table we get this:

cell one	cell two
cell three	cell four

before
start
end
after

CHAPTER 16

Images

Images are added to the document using either the fo:external-graphic or fo:instream-foreign-object elements. Use the fo:external-graphic element to include a file in JPEG, GIF, TIFF, BMP, or PNG formats. Use the fo:instream-foreign-object element to include an image defined in Scalable Vector Graphics (SVG) format.

The properties used to format the fo:external-graphic and fo:instream-foreign-object elements are the same.

The size of the image is distinct from the size of the area in which the image is placed.

The height and width attributes specify the size of the area into which the graphic will be placed. If these properties are not specified they default to an area large enough to contain the image.

The content-width and content-height attributes control the size of the image. These can be values such as '3cm' or percentages such as '120%'. If not specified the image defaults to the size in pixels specified in the image file itself.

This means that if you do not specify any of the above attributes the image will be as large as specified in the image file, and will be placed in an area the same size.

An image is an inline element, so for formatting purposes it can be placed in a sentence surrounded by text and is treated as a single large word.

16.1. Using fo:external-graphic

The fo:external-graphic element is used to include an image which is in a file external to the formatting objects XML. The name of the file to be included is specified using the src attribute.

The src attribute is called a *uri-specification* and must follow the following rules:

> A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

This means the following are all valid values for the src attribute:

uri(ibex.jpg)

uri("ibex.jpg")

uri('ibex.jpg')

url(http://www.xmlpdf.com/images/download2.gif)

As the src attribute is a URL, an image which exists on a web server can be downloaded automatically by Ibex as the PDF file is created. The following XML will fetch the file download2.gif from www.xmlpdf.com as the document is created:

```
<fo:block space-before="6pt">
 <fo:external-graphic border="1pt solid black"
     src="url(http://www.xmlpdf.com/images/download2.gif)"
     content-width="200%" content-height="200%"/>
</fo:block>
```

This XML displays this image:

The fo:external-graphic element can be used to include image files in PNG, JPEG, TIFF, BMP and GIF formats. It can also be used to include SVG images held in external files.

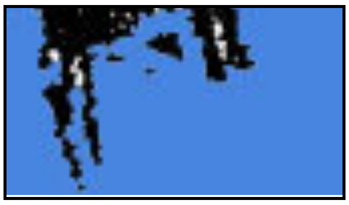
The fo:inline-foreign-object is used for loading images from SVG which is contained inline in the XSL-FO XML. See SVG Images on page 74.

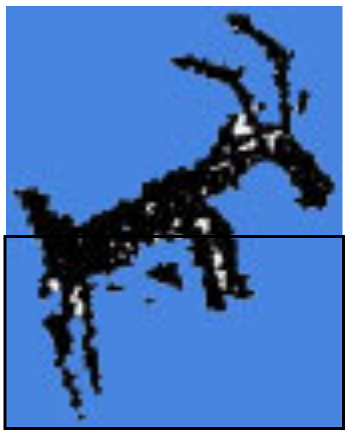
16.2. Clipping

If the image ends up larger than the area in which it is contained then the area *may* be clipped. This is shown here:

First the image at its natural size:

If we specify the height of the fo:external-graphic element as 2.5cm and specify overflow="hidden", the image will be clipped to this height, like this:

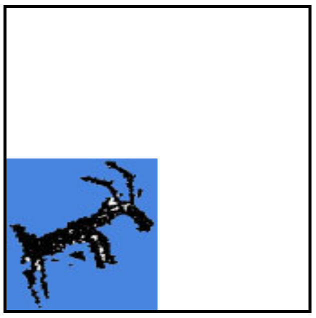




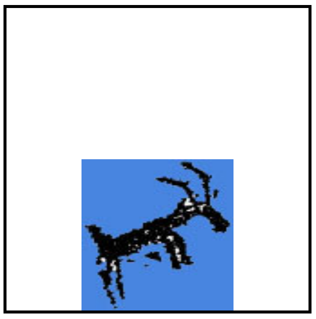
If we specify the height of the fo:external-graphic element as 2.5cm and and do not specify overflow="hidden", the image will not be clipped to this height, and will overwrite other content as shown to the right. Because the image is positioned on the same baseline as text, the overflow will be at the top of the area containing the image.

16.3. Image size and alignment

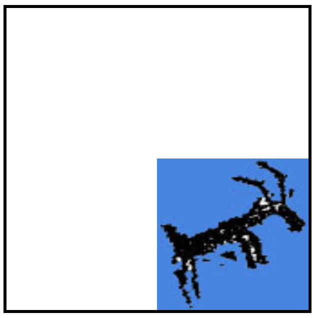
If the image is smaller than the containing area we can control where it appears in that area using the display-align and text-align attributes. The display-align attribute controls the vertical alignment, and text-align controls the horizontal alignment. By default the image appears in the top left corner of the inline area created by the fo:external-graphic or fo:instream-foreign-object element like this:



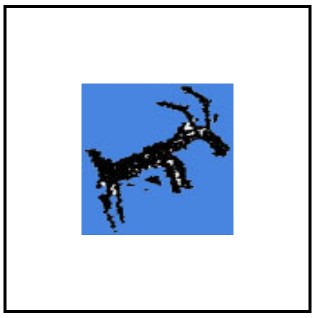
If we specify text-align='center' the image will move to the horizontal center of the inline area, like this:



If we specify text-align='right' the image will move to the right of the inline area, like this:



If we specify text-align='center' and display-align='center' the image will move to the horizontal and vertical center of the inline area, like this:



16.4. Image Resolution

The resolution of an image in dots per inch (dpi) can be set using the dpi attribute on the fo:external-graphic element. For example if we wanted to store an image in the PDF file at 1200dpi, we would use this XML.

```
<fo:block space-before="6pt">
 <fo:external-graphic border="1pt solid black"
     src="url(http://www.xmlpdf.com/images/download2.gif)"
     content-width="200%" content-height="200%"
     dpi='1200'/>
</fo:block>
```

Changing the dpi value has a significant impact on the size of the PDF file. It is a simple way to reduce the size of the PDF file by reducing the resolution of the images. If you know a document is intended only for displaying in a screen, which is typically 72dpi, or a printer, which is typically 300dpi, then you can reduce your file size by reducing image quality accordingly.

The dpi attribute is an Ibex extension. It is not part of the XSL-FO standard.

Scalable Vector Graphics (SVG) images

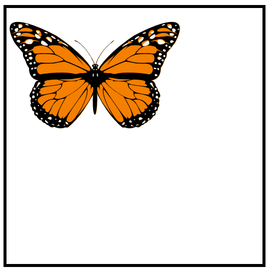
SVG support is provided by the SVG# library. Currently version 0.3 of SVG# is used. The SVG# code is compiled into the Ibex DLL (since Ibex 2.2).

SVG images held in external files (either locally or remotely) are loaded using the fo:external-graphic element in the same was any any other image type, using XML like this:

```
<fo:block space-before="6pt">
```

```
        <fo:external-graphic border="1pt solid black"
            src="url(butterfly.svg)"
            content-width="20%" content-height="20%"/>
       </fo:block>
```

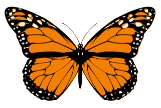
This produces the following image:



If the image is held inline within the XSL-FO XML it can be loaded using the fo:instream-foreign-object element. An example of this XML is shown below. The actual content of the SVG image has been shortened to fit on this page.

```
<fo:instream-foreign-object width="20%" height="1cm">
    <svg xmlns:fo="http://www.w3.org/TR/xsl/Format"
                xmlns="http://www.w3.org/2000/svg">
         <path style="stroke-width:1;fill:rgb(246,127,0);"
              d="M204.33 139.83 C196.33 133.33 z" />
    </svg>
</fo:instream-foreign-object>
```

The image created by the fo:instream-foreign-object is shown here:



Loading an Image from Memory

Ibex has the facility to load an image which is stored in memory. This permits an application to dynamically generate an image which can the be placed in the PDF file.

The image must be stored in a byte array or a Stream (from the System.IO namespace).

The image must be given a unique identifier by which it can be retrieved during the PDF creation process. This is done using the addNamedImage() method, which takes a string which identifies the image and the image itself.

For example if we had an image in a byte array called 'image' and we wanted to give it the identifier '1029' we would do this:

```
byte[] image = ... dynamically create

FODocument document = new FODocument();

document.addNamedImage( "1029", image );
```

This should be done before calling generate() to create the PDF file.

Within the FO file the image is retrieved from memory using the following syntax:

```
<fo:external-graphic src="url(data:application/ibex-image,1029)"/>
```

The value of the src attribute must have the string "url(data:application/ibex-image," followed by the unique identifier which was passed to addNamedImage().

This syntax for the url attribute conforms to http://www.faqs.org/rfcs/rfc2397.html.

16.5. Transparent Images

Ibex can use to make some parts of an image appear transparent. This is an extension to the XSL-FO standard.

Image masking works by defining colors from the image which should not be displayed. Adobe Acrobat will not compare each pixel in the image with the mask and not display pixels which match the mask, effectively making these pixels transparent and so displaying the content behind the image.

The image mask is defined using the <ibex:image> element, which must be contained within a fo:external-graphic element, like this:

```
<fo:external-graphic src="url(ixsl.jpg)"  z-index='10'>
 <ibex:mask
        red-min='255' red-max='255'
        green-min='255' green-max='255'
        blue-min='255' blue-max='255'/>
</fo:external-graphic>
```

To use the ibex:mask element you must reference the ibex namespace in your FO file like this:

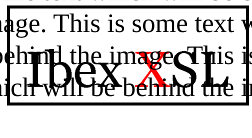
```
<fo:root
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">
```

The mask defines the minimum and maximum values for each of the red, green and blue components of an image. It is therefore applicable only to images which are in RGB format with 24 bits per pixel. This may change in later releases.

The image mask shown above has this effect: any pixel which has red=255, green=255 and blue=255 will not be rendered. As a pixel with red, green and blue all equal to 255 is white, this means any white pixels will not be rendered.

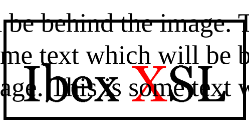
The following shows some text over which we have placed an image with red and black letters on a white background:

This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image.



If we add add a mask to eliminate white pixels the image then looks like this:

This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image.



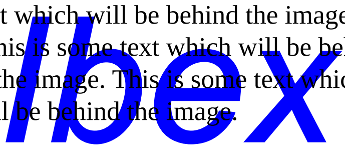
Transparent Images using SVG

Transparent images can also be implemented by placing a SVG image over the top of other content. This approach uses the vector SVG renderer introduced in Ibex 2.1.2 and is only available when using the .NET Framework 1.1 or higher. This is the best approach for transparent images because there is no background on the SVG image so the best clarity is acheived, and because the vector renderer creates a smaller PDF file than using an image.

If we want to put the word 'ibex' over some text, the XML looks like this:

```
<fo:block-container space-before="6pt"
        relative-position='relative' top='-1.6cm'
        left='5cm'>
  <fo:block>
     <fo:instream-foreign-object z-index="30">
        <svg width="315" height="100"
            xmlns="http://www.w3.org/2000/svg">
             <text  x="30" y="60" fill='blue' stroke='blue'
                font-size='61pt' font-style='italic'
                style="font-family:arial;stroke-width:0.5">
                  Ibex
             </text>
        </svg>
     </fo:instream-foreign-object>
  </fo:block>
</fo:block-container>
```

This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image. This is some text which will be behind the image.



CHAPTER 17

Absolute Positioning

Content can be positioned anywhere on the page by placing the content in a fo:block-container element and setting the absolute-position attribute to "absolute". The content will then be positioned on the page *relative* to the area which contains the fo:block-container element.

The content contained in the fo:block-container element will be positioned offset from the top left corner of the containing area by the amount specified in the of the left and top attributes.

For example this XML outputs the word 'hello', with an image positioned 3 cm to the left of the word.

```
<fo:block space-before='1cm' space-after='1cm'>
Hello
  <fo:block-container absolute-position='absolute' left='3cm'
       top='-14pt'>
    <fo:block>
      <fo:external-graphic src='url(ibex.jpg)'
         content-height='1cm' scaling='uniform'/>
    </fo:block>
  </fo:block-container>
</fo:block>
```

Hello

We can also move the image to the left of the text by setting a negative left attribute value like this:

```
<fo:block>
Hello
  <fo:block-container absolute-position='absolute' left='-1.5cm' top='0pt'>
     <fo:block>
  <fo:external-graphic src='url(ibex.jpg)' content-height='1cm'
scaling='uniform'/>
     </fo:block>
  </fo:block-container>
</fo:block>
```

Hello

In both cases the absolutely positioned content has no effect on the rest of the document.

If the fo:block-container element is within a fo:block element then the content will be positioned relative to that block element.

If the fo:block-container element is not within a fo:block element, i.e. is an immediate descendent of a flow or fo:static-content element, then the content will be positioned relative to the containing region.

17.1. Content Size

The size of absolutely position content (and indeed any content) can be controlled using the height and width attributes. The following example shows a block which has its height and width set to 5 cm, and has a background color to show the size of the block:

```
<fo:block-container absolute-position='absolute' left='1.5cm'
  top='17cm' width='5cm' height='5cm'
  background-color='#777777'>
  <fo:block>
     This content will be positioned and word-wrapped
     to a width of 5 cm.
  </fo:block>
</fo:block-container>
```

As this fo:block-container element appears in the XML immediately below the fo:flow element it will be positioned relative to the body region.

This content will be positioned and word-wrapped to a width of 5 cm. Top is 17cm so the content will start 17cm down from the top of the containing area, which in this case is the body region.

CHAPTER 18

Configuration

All configuration of Ibex is done using the xslfo.UserAgent class. This class has many static properties which can be changed to configure the operation of Ibex.

Properties of the UserAgent class should be changed prior to calling the generate() method on the FODocument object. This example shows how to set the default line height to 1.4em.

```
using System;
using xslfo;

public class IbexTester {

   public static void Main(string[] args) {

      UserAgent.LineHeightNormal = "1.4em";

      using( Stream xml =
             new FileStream( args[0], FileMode.Open, FileAccess.Read ) ) {
          using ( Stream output =
              new FileStream( args[1], FileMode.Create, FileAccess.Write ) ) {
                    new FODocument().generate( xml, output );
               }
          }
   }
}
```

18.1. Fonts

The following properties on the UserAgent change the way fonts a processed. By default each absolute font size (small,medium,large etc.) is 1.2 times larger than the previous size.

Property	Type	Default	Notes
UserAgent.XX_Small	string	7.0pt	Must end in 'pt'.
UserAgent.X_Small	string	8.3pt	Must end in 'pt'.
UserAgent.Small	string	10.0pt	Must end in 'pt'.
UserAgent.Medium	string	12.0pt	Must end in 'pt'.
UserAgent.Large	string	14.4pt	Must end in 'pt'.
UserAgent.X_Large	string	17.4pt	Must end in 'pt'.
UserAgent.XX_Large	string	20.7pt	Must end in 'pt'.

Property	Type	Default	Notes
UserAgent.Smaller	string	0.8em	Must end in 'em'.
UserAgent.Larger	string	1.2em	Must end in 'em'.

18.2. Line Height

The following properties on the UserAgent change the default line height. Ideally UserAgent.LineHeightNormal should end in 'em' to make line height proportional to the font height.

Property	Type	Default	Notes
UserAgent.LineHeightNormal	string	1.2em	

18.3. Page Size

The following properties on the UserAgent change the default page size.

Property	Type	Default	Notes
UserAgent.PageHeight	string	297mm	
UserAgent.PageWidth	string	210mm	

18.4. Include Paths

The following properties on the UserAgent effect retrieving XML or XSL files.

Property	Type	Default	Notes
UserAgent.BaseURI_XML	string		This value sets the base URI for including images and other XML files. When an fo:external-graphic element specifies a relative path, UserAgent.BaseURI_XML is the base URI used in accordance with the rfc2396 URI Specification. When an XML file uses an entity to include another XML file, UserAgent.BaseURI_XML is the base URI used when Ibex searches for the included XML file.
UserAgent.BaseURI_XSL	string		This value sets the base URI for including other XSL files. When an xsl:include element is used to include another XSL stylesheet, UserAgent.BaseURI_XSL can be used to specify the location the included stylesheet should be loaded from.

18.5. Images

The following properties on the UserAgent effect retrieving images specified using the fo:external-graphic element.

Property	Type	Default	Notes
UserAgent.BaseURI_XML	string		This value sets the base URI for including images and other XML files. When an fo:external-graphic element specifies a relative path, UserAgent.BaseURI_XML is the base URI used in accordance with the rfc2396 URI Specification. When an XML file uses an entity to include another XML file, UserAgent.BaseURI_XML is the base URI used when Ibex searches for the included XML file.
UserAgent.WebRequestTimeoutMs	int	300	When an fo:external-graphic element specifies an image is retrieved from a web server, this is the timeout used for the call to the web server. Units are milliseconds.

18.6. Border Widths

The following properties on the UserAgent change the values for border widths specified with constants like 'thin'.

Property	Type	Default	Notes
UserAgent.BorderWidthThin	string	1pt	
UserAgent.BorderWidthMedium	string	2pt	
UserAgent.BorderWidthThick	string	3pt	

18.7. Layout

The following properties on the UserAgent change the appearance of content in the PDF file.

Property	Type	Default	Notes
UserAgent.OverflowIsVisible	bool	true	By default a region has overflow='auto', leaving it up the Ibex to decide whether content which overflows the bottom edge of a region is displayed or discarded. If UserAgent.OverflowIsVisible is true, the content will be displayed, if false it will be discarded. This property applies only if the XSL-FO attribute 'overflow' is not set or is set to 'auto'.

18.8. Leaders

The following properties on the UserAgent change the values for fo:leader formatting objects.

Property	Type	Default	Notes
UserAgent.LeaderDot	char	.	When leader-pattern='dots', this is the character used as the dot

CHAPTER 19

Using Ibex with Visual Basic and ASP

Ibex ships with binaries and source code for a COM wrapper which enables Ibex to be used from Visual Basic 6.0 and ASP applications. This chapter describes how this COM wrapper works.

19.1. The COM Wrapper

Within the Ibex distribution is a subdirectory called 'ibexcom'. In this directory is the source for a .NET project, in the file ibexcom.csproj. This source is used to build the file ibexcom.dll which is the COM wrapper for Ibex.

19.2. Building the Wrapper

The COM wrapper can be build using the 'nmake' command. The makefile executes the following commands to build and register ibexcom.dll and ibexcom.tlb:

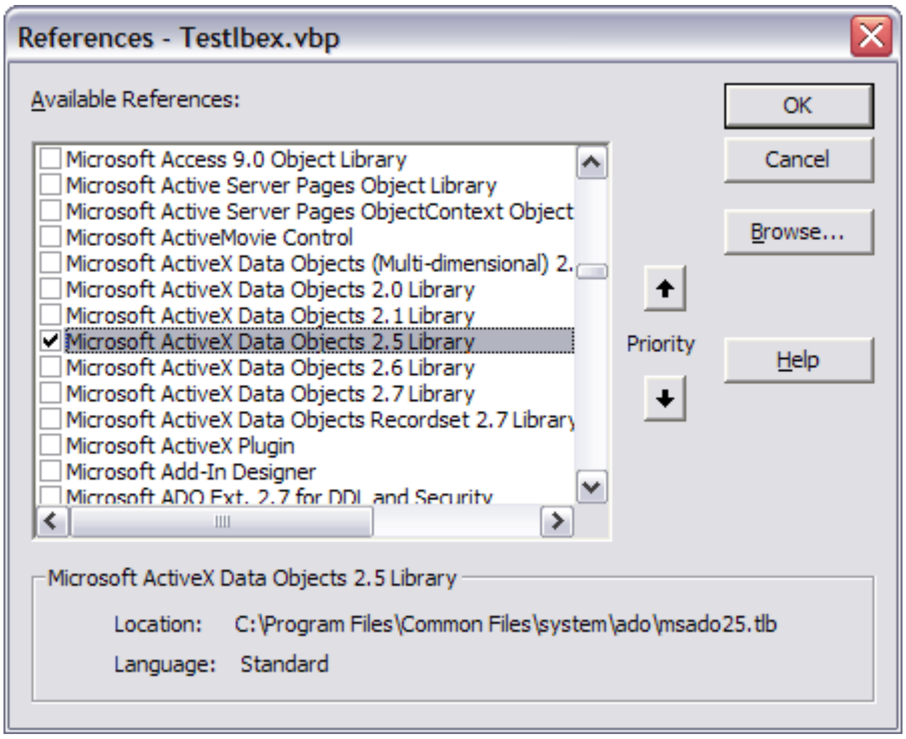
```
   copy ..\ibex11.dll ibex11.dll
   csc /r:ibex11.dll /target:library /out:ibexcom.dll /unsafe assemblyinfo.cs
wrapper.cs comstream.cs ibexinterface.cs
   regasm ibexcom.dll /tlb
   gacutil -i ibexcom.dll
   gacutil -i ibex11.dll
```

19.3. VB6 Example

This section shows how to use Ibex from VB6. Ibex ships with an example VB6 project in the file ibextest.vbp, the steps below detail how this project was created.

The XML files used in this example (book.xml, hello.fo) ship with Ibex and are in the ibexcom directory.

Create a new VB 6 project. Use the Project->References menu option to add a reference to the both ADO 2.5 and ibexcom.tlb file. The dialog should look something like this:



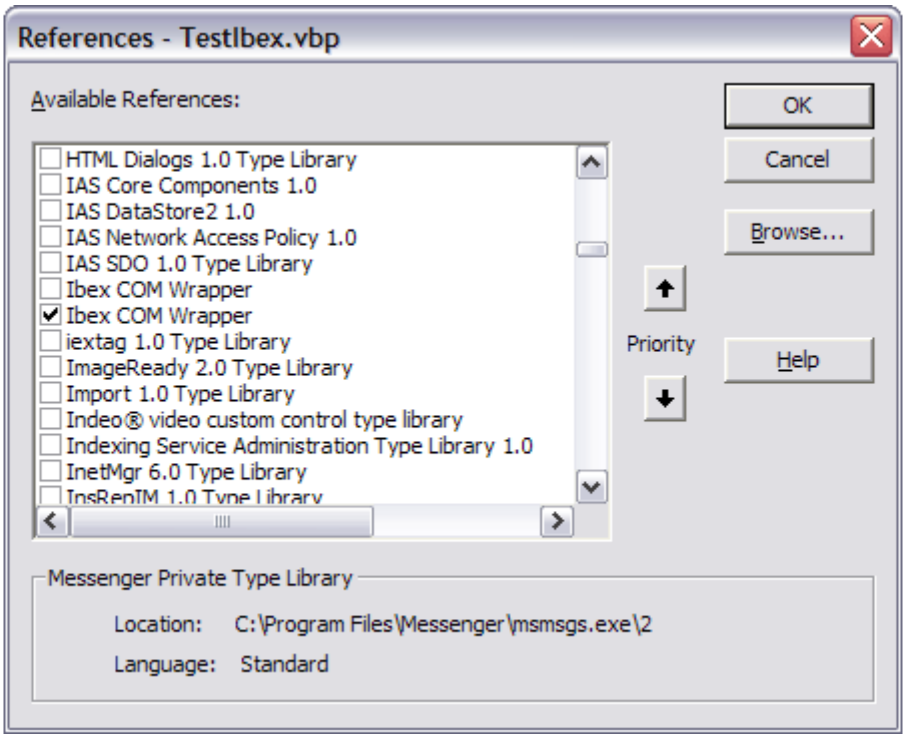
- [ ] Microsoft Access 9.0 Object Library
- [ ] Microsoft Active Server Pages Object Library
- [ ] Microsoft Active Server Pages ObjectContext Object
- [ ] Microsoft ActiveMovie Control
- [ ] Microsoft ActiveX Data Objects (Multi-dimensional) 2.
- [ ] Microsoft ActiveX Data Objects 2.0 Library
- [ ] Microsoft ActiveX Data Objects 2.1 Library
- [✓] Microsoft ActiveX Data Objects 2.5 Library
- [ ] Microsoft ActiveX Data Objects 2.6 Library
- [ ] Microsoft ActiveX Data Objects 2.7 Library
- [ ] Microsoft ActiveX Data Objects Recordset 2.7 Library
- [ ] Microsoft ActiveX Plugin
- [ ] Microsoft Add-In Designer
- [ ] Microsoft ADO Ext. 2.7 for DDL and Security

Microsoft ActiveX Data Objects 2.5 Library

Location: C:\Program Files\Common Files\system\ado\msado25.tlb

Language: Standard

and



- [ ] HTML Dialogs 1.0 Type Library
- [ ] IAS Core Components 1.0
- [ ] IAS DataStore2 1.0
- [ ] IAS Network Access Policy 1.0
- [ ] IAS SDO 1.0 Type Library
- [ ] Ibex COM Wrapper
- [✓] Ibex COM Wrapper
- [ ] iextag 1.0 Type Library
- [ ] ImageReady 2.0 Type Library
- [ ] Import 1.0 Type Library
- [ ] Indeo® video custom control type library
- [ ] Indexing Service Administration Type Library 1.0
- [ ] InetMgr 6.0 Type Library
- [ ] InsRepIM 1.0 Type Library

Messenger Private Type Library

Location: C:\Program Files\Messenger\msmsgs.exe\2

Language: Standard

The COM wrapper provides an interface called IbexComInterface, which can be used to declare a variable like this:

```
Dim ibex As ibexcom.IbexCOMInterface
```

The implementation of the interface is called Wrapper, and can be created like this:

```
 Set ibex = New ibexcom.Wrapper
```

Note that each instance of the wrapper can be used to create only one PDF document.

The code contained in the example ibextest.vbp project is:

```
 Private Sub Form_Load()

On Error GoTo errorhandler

    Dim ibex As ibexcom.IbexCOMInterface


    ' create the wrapper object
    Set ibex = New ibexcom.Wrapper

    ibex.SetLoggingLevelToInfo
    ibex.LogToFile "ibex.log"

    ' test setting a useragent value
    ibex.BaseURI = "d:\xmlpdf"
    ibex.DefaultFontFamily = "courier"

    ' generate file to file
    ibex.GenerateFileFile "hello.fo", "hello.pdf"

    Set ibex = Nothing

    ' generate stream to stream
    Set ibex = New ibexcom.Wrapper

    Dim strmInput As New Stream

    strmInput.Open
    ' must match encoding of fo file or will get "invalid data at 1,1" message
    strmInput.Charset = "utf-8"
    strmInput.LoadFromFile "hello.fo"
    strmInput.Position = 0


    Dim strmPDF As New Stream
    strmPDF.Open

    ibex.GenerateXMLStreamPDFStream strmInput, strmPDF, True

    Set strmInput = Nothing

    ' save the PDF stream to file, just to show it works
    strmPDF.Position = 0

    ' will get write error here if file exists
    strmPDF.SaveToFile "hellostream.pdf"
    Set strmPDF = Nothing

    Set ibex = Nothing

    ' test xslt translation and pdf creation
    Set ibex = New ibexcom.Wrapper

    Dim strmXML As New Stream
```

```
          strmXML.Open
          ' must match encoding of fo file or will get "invalid data at 1,1" message
          strmXML.Charset = "utf-8"
          strmXML.LoadFromFile "book.xml"
          strmXML.Position = 0

          Dim strmXSL As New Stream

          strmXSL.Open
          ' must match encoding of fo file or will get "invalid data at 1,1" message
          strmXSL.Charset = "utf-8"
          strmXSL.LoadFromFile "book.xsl"
          strmXSL.Position = 0

          Set strmPDF = New Stream
          strmPDF.Open

          ibex.GenerateXMLStreamXSLStreamPDFStream strmXML, strmXSL, strmPDF, True

          Set strmXML = Nothing
          Set strmXSL = Nothing

          ' save the PDF stream to file, just to show it works
          strmPDF.Position = 0
          strmPDF.SaveToFile "helloxsl.pdf"
          Set strmPDF = Nothing

          Set ibex = Nothing


          End

      errorhandler:
          MsgBox Err.Description
          End

      End Sub
```

Key things to note are:

A PDF file can be created from an FO file to a PDF file on disk using this call:

```
          ibex.GenerateFileFile "hello.fo", "hello.pdf"
```

A PDF file can be created from FO contained in Stream to a PDF Stream using this call:

```
          ibex.GenerateXMLStreamPDFStream strmInput, strmPDF, True
```

The final parameter (True in this case) indicates the output stream should be closed after the PDF file is created.

The input Stream object is part of the ADO 2.5 namespace, which is why we added a reference to ADO 2.5 earlier on. A stream can be populated from a file on disk using code like this:

```
          strmInput.Open

          ' must match encoding of fo file or will get "invalid data at 1,1" message
          strmInput.Charset = "utf-8"

          strmInput.LoadFromFile "hello.fo"
          strmInput.Position = 0
```

To create a PDF file using XSLT translation, use this call, with the XML and XSL read from Stream objects and the PDF written to a Stream:

```
            ibex.GenerateXMLStreamXSLStreamPDFStream strmXML, strmXSL, strmPDF, True
```

Appendix A.

Extensions

This chapter details Ibex-specific extensions to the XSL-FO XML.

A.1. Ibex Version

The ibex:version element inserts the version number of Ibex used to create the PDF file. This is an inline element which inserts characters into the document.

For example this XML:

```
<fo:block>
  the version is <ibex:version/>
</fo:block>
```

produces this output:

the version is 3.9.21

The Ibex extensions have a namespace which is specified using the xmlns attribute as shown above.

The ibex:version element must occur before any output is generated.

A.2. Document Security

Ibex implements encryption of PDF documents and the setting of various document permissions. This is done using the ibex:security element like this:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:security deny-print='true' deny-extract='true'
 deny-modify='true' user-password='user' owner-password='owner'/>

...
```

The Ibex extensions have a namespace which is specified using the xmlns attribute as shown above.

The ibex:security element must occur before any output is generated.

The attributes of the ibex:security element are:

Attribute	Values	Meaning
user-password		Specifies a password required to open the document in Acrobat. Once the document is opened with the correct user password, access is limited to permissions given using the attributes below.
owner-password		Specifies a password required to get all rights to the document in Acrobat. Once the document is opened with the correct owner password the user has total control of the document.
deny-print	true false	If this is set to true a user who opens the document with the user password will not be able to print the document.
deny-extract	true false	If this is set to true a user who opens the document with the user password will not be able to use cut-and-paste functionality to copy part of the document.
deny-modify	true false	If this is set to true a user who opens the document with the user password will not be able to modify the document.

Setting any of the attributes listed above will cause Ibex to encrypt the document.

Specifying the user-password but not the owner-password will set the owner-password to the same value as the user-password. This means anyone who can open the document using the user password has complete control of the document.

Specifying the owner-password but not the user-password is common usage. This means the user can open the document with limited rights without needing a password, but cannot then change or exceed those rights without knowing the owner password.

A.3. Standard Document Properties

Ibex allows you to set the various properties associated with a PDF document. These properties can be viewed in Acrobat by using the File | Document Properties | Summary menu option or just pressing control-d.

The properties are set using the ibex:properties element like this:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:properties
     title='Ibex User Manual'  subject='Ibex'
     author='visual programming limited'
     keywords='xml,pdf' creator='xtransform' />
...
```

The Ibex extensions have a namespace which is specified using the xmlns attribute as shown above.

The ibex:properties element must occur before any output is generated.

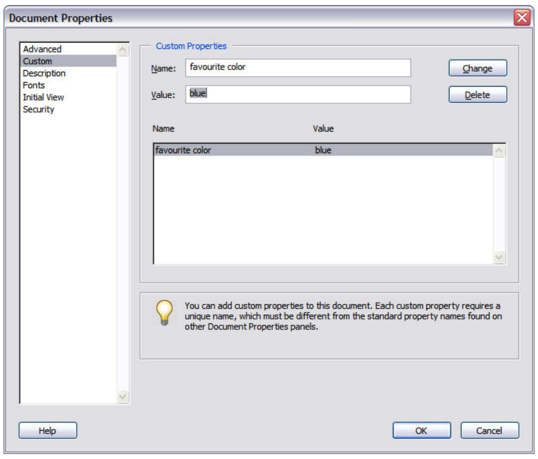
The attributes of the ibex:properties element are:

Attribute	Values	Meaning
title		Specifies a string which becomes the title property of the document.
subject		Specifies a string which becomes the subject property of the document.
author		Specifies a string which becomes the author property of the document.
keywords		Specifies a string which becomes the keywords property of the document. Separate individual keywords with commas.
creator		Specifies a string which becomes the creator property of the document. This should be the name of the application which created the XSL-FO document from which the PDF file was created.
page-mode	none bookmarks thumbs fullscreen	Specifies how Acrobat will display the document when it is first opened. If set to 'bookmarks' then if the document has bookmarks they will be displayed. If set to 'thumbs' then the thumbnails tab in Acrobat will be displayed. If set to 'fullscreen' the document will be displayed without any toolbar, border etc.

Following the PDF standard, the document creator property should be the name of the product which converted the content to PDF format, so this is always Ibex. Other document properties such as creation and modication date are populated automatically by Ibex.

A.4. Custom Document Properties

Acrobat supports the display and editing of custom document properties. These properties are a set of name value pairs stored within the PDF file. In Acrobat 6.0 these properties can be viewed by using the File | Document Properties menu option and clicking on the "Custom" entry in the list box to display a screen like this:



Name: favourite color

Value: blue

Name	Value
favourite color	blue

You can add custom properties to this document. Each custom property requires a unique name, which must be different from the standard property names found on other Document Properties panels.

These custom properties are inserted into the PDF usin the <ibex:custom> element like this:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:properties title='Ibex User Manual'>
    <ibex:custom name='favourite color' value='blue'/>
</ibex:properties>
...
```

Each property must have a name and value attribute.

A.5. Image Resolution

Ibex adds the dpi attribute to the external-graphic element to permit managing the dots per inch resolution of images. See Image Resolution

A.6. Bookmarks

Bookmarks are a PDF specific feature and not supported in the XSL-FO standard.

Bookmarks are inserted into the document using the <ibex:bookmark> element.

Each bookmark element has the following attributes:

Attribute	Values	Meaning
internal-destination		Similarly to the internal-destination attribute of a fo:basic-link element, this should match the id attribute of element within the content which Acrobat will move to when the bookmark is clicked.
title		The text which appears on the bookmark
open	true false	A value which specifies if, for a bookmark which has child bookmarks, those child bookmarks are visible when the PDF file is first opened.

The ibex:bookmark elements should appear after the fo:layout-master-set and before the first fo:page-sequence element.

ibex::bookmark elements can be nested, to create a nested bookmark structure in the PDF document. For example the Ibex manual has the following bookmark elements to create the Introduction and Usage sections of the bookmark tree.

```
<ibex:bookmark internal-destination="chapter-refid-intro"
               title="Introduction" open="false"/>
<ibex:bookmark internal-destination="chapter-refid-2"
               title="2. Usage" open="false">
        <ibex:bookmark internal-destination="ahead-refid-2.1"
                  title="2.1. Ibex command line program"/>
        <ibex:bookmark internal-destination="ahead-refid-2.2"
                  title="2.2. Ibex API"/>
        <ibex:bookmark internal-destination="ahead-refid-2.3"
                  title="2.3. Generating to File"/>
        <ibex:bookmark internal-destination="ahead-refid-2.4"
                  title="2.4. Generating Using Streams"/>
</ibex:bookmark>
```

As the title attribute is specified on the ibex:bookmark element there is no requirement that the bookmark text match any text which appears in the document. The only requirement is that the internal-destination attribute match the id attribute of some block in the document.

A.7. Document Base URL

The PDF format supports setting a base URL for links created with a fo:basic-link element. This base URL is prepended to the destination specified with an external-destination attribute if (and only if) the specified destination does not start with a '/' character.

For example the following XML creates a document with 'http://www.xmlpdf.com' as the base URL and a link to the page 'index.html'. When the user clicks on the link in the PDF file, it will go to 'http://www.xmlpdf.com/index.html'.

```
<ibex:document-base-url value="http://www.xmlpdf.com"/>

..


<fo:block>
   <fo:basic-link external-destination='url(index.html)'>
      index.html
   </fo:basic-link>
</fo:block>
```

The base URL is a document-wide property and can be set only once.

This property should not be confused with the UserAgent.BaseURI value which specifies a base URI to be used when Ibex retrieves images, stylesheets and XML during creation of the PDF file.

Appendix B.

PDF/X

This chapter details Ibex-specific extensions to the XSL-FO XML to support creation of PDF files which conform to the PDF/X standard.

Ibex implements the PDF/X standard using the ibex:pfdx element like this:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:pdfx />

...
```

The Ibex extensions have a namespace which is specified using the xmlns attribute as shown above.

The ibex:pdfx element must occur before any output is generated.

Using the ibex:pdfx element will automatically set the document color space to CMYK.

The existence of the ibex:pdfx element causes Ibex to create a PDF/X compatible PDF file. The field UserAgent.PDFXMode used in earlier releases has been removed.

The attributes of the ibex:pdfx element are:

Attribute	Values	Meaning
color-profile-file-name		Full or relative path to a ICC color profile file
registry-name		Registry Name used in the PDF OutputIntents structure. If not specified this defaults to "http://www.color.org".
info		Optional text which will become the Info value in the first OutputIntents array entry.
output-condition-identifier		Optional text which will become the OutputConditionIdentifier value in the first OutputIntents array entry. This defaults to "Custom"
output-condition		Optional text which will become the OutputCondition value in the first OutputIntents array entry. This defaults to "Custom". Acrobat proposes values such as "TR001 SWOP/CGATS".

The color profiles is read from the specified ICC file, compressed, and embedded in the PDF file.

Media Box

The MediaBox size within the PDF file will be set to the size of the page as specified on the fo:simple-page-master for that page.

Bleed Box

The BleedBox size defaults to the MediaBox size. The BleedBox can be specified as a change from the MediaBox by specifying the ibex:bleed-width attribute on the fo:simple-page-master. This attribute specifies the distance by which the BleedBox is smaller than the MediaBox like this:

```
<fo:simple-page-master page-height="313mm" page-width="226mm"
     master-name="page" ibex:bleed-width="3mm">
```

If one value is used it applies to all sides of the page, if two values are used the top and bottom edges use the first value and the left and right edges use the second. If there are three values the top is set to the first value, the sides are set to the second value, and the bottom is set to the third value. If there are four values, they apply to the top, right, bottom and left edges in that order.

The following attributes can be specified to set each side expliclity: bleed-top-width, bleed-bottom-width, bleed-right-width, bleed-left-width.

Trim Box

The TrimBox size defaults to the BleedBox size. The TrimBox can be specified as a change from the BleedBox by specifying the ibex:trim-width attribute on the fo:simple-page-master. This attribute specifies the distance by which the TrimBox is smaller than the BleedBox like this:

```
<fo:simple-page-master page-height="313mm" page-width="226mm"
    master-name="page" ibex:trim-width="3mm">
```

If one value is used it applies to all sides of the page, if two values are used the top and bottom edges use the first value and the left and right edges use the second. If there are three values the top is set to the first value, the sides are set to the second value, and the bottom is set to the third value. If there are four values, they apply to the top, right, bottom and left edges in that order.

The following attributes can be specified to set each side expliclity: trim-top-width, trim-bottom-width, trim-right-width, trim-left-width.

Overprint

Overprint mode can be enabled for the entire page by specifying the ibex:ibex-overprint-stroking, ibex:overprint-nonstroking and ibex:overprint-mode attributes like this:

```
<fo:simple-page-master page-height="313mm" page-width="226mm"
  master-name="page" ibex:overprint-stroking="true"
  ibex:overprint-nonstroking="true" ibex:overprint-mode="1">
```

Appendix C.

Formatting Object/Property Reference

This chapter describes each major formatting object and its usage.

C.1. fo:block

Description

This element is the main container for text content. The simplest block element looks like this:

```
<fo:block>this is text</fo:block>
```

The fo:block is a block-level element. The other block-level elements are fo:table, fo:list-block, and fo:block-container.

A fo:block element can contain other block-level elements as well as text. A typical usage would be to insert an empty block into a paragraph of text to cause a line break, like this:

```
<fo:block>this will be line 1</fo:block>
<fo:block/>
<fo:block>this will be line 2</fo:block>
```

Another use of nested blocks is to keep two other block-level objects together by using the keep-together attribute on the previous block, like this:

```
<fo:block keep-together='always'>
     <fo:block>this will be line 1</fo:block>
     <fo:block>this will be line 2</fo:block>
</fo:block>
```

To keep a block together and prevent it being split by a page break use the keep-together attribute.

To keep a block with the block following it use the keep-with-next attribute.

To keep a block with the block before it use the keep-with-previous attribute.

To format a block of text retaining line-feeds which were in the XML, use the linefeed-treatment attribute.

To change the color of text use the color attribute.

To align a paragraph to the left, right or both margins use the text-align and text-align-last attributes.

An fo:block may contain a fo:retrieve-marker only if the fo:block is inside an fo:static-content element.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

fo:bidi-override (zero or more)

fo:character (zero or more)

fo:external-graphic (zero or more)

fo:instream-foreign-object (zero or more)

fo:inline (zero or more)

fo:inline-container (zero or more)

fo:leader (zero or more)

fo:page-number (zero or more)

fo:page-number-citation (zero or more)

fo:retrieve-marker (zero or more)

fo:marker (zero or more)

Parent element(s)

fo:float

fo:flow

fo:static-content

fo:block

fo:block-container

fo:table-cell

fo:list-item-label

fo:list-item-body

fo:marker

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom break-after break-before color end-indent font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight id intrusion-displace keep-together keep-with-next keep-with-previous last-line-end-indent left linefeed-treatment line-height line-height-shift-adjustment margin margin-bottom margin-left margin-right margin-top orphans padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right space-after space-before span start-indent text-align text-align-last text-altitude text-depth text-indent top visibility white-space-collapse white-space-treatment widows wrap-option

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.2. fo:block-container

Description

This element is used to create an area (a "reference area" in the specifications terms) which has a different writing direction or rotation. If you want to acheive other ends such as keeping two blocks together use an fo:block as the container.

If you do use reference-orientation to rotate the content to be vertical on the page then you need to specify inline-progression-dimension to limit the vertical height of the content.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

fo:float

fo:flow

fo:static-content

fo:block

fo:block-container

fo:table-cell

fo:list-item-label

fo:list-item-body

fo:marker

Attributes

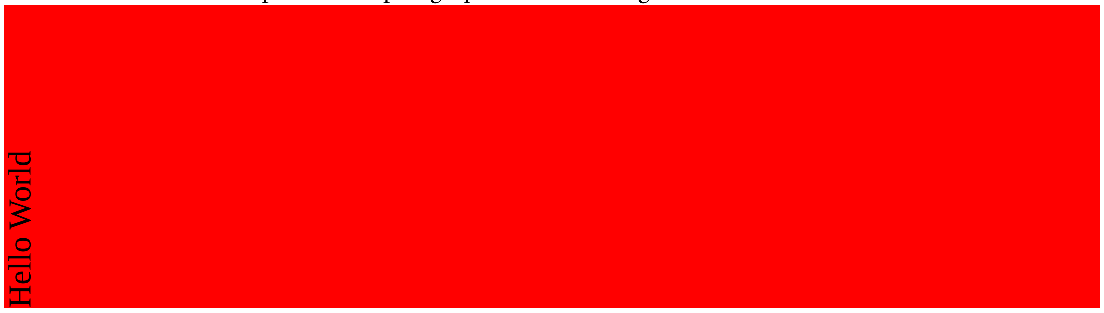
absolute-position background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat block-progression-dimension border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom break-after break-before clip display-align end-indent height id inline-progression-dimension intrusion-displace keep-together keep-with-next keep-with-previous left margin margin-bottom margin-left margin-right margin-top overflow padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top reference-orientation right space-after space-before span start-indent top width

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block-container reference-orientation='90'
               inline-progression-dimension='4cm'
               background-color='red'>
               <fo:block>Hello World</fo:block>
         </fo:block-container>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

This fo:block-container produces a paragraph rotated 90 degrees counter-clockwise like this:



C.3. fo:character

Description

This element is used to insert a single character into the content. Given that modern XML editors can insert all Unicode characters there is little requirement to use this element.

Parent element(s)

fo:block

fo:marker

Attributes

alignment-adjust alignment-baseline background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom character color dominant-baseline font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight glyph-orientation-horizontal glyph-orientation-vertical id keep-with-next keep-with-previous left letter-spacing line-height margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right score-spaces suppress-at-line-break text-altitude text-decoration text-depth text-shadow text-transform top treat-as-word-space visibility word-spacing

C.4. fo:conditional-page-master-reference

Description

This element associates a page master and a condition such that the page master will be used when the condition is true.

The conditions which are associated with this element are page-position, odd-or-even, and blank-or-not-blank. Each condition on each fo:conditional-page-master-reference in a fo:repeatable-page-master-alternatives element is evaluated in turn until one is found which is true, and that fo:conditional-page-master-reference is used.

Parent element(s)

fo:repeatable-page-master-alternatives

Attributes

blank-or-not-blank master-reference odd-or-even page-position

Example

```
<fo:page-sequence-master master-name='chapter'>
 <fo:repeatable-page-master-alternatives>
   <fo:conditional-page-master-reference
       page-position="first"
       master-reference='chapter-odd-no-header'/>

   <fo:conditional-page-master-reference
```

```
              odd-or-even='odd'
              master-reference='chapter-odd'/>

         <fo:conditional-page-master-reference
              odd-or-even='even'
              master-reference='chapter-even'/>
         </fo:repeatable-page-master-alternatives>
      </fo:page-sequence-master>
```

C.5. fo:declarations

Description

The fo:declarations formatting object is used to group global declarations for a stylesheet.

Child element(s)

fo:color-profile

Parent element(s)

fo:root

C.6. fo:external-graphic

Description

This element is used to include an image into the document.

This is an inline element so it must be contained in an fo:block element.

The image source is defined by the src attribute.

The src attribute is called a *uri-specification* and must follow the following rules:

> A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or both be absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

This means the following are all valid values for the src attribute:

> uri(ibex.jpg)
>
> uri("ibex.jpg")
>
> uri('ibex.jpg')
>
> url(http://www.xmlpdf.com/images/download2.gif)

To set the size of the image use the content-height and content-width attributes.

Parent element(s)

fo:block

fo:marker

Attributes

alignment-adjust alignment-baseline background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat baseline-shift block-progression-dimension border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom clip content-height content-type content-width display-align dominant-baseline end-indent height id inline-progression-dimension keep-with-next keep-with-previous left line-height margin margin-bottom margin-left margin-right margin-top overflow padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right scaling scaling-method space-after space-before src start-indent text-align top width

Example

If the src attribute is a URL an image which exists on a web server can be downloaded automatically by Ibex as the PDF file is created. The following XML will fetch the file download2.gif from www.xmlpdf.com as the document is created:

```
<fo:block space-before="6pt">
  <fo:external-graphic border="1pt solid black"
     src="url(http://www.xmlpdf.com/images/download2.gif)"
        content-width="200%" content-height="200%"/>
   </fo:block>
```

C.7. fo:float

Description

This element is used to position content either (a) at the top of a page or (b) to the side of a page so that text flows around it.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

Attributes

float

Example

```
<fo:block>
  <fo:float float="start">
   <fo:block-container inline-progression-dimension="3cm" padding="5mm">
      <fo:block padding="2mm" space-before.conditionality="retain" border="1pt
solid white" width="2cm">
         <fo:block padding-left="2mm">
          <fo:external-graphic src="url(ibexorange.jpg)" content-width='50%'/>
         </fo:block>
     </fo:block>
   </fo:block-container>
 </fo:float>
 <fo:block padding-top="2mm" padding-bottom='2mm' space-before="9pt" font="10pt
 'minion regular'">
 This text should appear to the right of the image until we pass the bottom of
the image
 and then appear below the image as well.
 </fo:block>
 <fo:block font="10pt 'minion regular'">
 We have lots of text here just to show that it will be formatted in the
correct way and eventually
 there will be enough text to go past the image and appear below it on the
page.  Then we will have some
 XML which shows how to acheive this effect.
 We have lots of text here just to show that it will be formatted in the
correct way and eventually
 there will be enough text to go past the image and appear below it on the
page.  Then we will have some
 XML which shows how to acheive this effect.
 We have lots of text here just to show that it will be formatted in the
correct way and eventually
 there will be enough text to go past the image and appear below it on the
page.  Then we will have some
 XML which shows how to acheive this effect.

 </fo:block>

</fo:block>
```

C.8. fo:flow

Description

This element contains block-level objects which create content which will appear in the body region of the page.

The flow-name attribute must correspond to a region-name used on the body region of the current page master. Which page master this is, is determined by the master-reference attribute of the containing fo:page-sequence. If the flow-name does not match the region-name the content will not appear.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

fo:page-sequence

Attributes

flow-name

C.9. fo:inline

Description

This element is used to format some text in a way which is different to the containing fo:block such as giving it a different font.

Parent element(s)

fo:block

fo:marker

Attributes

alignment-adjust alignment-baseline background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat baseline-shift block-progression-dimension border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom color dominant-baseline font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight height id inline-progression-dimension keep-together keep-with-next keep-with-previous left letter-spacing line-height margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right text-decoration top visibility width wrap-option

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>
         Hello
         <fo:inline font-size='40pt'>
         World
         </fo:inline
         </fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.10. fo:instream-foreign-object

Description

This element is used to place an object which is contained in the XML into the PDF document. The only supported object type is an SVG image.

An example of include an inline SVG image is:

```
<fo:instream-foreign-object width="20%" height="1cm">
    <svg xmlns:fo="http://www.w3.org/TR/xsl/Format"
                xmlns="http://www.w3.org/2000/svg">
         <path style="stroke-width:1;fill:rgb(246,127,0);"
              d="M204.33 139.83 C196.33 133.33 z" />
    </svg>
</fo:instream-foreign-object>
```

Not all implementations of Ibex support SVG images.

Parent element(s)

fo:block

fo:marker

Attributes

alignment-adjust alignment-baseline background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat baseline-shift block-progression-dimension border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom clip content-height content-type content-width display-align dominant-baseline end-indent height id inline-progression-dimension keep-with-next keep-with-previous left line-height margin margin-bottom margin-left margin-right margin-top overflow padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right scaling scaling-method space-after space-before start-indent text-align top width

C.11. fo:layout-master-set

Description

This element contains all the page master elements (fo:simple-page-master, fo:page-sequence-master) used to create individual pages or sequence of pages.

At least one child element must exist or the document will contain no pages.

Child element(s)

fo:simple-page-master (zero or more)

fo:page-sequence-master (zero or more)

Parent element(s)

fo:root

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.12. fo:leader

Description

This element is used to draw a horizontal line across the page.

A simple line is drawn like this:

```
<fo:block>
  <fo:leader leader-pattern='rule' rule-thickness="0.2pt"/>
</fo:block>
```

The leader can also be drawn between other pieces of text on the same line and can be set to expand to fill available space like this:

```
<fo:block text-align="justify" text-align-last='justify'>
 This is before the leader
  <fo:leader leader-pattern='rule' rule-thickness="0.2pt"/>
 this is after the leader
</fo:block>
```

producing the effect below. Note the use of text-align-last which is required to justify the single line paragraph.

```
This is before the leader ________________________________ this is after the leader
```

Setting the leader-pattern attribute to 'dots' changes the line into dots like this:

```
This is before the leader  . . . . . . . . . . . . . . . . . . . .  this is after the leader
```

Setting the leader-pattern attribute to 'space' changes the line into spaces like this:

```
This is before the leader                                    this is after the leader
```

Parent element(s)

fo:block

fo:marker

C.13. fo:list-block

Description

This element is used to create a list, which is similar to a two column table.

A simple list looks like this:

```
 <fo:list-block provisional-distance-between-starts=".5cm"
     provisional-label-separation="0.1cm">
    <fo:list-item>
       <fo:list-item-label end-indent="label-end()">
          <fo:block font='8pt arial'>&#x25CF;</fo:block>
       </fo:list-item-label>
       <fo:list-item-body start-indent="body-start()">
            <fo:block>
                item one
            </fo:block>
        </fo:list-item-body>
   </fo:list-item>
    <fo:list-item>
       <fo:list-item-label end-indent="label-end()">
          <fo:block font='8pt arial'>&#x25CF;</fo:block>
       </fo:list-item-label>
       <fo:list-item-body start-indent="body-start()">
            <fo:block>
                item two
            </fo:block>
        </fo:list-item-body>
   </fo:list-item>
</fo:list-block>
```

producing the following content:

- item one
- item two

The list is rendered as two columns. The first column is called the label, the second is called the body.

The distance from the start of the label column to the start of the body column is set by the provisional-distance-between-starts attribute. The gap between the columns is set by the provisional-label-separation attribute. The width of the label column is therefore:

```
provisional-distance-between-starts
  - provisional-label-separation
```

Each item in the list is contained in a fo:list-item element. The fo:list-item contains exactly one fo:list-item-label and fo:list-item-body element, with the fo:list-item-label coming first.

The fo:list-item-label should always have its end-indent attribute set to "label-end()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the end-indent is not so specified the label column will overlap the body column.

The fo:list-item-body should always have its start-indent attribute set to "body-start()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the start-indent is not so specified the label column will overlap the body column.

Child element(s)

fo:list-item

Parent element(s)

fo:float

fo:flow

fo:static-content

fo:block

fo:block-container

fo:table-cell

fo:list-item-label

fo:list-item-body

fo:marker

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom break-after break-before end-indent id intrusion-displace keep-together keep-with-next keep-with-previous left margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top provisional-distance-between-starts provisional-label-separation relative-position right space-after space-before start-indent top

C.14. fo:list-item

Description

This element contains the label and body of an entry in a list.

The height of the fo:list-item will be the taller of the label and body items it contains.

Child element(s)

fo:list-item-label

fo:list-item-body

Parent element(s)

fo:list-block

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom break-after break-before end-indent id intrusion-displace keep-together keep-with-next keep-with-previous left margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-align relative-position right space-after space-before start-indent top

Example

```
<fo:list-block provisional-distance-between-starts=".5cm"
    provisional-label-separation="0.1cm">
   <fo:list-item>
      <fo:list-item-label end-indent="label-end()">
         <fo:block font='8pt arial'>&#x25CF;</fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">
           <fo:block>
               item one
           </fo:block>
       </fo:list-item-body>
  </fo:list-item>
   <fo:list-item>
      <fo:list-item-label end-indent="label-end()">
         <fo:block font='8pt arial'>&#x25CF;</fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">
           <fo:block>
               item two
           </fo:block>
       </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```

C.15. fo:list-item-body

Description

This element contains the body part of a list item.

The fo:list-item-body contains block-level elements, it does not itself contain text.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

fo:list-item

Attributes

id keep-together

Example

```
       <fo:list-block provisional-distance-between-starts=".5cm"
           provisional-label-separation="0.1cm">
          <fo:list-item>
             <fo:list-item-label end-indent="label-end()">
                <fo:block font='8pt arial'>&#x25CF;</fo:block>
             </fo:list-item-label>
             <fo:list-item-body start-indent="body-start()">
                  <fo:block>
                      item one
                  </fo:block>
              </fo:list-item-body>
         </fo:list-item>
          <fo:list-item>
             <fo:list-item-label end-indent="label-end()">
                <fo:block font='8pt arial'>&#x25CF;</fo:block>
             </fo:list-item-label>
             <fo:list-item-body start-indent="body-start()">
                  <fo:block>
                      item two
                  </fo:block>
              </fo:list-item-body>
         </fo:list-item>
      </fo:list-block>
```

C.16. fo:list-item-label

Description

This element contains the label part of a list item.

The fo:list-item-label contains block-level elements, it does not itself contain text.

The fo:list-item-label should always have its end-indent attribute set to "label-end()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the end-indent is not so specified the label column will overlap the body column.

The fo:list-item-body should always have its start-indent attribute set to "body-start()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the start-indent is not so specified the label column will overlap the body column.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

fo:list-item

Attributes

id keep-together

Example

```
    <fo:list-block provisional-distance-between-starts=".5cm"
        provisional-label-separation="0.1cm">
       <fo:list-item>
          <fo:list-item-label end-indent="label-end()">
             <fo:block font='8pt arial'>&#x25CF;</fo:block>
          </fo:list-item-label>
          <fo:list-item-body start-indent="body-start()">
               <fo:block>
                   item one
               </fo:block>
           </fo:list-item-body>
      </fo:list-item>
       <fo:list-item>
          <fo:list-item-label end-indent="label-end()">
             <fo:block font='8pt arial'>&#x25CF;</fo:block>
          </fo:list-item-label>
          <fo:list-item-body start-indent="body-start()">
               <fo:block>
                   item two
               </fo:block>
           </fo:list-item-body>
      </fo:list-item>
   </fo:list-block>
```

C.17. fo:marker

Description

This element contains some content which will be retrieved elsewhere in the document using a fo:retrieve-marker element.

Typically fo:marker is used to set some piece of text such as the current chapter title which is then retrieved within an fo:static-content element for placing in the page header. The Ibex manual uses this technique to place the current chapter name in the top right corner of most pages.

An fo:marker cannot be used in fo:static-content elements, and an fo:retrieve-maker can be used only in fo:static-content elements.

An fo:marker uses the marker-class-name attribute to group markers which have a common purpose. The fo:retrieve-marker element has some attributes to specify which marker should be retrieved, such as the first or last one in the document or the first or last one on that page.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

fo:bidi-override (zero or more)

fo:character (zero or more)

fo:external-graphic (zero or more)

fo:instream-foreign-object (zero or more)

fo:inline (zero or more)

fo:inline-container (zero or more)

fo:leader (zero or more)

fo:page-number (zero or more)

fo:page-number-citation (zero or more)

Parent element(s)

fo:block

Attributes

marker-class-name

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple">
     <fo:region-body margin="2.5cm" region-name="body"/>
     <fo:region-before extent="2cm" region-name="header"/>
   </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="simple">

  <fo:static-content flow-name='header'>
     <fo:block>the retrieved marker is [
      <fo:retrieve-marker retrieve-class-name='subject'/> ]
     </fo:block>
  </fo:static-content>

  <fo:flow flow-name="body">
    <fo:block>
       <fo:marker marker-class-name='subject'>
         Page Numbering</fo:marker>
          this is page number <fo:page-number/>
      </fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.18. fo:page-number

Description

This element is used to insert the current page number into the document.

The page-number string is formatted using the string conversion properties of the containing fo:page-sequence, namely format, grouping-separator, grouping-size, letter-value, country and language.

Parent element(s)

fo:block

fo:marker

Attributes

alignment-adjust alignment-baseline background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat baseline-shift border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom dominant-baseline font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight id keep-with-next keep-with-previous left letter-spacing line-height margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right score-spaces text-altitude text-decoration text-depth text-shadow top visibility word-spacing wrap-option

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>
             this is page number <fo:page-number/>
         </fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

This example produces a paragraph containing the page number like this:

this is page number 113

C.19. fo:page-number-citation

Description

This element is used to insert the page number on which the content created by some other element occurs.

The page-number string is formatted using the string conversion properties of the containing fo:page-sequence, namely format, grouping-separator, grouping-size, letter-value, country and language.

The fo:page-number-citation has a ref-id attribute which should match the id attribute of the element whose page number we want to appear.

Parent element(s)

fo:block

fo:marker

Attributes

alignment-adjust alignment-baseline background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat baseline-shift border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom dominant-baseline font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight id keep-with-next keep-with-previous left letter-spacing line-height margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top ref-id relative-position right score-spaces text-altitude text-decoration text-depth text-shadow text-transform top visibility word-spacing wrap-option

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"/>
      </fo:simple-page-master>
   </fo:layout-master-set>

  <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block id='22'>
         Hello
         </fo:block>
      </fo:flow>
  </fo:page-sequence>

  <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>
           The block with id='22' is on page
           <fo:page-number-citation ref-id='22'/>
         </fo:block>
      </fo:flow>
  </fo:page-sequence>

</fo:root>
```

C.20. fo:page-sequence

Description

This element contains content for one or more pages. The content is contained in fo:static-content elements which hold content for the page header, footer and other outer regions, and a single fo:flow element which contains content to be placed in the body region of the page.

The page-sequence has a master-reference attribute which should correspond to the master-name of an element contained within the documents fo:layout-master-set, such as a fo:single-page-master. It is this page master which determines how many pages can be created from the content.

The page number for the first page created by this page sequence can be set using the initial-page-number attribute. The format of the page number is controlled using the format attribute.

Child element(s)

fo:static-content (zero or more)

fo:flow (exactly one)

Parent element(s)

fo:root

Attributes

format initial-page-number master-reference

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.21. fo:page-sequence-master

Description

This element is used to define the sequence in which one or more page master elements (fo:simple-page-master, fo:repeatable-page-master) are used to create pages.

The element describes a sequence of page layouts and has a master-name which uniquely identifies it. This master-name is used as the master-reference on a fo:page-sequence element in order to create pages using the sequence described by this fo:page-sequence-master.

Each child of this element specifies a sequence of one or more pages:

A fo:single-page-master-reference element is used define the layout for one page.

A fo:repeatable-page-master-reference element is used define multiple pages which have the same layout because they use the same page master.

A fo:repeatable-page-master-alternatives element is used define multiple pages which can have different layouts created using different page master elements.

Child element(s)

fo:single-page-master-reference (zero or more)

fo:repeatable-page-master-reference (zero or more)

fo:repeatable-page-master-alternatives (zero or more)

Parent element(s)

fo:layout-master-set

Attributes

master-name

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
      <fo:page-sequence-master master-name='repeated'>
          <fo:repeatable-page-master-reference
             master-reference='simple'/>
      </fo:page-sequence-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="repeated">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.22. fo:region-after

Description

This element defines the shape of a region which is at the bottom of an unrotated page. Content from fo:static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-after" which is usually changed to something simpler such as 'footer' using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the fo:region-body element the fo:region-after does not have margin properties. The size of the region is defined using the extent attribute.

By default the before region is reduced in width by the presence of the fo:region-start and fo:region-end elements. This can be changed by setting the precedence attribute to "true".

Parent element(s)

fo:simple-page-master

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width clip display-align extent padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top precedence reference-orientation region-name writing-mode

Example

```
<fo:simple-page-master master-name="front-page" margin='1.5cm'
     page-height="297mm" page-width="210mm">
  <fo:region-body region-name="body"
        margin='0.75cm 0.5cm 0.75cm 3cm'/>
  <fo:region-before region-name="header" extent="2.5cm"/>
  <fo:region-after region-name="footer" extent="1cm"/>
  <fo:region-start extent='1cm' background-color='#eeeeee'/>
  <fo:region-end extent='1cm' background-color='#eeeeee'/>
</fo:simple-page-master>
```

C.23. fo:region-before

Description

This element defines the shape of a region which is at the top of an unrotated page. Content from fo:static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-before" which is usually changed to something simpler such as 'header' using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the fo:region-body element the fo:region-before does not have margin properties. The size of the region is defined using the extent attribute.

By default the before region is reduced in width by the presence of the fo:region-start and fo:region-end elements. This can be changed by setting the precedence attribute to "true".

Parent element(s)

fo:simple-page-master

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width clip display-align extent padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top precedence reference-orientation region-name writing-mode

Example

```
<fo:simple-page-master master-name="front-page" margin='1.5cm'
     page-height="297mm" page-width="210mm">
  <fo:region-body region-name="body"
        margin='0.75cm 0.5cm 0.75cm 3cm'/>
  <fo:region-before region-name="header" extent="2.5cm"/>
  <fo:region-after region-name="footer" extent="1cm"/>
  <fo:region-start extent='1cm' background-color='#eeeeee'/>
  <fo:region-end extent='1cm' background-color='#eeeeee'/>
</fo:simple-page-master>
```

C.24. fo:region-body

Description

This element defines the shape of the main area on the page into which content from fo:flow elements will be placed.

The region has a default name of "xsl-region-body" which is usually changed to something simpler using the region-name attribute.

A page can be defined which has multiple columns by using the column-count and column-gap attributes on this region.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Parent element(s)

fo:simple-page-master

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width clip column-count column-gap display-align end-indent margin margin-bottom margin-left margin-right margin-top overflow padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top reference-orientation region-name space-after space-before start-indent writing-mode

Example

```
<fo:simple-page-master master-name="front-page" margin='1.5cm'
     page-height="297mm" page-width="210mm">
  <fo:region-body region-name="body"
         margin='0.75cm 0.5cm 0.75cm 3cm'/>
  <fo:region-before region-name="header" extent="2.5cm"/>
  <fo:region-after region-name="footer" extent="1cm"/>
  <fo:region-start extent='1cm' background-color='#eeeeee'/>
  <fo:region-end extent='1cm' background-color='#eeeeee'/>
</fo:simple-page-master>
```

C.25. fo:region-end

Description

This element defines the shape of a region which is at the right of an unrotated page. Content from fo:static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-start" which is usually changed to something simpler such as 'right' using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the fo:region-body element the fo:region-end does not have margin properties. The size of the region is defined using the extent attribute.

Parent element(s)

fo:simple-page-master

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width clip display-align extent padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top reference-orientation region-name writing-mode

Example

```
<fo:simple-page-master master-name="front-page" margin='1.5cm'
     page-height="297mm" page-width="210mm">
  <fo:region-body region-name="body"
         margin='0.75cm 0.5cm 0.75cm 3cm'/>
  <fo:region-before region-name="header" extent="2.5cm"/>
  <fo:region-after region-name="footer" extent="1cm"/>
  <fo:region-start extent='1cm' background-color='#eeeeee'/>
  <fo:region-end extent='1cm' background-color='#eeeeee'/>
</fo:simple-page-master>
```

C.26. fo:region-start

Description

This element defines the shape of a region which is at the left of an unrotated page. Content from fo:static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-start" which is usually changed to something simpler such as 'left' using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the fo:region-body element the fo:region-start does not have margin properties. The size of the region is defined using the extent attribute.

Parent element(s)

fo:simple-page-master

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width clip display-align extent padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top reference-orientation region-name writing-mode

Example

```
<fo:simple-page-master master-name="front-page" margin='1.5cm'
     page-height="297mm" page-width="210mm">
  <fo:region-body region-name="body"
         margin='0.75cm 0.5cm 0.75cm 3cm'/>
  <fo:region-before region-name="header" extent="2.5cm"/>
  <fo:region-after region-name="footer" extent="1cm"/>
  <fo:region-start extent='1cm' background-color='#eeeeee'/>
  <fo:region-end extent='1cm' background-color='#eeeeee'/>
</fo:simple-page-master>
```

C.27. fo:repeatable-page-master-alternatives

Description

This element contains a set of fo:conditional-page-master-reference elements, each of which specifies a page master and some conditional information.

When the rendering of content from a fo:flow element triggers the creation of a new page each fo:conditional-page-master-reference contained in this element is evaluated to see if it should be used.

Typically the fo:conditional-page-master-reference elements are used to specify different page layouts for the first page of a sequence or for odd and even pages. The Ibex manual uses this approach, so that the first page of each chapter has no header.

Child element(s)

fo:conditional-page-master-reference (zero or more)

Parent element(s)

fo:page-sequence-master

Example

```
<fo:page-sequence-master master-name='chapter'>
 <fo:repeatable-page-master-alternatives>
   <fo:conditional-page-master-reference
       page-position="first"
       master-reference='chapter-odd-no-header'/>

   <fo:conditional-page-master-reference
        odd-or-even='odd'
        master-reference='chapter-odd'/>

   <fo:conditional-page-master-reference
        odd-or-even='even'
        master-reference='chapter-even'/>
   </fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
```

C.28. fo:repeatable-page-master-reference

Description

This element specifies that the fo:simple-page-master which has a master-name corresponding to the master-reference of this element should be used to define the layout for a one or more pages.

The difference between this and a fo:single-page-master-reference is that the single-page-master-reference produces one page whereas this element can produce multiple pages. The maxmium number of pages created by this element is controlled by the maximum-repeats attribute which by default is unlimited.

Parent element(s)

fo:page-sequence-master

Attributes

master-reference maximum-repeats

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
      <fo:page-sequence-master master-name='repeated'>
          <fo:repeatable-page-master-reference
             master-reference='simple'/>
      </fo:page-sequence-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="repeated">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.29. fo:retrieve-marker

Description

The fo:marker element contains some content which will be retrieved elsewhere in the document using a fo:retrieve-marker element.

Typically fo:marker is used to set some piece of text such as the current chapter title which is then retrieved within an fo:static-content element for placing in the page header. The Ibex manual uses this technique to place the current chapter name in the top right corner of most pages.

The fo:marker element cannot be used in fo:static-content elements and the fo:retrieve-maker element can be used only in fo:static-content elements.

An fo:marker uses the marker-class-name attribute to group markers which have a common purpose. The fo:retrieve-marker element has some attributes to specify which marker should be retrieved, such as the first or last one in the document or the first or last one on that page.

For the fo:retrieve-marker element to work its retrieve-class-name attribute must have the same value as the maker-class-name attribute used on some fo:marker element.

Parent element(s)

fo:block

Attributes

retrieve-boundary retrieve-class-name retrieve-position

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple">
     <fo:region-body margin="2.5cm" region-name="body"/>
     <fo:region-before extent="2cm" region-name="header"/>
   </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="simple">

  <fo:static-content flow-name='header'>
     <fo:block>the retrieved marker is [
      <fo:retrieve-marker retrieve-class-name='subject'/> ]
     </fo:block>
  </fo:static-content>

  <fo:flow flow-name="body">
    <fo:block>
       <fo:marker marker-class-name='subject'>
         Page Numbering</fo:marker>
          this is page number <fo:page-number/>
      </fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.30. fo:root

Description

This is the top level element in the formatting objects XML and contains the fo:layout-master-set, an optional fo:declarations and one or more fo:page-sequence elements. These child elements must be in the order listed.

Child element(s)

fo:layout-master-set (exactly one)

fo:declarations (zero or one)

fo:page-sequence (one or more)

Attributes

media-usage

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.31. fo:simple-page-master

Description

This element defines the layout of a single page. It is uniquely identified by its master-name which is used on fo:page-sequence and other elements to create pages which use this layout.

The content of the page goes into the named regions which are specified by the child elements of this element.

The size of the page is defined using the page-height and page-width attributes. The default page size is A4.

Child element(s)

fo:region-body (exactly one)

fo:region-before (zero or one)

fo:region-after (zero or one)

fo:region-start (zero or one)

fo:region-end (zero or one)

Parent element(s)

fo:layout-master-set

Attributes

end-indent margin margin-bottom margin-left margin-right margin-top master-name page-height page-width reference-orientation space-after space-before start-indent writing-mode

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.32. fo:single-page-master-reference

Description

This element specifies that the fo:simple-page-master which has a master-name corresponding to the master-reference of this element should be used to define the layout for a single page.

Parent element(s)

fo:page-sequence-master

Attributes

master-reference

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.33. fo:static-content

Description

This element is used to create content in a region other then the body region. The term 'static' refers to the fact that the content will go only on the current page, unlike the content of a fo:flow element which may extend to many pages.

Static content is commonly used for page headers and footers. The content is usually different on each page as the page number changes.

The flow-name attribute may correspond to a region-name used on a non-body region of the current page master. Which page master this is is determined by the master-reference attribute of the containing fo:page-sequence. If the flow-name does not match a region-name the content will not appear. This makes it possible to have a fo:page-sequence which contains many static content elements each matching a different page layout. Only the static content which matches a region which is on the current page layout will be displayed.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

fo:page-sequence

Attributes

flow-name

Example

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"/>
         <fo:region-before extent="2cm" region-name="header"/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:static-content flow-name='header'>
         <fo:block>this is the header</fo:block>
      </fo:static-content>

      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

C.34. fo:table

Description

This element creates a table. Tables have rows and columns and possibly also headers and footers.

The size of table columns can either be calculated from the content of cells, or specified using fo:table-column elements. Using fo:table-column elements results in consistent output regardless of cell contents.

The width and other characteristics of columns are defined using fo:table-column elements. An optional table header, which by default is repeated after each page break, is specified using the fo:table-header element. An optional table footer, which by default is repeated before each page break, is specified using the fo:table-footer element.

Table rows are contained in one or more fo:table-body elements.

Table borders are controlled using the border-collapse attribute. If this has a value of "collapse" then table and cell borders are collapsed into a single border. If the value is "separate" then table, row and cell borders are all drawn separately, one inside the other.

The default value for border-collapse is "collapse". To create the kind of borders used in CSS where the cell borders appears inside the row and table borders set border-collapse to "separate".

Child element(s)

fo:table-column (zero or more)

fo:table-header (zero or one)

fo:table-footer (zero or one)

fo:table-body (one or more)

Parent element(s)

fo:float

fo:flow

fo:static-content

fo:table-and-caption

fo:block

fo:block-container

fo:table-cell

fo:list-item-label

fo:list-item-body

fo:marker

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat block-progression-dimension border border-after border-after-color border-after-precedence border-after-style border-after-width border-before border-before-color border-before-precedence border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-precedence border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-precedence border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom break-after break-before end-indent font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight height id inline-progression-dimension intrusion-displace keep-together keep-with-next keep-with-previous left margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-position right space-after space-before start-indent table-layout table-omit-footer-at-break table-omit-header-at-break top width writing-mode

C.35. fo:table-and-caption

Description

This element is used to create a table which has a caption above or below it, and to keep the table and caption together.

By default the caption appears above the table. Set caption-side="bottom" to make the caption appear below the table.

Child element(s)

fo:table-caption (zero or one)

fo:table (zero or more)

Parent element(s)

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-style border-after-width border-before border-before-color border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-style border-start-width border-top border-top-color border-top-style border-top-width break-after break-before caption-side end-indent id margin margin-bottom margin-left margin-right margin-top padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top space-after space-before start-indent

C.36. fo:table-body

Description

This element is a container for fo:table-row and fo:table-cell elements. A single fo:table element can contain multiple fo:table-body elements which are output in the order in which they appear in the XML.

Child element(s)

fo:table-cell (zero or more)

fo:table-row (zero or more)

Parent element(s)

fo:table

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat block-progression-dimension border border border-after border-after border-after-color border-after-color border-after-precedence border-after-style border-after-style border-after-width border-after-width border-before border-before border-before-color border-before-color border-before-precedence border-before-style border-before-style border-before-width border-before-width border-bottom border-bottom border-bottom-color border-bottom-color border-bottom-style border-bottom-style border-bottom-width border-bottom-width border-end border-end border-end-color border-end-color border-end-precedence border-end-style border-end-style border-end-width border-end-width border-left border-left border-left-color border-left-color border-left-style border-left-style border-left-width border-left-width border-right border-right border-right-color border-right-color border-right-style border-right-style border-right-width border-right-width border-start border-start border-start-color border-start-color border-start-precedence border-start-style border-start-style border-start-width border-start-width border-top border-top border-top-color border-top-color border-top-style border-top-style border-top-width border-top-width bottom id left relative-position right top visibility

Notes on attributes

As described in section 6.7.8 of the XSL-FO specification, only the background properties from this set apply. If the value of border-collapse on the table is "collapse" or "collapse-with-precedence" the border properties also apply.

C.37. fo:table-cell

Description

This element is a container for content in a cell within a table. Cell content is contained in block-level elements within the cell. A common error is to place text directly within the fo:table-cell element, which results in the text being discarded.

A fo:table-cell element can contain any number of block level elements.

Contents of a cell are aligned vertically using the display-align attribute.

To have a cell span mutiple columns use the number-columns-spanned attribute. To span multiple rows use the number-rows-spanned attribute.

Child element(s)

fo:block (zero or more)

fo:block-container (zero or more)

fo:list-block (zero or more)

fo:table (zero or more)

Parent element(s)

fo:table-header

fo:table-row

fo:table-footer

fo:table-body

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat block-progression-dimension border border-after border-after-color border-after-precedence border-after-style border-after-width border-before border-before-color border-before-precedence border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-precedence border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-precedence border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom column-number display-align empty-cells ends-row height id inline-progression-dimension left number-columns-spanned number-rows-spanned padding padding-after padding-before padding-bottom padding-end padding-left padding-right padding-start padding-top relative-align relative-position right starts-row top width

C.38. fo:table-column

Description

This element is used to specify characteristics for columns in a table such as the background color and the width.

A table would typically have multiple fo:table-column elements looking something like this:

```
<fo:table>
    <fo:table-column column-width='20%'/>
    <fo:table-column column-width='30%'/>
    <fo:table-column column-width='50%'/>
    <fo:table-body>
         <fo:table-row>
              <fo:table-cell>col 1</fo:table-cell>
              <fo:table-cell>col 2</fo:table-cell>
              <fo:table-cell>col 3</fo:table-cell>
         </fo:table-row>
    </fo:table-body>
</fo:table>
```

This defines a table with three columns. Implicitly the three fo:table-column elements specify the width of columns one, two and three in that order. This can be made explict using the column-number attribute like this:

```
<fo:table>
    <fo:table-column column-number='1'
      column-width='20%'/>
    <fo:table-column column-number='2'
      column-width='30%'/>
    <fo:table-column column-number='3'
      column-width='50%'/>
    <fo:table-body>
         <fo:table-row>
              <fo:table-cell>col 1</fo:table-cell>
              <fo:table-cell>col 2</fo:table-cell>
              <fo:table-cell>col 3</fo:table-cell>
         </fo:table-row>
    </fo:table-body>
</fo:table>
```

A single fo:table-column can be used to set the width and other characteristics of multiple columns by using the columns-spanned attribute. In the example below the first fo:table-column sets the width of the first two columns to 20% and the third column to 50%:

```
<fo:table>
    <fo:table-column columns-spanned='2'
         column-width='20%'/>
    <fo:table-column
        column-width='50%'/>
    <fo:table-body>
         <fo:table-row>
              <fo:table-cell>col 1</fo:table-cell>
              <fo:table-cell>col 2</fo:table-cell>
              <fo:table-cell>col 3</fo:table-cell>
         </fo:table-row>
    </fo:table-body>
</fo:table>
```

Percentage values used in the column-width attribute refer to the width of the table.

If fo:table-column elements are not used all columns will be of equal width.

Parent element(s)

fo:table

C.39. fo:table-footer

Description

This element creates a footer which appears once at the bottom of the table and and before each page break. To prevent this repetition set table-omit-footer-at-break to "true" on the containing table.

A fo:table-footer is itself a table and contains rows and cells in the same manner as fo:table element.

Child element(s)

fo:table-cell (zero or more)

fo:table-row (zero or more)

Parent element(s)

fo:table

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-precedence border-after-style border-after-width border-before border-before-color border-before-precedence border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-precedence border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-precedence border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom end-indent font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight id left margin margin-bottom margin-left margin-right margin-top relative-position right space-after space-before start-indent top visibility

Notes on attributes

As described in section 6.7.7 of the XSL-FO specification, only the background properties from this set apply. If the value of border-collapse on the table is "collapse" or "collapse-with-precedence" the border properties also apply.

C.40. fo:table-header

Description

This element creates a header which appears once at the top of the table and is then repeated after each page break. To prevent this repetition set table-omit-header-at-break to "true" on the containing table.

A fo:table-header is itself a table and contains rows and cells in the same manner as fo:table element.

Child element(s)

fo:table-cell (zero or more)

fo:table-row (zero or more)

Parent element(s)

fo:table

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat border border-after border-after-color border-after-precedence border-after-style border-after-width border-before border-before-color border-before-precedence border-before-style border-before-width border-bottom border-bottom-color border-bottom-style border-bottom-width border-end border-end-color border-end-precedence border-end-style border-end-width border-left border-left-color border-left-style border-left-width border-right border-right-color border-right-style border-right-width border-start border-start-color border-start-precedence border-start-style border-start-width border-top border-top-color border-top-style border-top-width bottom end-indent font-family font-selection-strategy font-size font-size-adjust font-stretch font-style font-variant font-weight id left margin margin-bottom margin-left margin-right margin-top relative-position right space-after space-before start-indent top visibility

Notes on attributes

As described in section 6.7.6 of the XSL-FO specification, only the background properties from this set apply. If the value of border-collapse on the table is "collapse" or "collapse-with-precedence" the border properties also apply.

C.41. fo:table-row

Description

This element acts as a container for fo:table-cell elements.

Table row elements are not required. A table-body element can contain table-cell elements directly using the starts-row and ends-row attributes on the cells to determine where rows start and end.

The height of a row is by default the height of the tallest cell in the row. This can be overridden using the height or block-progression-dimension attributes. Use block-progression-dimension.minimum to set a minimum height, block-progression-dimension.maximum to set a maximum height.

Rows cannot have padding. This is stated in section 6.7.9 of the XSL-FO specification.

Child element(s)

fo:table-cell (one or more)

Parent element(s)

fo:table-header

fo:table-footer

fo:table-body

Attributes

background-attachment background-color background-image background-position-horizontal background-position-vertical background-repeat block-progression-dimension border border border-after border-after border-after-color border-after-color border-after-precedence border-after-style border-after-style border-after-width border-after-width border-before border-before border-before-color border-before-color border-before-precedence border-before-style border-before-style border-before-width border-before-width border-bottom border-bottom border-bottom-color border-bottom-color border-bottom-style border-bottom-style border-bottom-width border-bottom-width border-end border-end border-end-color border-end-color border-end-precedence border-end-style border-end-style border-end-width border-end-width border-left border-left border-left-color border-left-color border-left-style border-left-style border-left-width border-left-width border-right border-right border-right-color border-right-color border-right-style border-right-style border-right-width border-right-width border-start border-start border-start-color border-start-color border-start-precedence border-start-style border-start-style border-start-width border-start-width border-top border-top border-top-color border-top-color border-top-style border-top-style border-top-width border-top-width bottom break-after break-before height id keep-together keep-with-next keep-with-previous left relative-position right top visibility

C.42. absolute-position

Default value

auto

Values

auto
absolute
fixed
inherit

C.43. background-color

Description

Sets the background color for the element.

Default value

transparent

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
transparent	
inherit	

C.44. block-progression-dimension

Description

Sets the dimension of content in the block progression direction, which for an unrotated page is down the page.

The content of an element excludes padding and borders. This means an element with block-progression-dimension='3cm' and border='.25cm' will have a height including borders and padding of 3.5cm.

Can be set as a single value such as:

```
block-progression-dimension='20cm'
```

or you can specify minimum and maximum values like this:

```
      block-progression-dimension.minimum='5cm'
      block-progression-dimension.maximum='25cm'
```

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
<percentage>	A percentage such as "10%". The value is calculated as a percentage of the parent elements height.
<length-range>	The value has three sub-components, namely minimim, optimum and maximum. Each of these can be set to a <length> value.
inherit	

C.45. border

Description

Sets the border for all four sides of an element to the same value.

Any of the values listed can be combined, for example you can have:

```
      border='12pt solid red'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.	
<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.

		this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge

inherit

C.46. border-after-color

Description

Sets the 'after' border color, which for an unrotated object is the bottom one. For example:

```
border-after-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

C.47. border-after-style

Description

Sets the 'after' border style, which for an unrotated object is the bottom one. For example:

```
border-after-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
inherit		

C.48. border-after-width

Description

Sets the 'after' border width, which for an unrotated object is the bottom one. For example:

```
border-after-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.

	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.49. border-before-color

Description

Sets the 'before' border color, which for an unrotated object is the top one. For example:

```
border-before-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.50. border-before-style

Description

Sets the 'before' border style, which for an unrotated object is the top one. For example:

```
border-before-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset

	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
inherit		

C.51. border-before-width

Description

Sets the 'before' border width, which for an unrotated object is the top one. For example:

```
border-before-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.52. border-bottom

Description

Sets the color, width and style of the bottom border of an element.

A shorthand way of setting border-bottom-color, border-bottom-width and border-bottom-style.

Any of the values listed can be combined, for example you can have:

```
border-bottom='12pt solid red'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.	
<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
inherit		

C.53. border-bottom-color

Description

Sets the bottom border color. For example:

```
border-bottom-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.54. border-bottom-style

Description

Sets the bottom border style. For example:

```
border-bottom-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
inherit		

C.55. border-bottom-width

Description

Sets the bottom border width. For example:

```
border-bottom-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.56. border-collapse

Description

Controls whether borders on adjacent rows, cells and table elements are collapsed into a single border or remain separate.

Default value

collapse

Values

collapse	borders are collapsed. Precedence rules are evaluated to see which borders take precedence.
collapse-with-precedence	borders are collapsed. Precedence rules are evaluated to see which borders take precedence. In addition the border-precedence attribute can be used to change the precedence rules.
separate	borders are not collapsed. Only cell and table borders are considered, borders on all other elements are ignored.
inherit	

C.57. border-color

Description

Sets the border color for all four sides of an element to the same color or to a number of different colors.

To set all borders to the same color use a single value like this:

```
border-color='red'
```

If there are two values the top and bottom borders are set to the first value and the side borders are set to the second, like this:

```
border-color='red blue'
```

If there are three values the top border is set to the first value, the side borders are set to the second, and the bottom is set to the third like this:

```
border-color='red blue green'
```

If there are four values the top border is set to the first value, the right border is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

```
border-color='red blue green black'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
transparent	
inherit	

C.58. border-end-color

Description

Sets the end border color (the right side of an unrotated page). For example:

```
border-end-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.59. border-end-style

Description

Sets the end border style (the right side of an unrotated page). For example:

```
border-end-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
inherit		

C.60. border-end-width

Description

Sets the end border width (the right side of an unrotated page). For example:

```
border-end-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.61. border-left

Description

Sets the color, width and style of the left border of an element.

A shorthand way of setting border-left-color, border-left-width and border-left-style.

Any of the values listed can be combined, for example you can have:

```
    border-left='12pt solid red'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.	
<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.

		this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
inherit		

C.62. border-left-color

Description

Sets the left border color. For example:

```
border-left-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.63. border-left-style

Description

Sets the left border style. For example:

```
border-left-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
inherit		

C.64. border-left-width

Description

Sets the left border width. For example:

```
border-left-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.

	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.65. border-right

Description

Sets the color, width and style of the right border of an element.

A shorthand way of setting border-right-color, border-right-width and border-right-style.

Any of the values listed can be combined, for example you can have:

```
border-right='12pt solid red'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.	
<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.

		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
inherit		

C.66. border-right-color

Description

Sets the right border color. For example:

```
border-right-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.67. border-right-style

Description

Sets the right border style. For example:

```
border-right-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example

		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
inherit		

C.68. border-right-width

Description

Sets the right border width. For example:

```
border-right-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.69. border-separation

Description

Sets the separation between cell borders in a table with border-collapse='separate'.

To set both horizontal and vertical separation the same use:

```
border-separation='3pt'
```

To set horizontal and vertical separation to different values use the inline-progression-dimension and block-progression-dimension components like this:

```
border-separation.inline-progression-dimension='3pt'
border-separation.block-progression-dimension='10pt'
```

For an unrotated page block-progression-dimension is down the page and inline-progression-dimension is across.

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.70. border-spacing

Description

This is a shorthand method of setting the border-separation attribute.

To set both horizontal and vertical separation the same use:

```
border-spacing='3pt'
```

To set horizontal and vertical separation to different values use two values separated by a space like this:

```
border-spacing='3mm 13mm'
```

The first value sets the horizontal spacing, the second sets the vertical spacing.

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.71. border-start-color

Description

Sets the start border color (the left side of an unrotated page). For example:

```
border-start-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.72. border-start-style

Description

Sets the start border style (the left side of an unrotated page). For example:

```
border-start-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
inherit		

C.73. border-start-width

Description

Sets the start border width (the left side of an unrotated page). For example:

```
border-start-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.74. border-style

Description

Sets the border style for all four sides of an element to the same style or to a number of different styles.

To set all borders to the same style use a single value like this:

```
border-style='solid'
```

If there are two values the top and bottom borders are set to the first value and the side borders are set to the second, like this:

```
border-style='solid none'
```

If there are three values the top border is set to the first value, the side borders are set to the second, and the bottom is set to the third like this:

```
border-style='solid none double'
```

If there are four values the top border is set to the first value, the right border is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

```
border-style='solid none double dotted'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
transparent		
inherit		

C.75. border-top

Description

Sets the color, width and style of the top border of an element.

A shorthand way of setting border-top-color, border-top-width and border-top-style.

Any of the values listed can be combined, for example you can have:

```
border-top='12pt solid red'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double
	dashed	See example this is 2pt black dashed
	dotted	See example this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter. this is 2pt blue inset
	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge
inherit		

C.76. border-top-color

Description

Sets the top border color. For example:

```
border-top-color='red'
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
inherit	

C.77. border-top-style

Description

Sets the top border style. For example:

```
border-top-style='solid'
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is .2pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
inherit		

C.78. border-top-width

Description

Sets the top border width. For example:

```
border-top-width='1pt'
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

C.79. border-width

Description

Sets the border width for all four sides of an element to the same width or to a number of different widths.

To set all borders to the same width use a single value like this:

```
border-width='1pt'
```

If there are two values the top and bottom borders are set to the first value and the side borders are set to the second, like this:

```
border-width='1pt 3pt'
```

If there are three values the top border is set to the first value, the side borders are set to the second, and the bottom is set to the third like this:

```
border-width='1pt 2pt 3pt'
```

If there are four values the top border is set to the first value, the right border is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

```
border-width='1pt 2pt 3pt 4pt'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is UserAgent.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is UserAgent.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is UserAgent.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
transparent		
inherit		

C.80. bottom

Description

This is used for absolutely and relatively positioned elements only. It sets the distance from the bottom edge of the containing element to the bottom edge of this element.

Default value

auto

C.81. break-after

Description

Use this element to insert a page break after this element.

Default value

auto

Values

auto	
column	
page	A page break will occur after this element.
inherit	

C.82. break-before

Description

Use this element to insert a page break before this element.

Default value

auto

Values

auto	
column	
page	A page break will occur before this element.
inherit	

C.83. character

Description

This attribute sets the character to be inserted by a fo:character element. For instance to insert the character 'A' into the content you would use an fo:character element like this:

```
<fo:character character='A'/>
```

Default value

This attribute has no default value, you must provide a value.

C.84. color

Description

This sets the foreground color of text.

Default value

inherited from parent

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

C.85. column-count

Description

Sets the number of columns in a body region. Only the body region can have more than one column.

For example to create a body region with three columns set column-count to 3 like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee' column-count='3'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

Default value

1

Values

<integer>	A non-negative integer. Sets the number of columns to this value.

C.86. column-gap

Description

Sets the gap between columns in a body region with column-count > 1. Only the body region can have more than one column.

For example to create a body region with two columns separated by a 4cm gap set column-count to 2 and column-gap to '4cm' like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"
            column-count='2' column-gap='4cm'/>
      </fo:simple-page-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block>Hello World</fo:block>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
```

Default value

12.0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.87. column-number

Description

This is used on a fo:table-column element to specify which column the fo:table-column element refers to.

This attribute is optional as the column number can be determined from the position of the fo:table-column element in the list of such elements.

```
<fo:table>
    <fo:table-column column-number='1'
      column-width='20%'/>
    <fo:table-column column-number='2'
      column-width='30%'/>
    <fo:table-column column-number='3'
      column-width='50%'/>
    <fo:table-body>
         <fo:table-row>
              <fo:table-cell>col 1</fo:table-cell>
```

```
                    <fo:table-cell>col 2</fo:table-cell>
                    <fo:table-cell>col 3</fo:table-cell>
               </fo:table-row>
          </fo:table-body>
      </fo:table>
```

Default value

current column number

C.88. column-width

Description

This is used on a fo:table-column element to specify the width of the column the fo:table-column element refers to.

For example to set the widths of three columns to 20%, 30% and 50% you would do this:

```
      <fo:table>
          <fo:table-column column-number='1'
            column-width='20%'/>
          <fo:table-column column-number='2'
            column-width='30%'/>
          <fo:table-column column-number='3'
             column-width='50%'/>
          <fo:table-body>
               <fo:table-row>
                    <fo:table-cell>col 1</fo:table-cell>
                    <fo:table-cell>col 2</fo:table-cell>
                    <fo:table-cell>col 3</fo:table-cell>
               </fo:table-row>
          </fo:table-body>
      </fo:table>
```

Default value

This attribute has no default value, you must provide a value.

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.89. content-height

Description

This is used on a graphic element such as a fo:external-graphic to set the height of the image.

The size of an image and the size of the area containing it are two seperate things. The height and width attributes set the size of the area containing the image, the content-height and content-width attributes set the size of the image itself.

Percentage values refer to percentages of the actual size of the image as determined from the image file.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	
scale-to-fit	

C.90. content-width

Description

This is used on a graphic element such as a fo:external-graphic to set the width of the image.

The size of an image and the size of the area containing it are two seperate things. The height and width attributes set the size of the area containing the image, the content-height and content-width attributes set the size of the image itself.

Percentage values refer to percentages of the actual size of the image as determined from the image file.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	
scale-to-fit	

C.91. display-align

Description

This attribute sets the vertical alignment of content contained within the element with this attribute.

Default value

inherited from parent

Values

auto	
before	Align to before edge which for unrotated content is the top.
center	Align to center
after	Align to after edge which for unrotated content is the bottom.

C.92. end-indent

Description

This attribute sets indentation of content from the end edge of the containing area. For unrotated content the end edge is the right edge.

This attribute sets the indentation of the content contained in the element. The content will be positioned the required distance from the right edge of the containing area, and any padding and border will then be placed outside the content.

For CSS style alignment of nested elements use the margin-left and margin-right attributes instead of start-indent and end-indent.

Default value

0pt

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.93. ends-row

Description

Within a fo:table-body (or fo:table-header and fo:table-footer) element a table has fo:table-cell elements. Normally cells are placed inside a fo:table-row element, but it is possible to place the cells directly below the fo:table-body element and not have any fo:table-row elements. In this case the formatter determines formation of rows by looking for ends-row and starts-row attributes on each fo:table-cell. If a fo:table-cell ends the row then the ends-row attribute should be set to "true", otherwise it should be set to "false" or not used at all.

A table which has two rows of three cells each and is created without row elements looks like this:

```
<fo:table>
    <fo:table-body>
         <fo:table-cell starts-row='true'>col 1</fo:table-cell>
         <fo:table-cell>col 2</fo:table-cell>
         <fo:table-cell ends-row='true'>col 3</fo:table-cell>
         <fo:table-cell starts-row='true'>col 1</fo:table-cell>
         <fo:table-cell>col 2</fo:table-cell>
         <fo:table-cell ends-row='true'>col 3</fo:table-cell>
    </fo:table-body>
</fo:table>
```

Default value

false

Values

false	This cell does not end the row
true	This cell ends the row

C.94. extent

Description

The extent attribute determines how large a region is. It is used on region elements other then the fo:region-body element.

The extent is the size of the region. The outer edge of the region is calculated from the edge of the page plus any margin on the fo:simple-page-master element. The inner edge of the region is the outer edge plus the value of the extent attribute.

Percentage values refer to the size of the page.

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.95. external-destination

Description

This attribute destination of an fo:basic-link element used to create a hyperlink in the document.

The format of the external-destination attribute must be a URI specification (RFC2396) as described below.

To link local file the format should be:

```
external-destination='url(external.pdf)'
```

or to link to a website use a format like this:

```
external-destination
    ='url(http://www.xmlpdf.com/builds/ibex.pdf)'
```

Default value

Values

<uri-specification>	A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

C.96. float

Description

Specifies how the block which is floated should be positioned. Specify float="start" or float="before" to move the block to the start of the page. Specify float="left" to position content to the left side of the page and have other content flow around the right side of the positioned content.

Default value

none

Values

inherit
before

start	
end	not implemented
left	
right	not implemented

C.97. flow-name

Description

This attribute is used on fo:flow and fo:static-content elements to define which region the content from the fo:flow and fo:static-content is placed.

For content to be placed on the page the flow-name attribute must correspond to the region-name attribute of a region of the current page layout. If the flow-name is not the same as one of the region names the content contained in the fo:flow and fo:static-content is discarded.

Default value

This attribute has no default value, you must provide a value.

Values

<name>	use a value which matches a region-name used on one of the regions on the current fo:simple-page-master.

C.98. font

Description

This attribute is shorthand for the font-style, font-variant, font-weight, font-size, font-family, line-height attributes.

Typically the font attribute will be set to a value which defines the font name and size plus possibly bold or italic. Some example of this are:

```
font='12pt arial'
```

```
font='bold 12pt arial'
```

```
font='bold 12pt "minion regular"'
```

The elements of the font attribute must be specified in this order:

style (normal, italic)

variant (normal, smallcaps)

weight (bold, bolder, lighter etc.)

size (1em, 12pt)

line height (12pt/14pt)

font-family (helvetica)

If the font name contains spaces it should be placed in quotes. If the attribute value is in single quotes, place the font name in double quotes like this:

```
font='12pt "minion regular" '
```

If the attribute value is in double quotes, place the font name in single quotes like this:

```
font="12pt 'minion regular' "
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

C.99. font-family

Description

This sets the font family for the element.

This attribute can be set to a single font name like this:

```
font-family='arial'
```

or a list of fonts separated by commas, like this:

```
font-family='arial, "minion regular"'
```

If the font name contains spaces it should be placed in quotes. If the attribute value is in single quotes, place the font name in double quotes like this:

```
font='12pt "minion regular" '
```

If the attribute value is in double quotes, place the font name in single quotes like this:

```
font="12pt 'minion regular' "
```

In addition to actual font names the following values can be used:

"serif", "sans-serif", "cursive", "fantasy", "monospace"

These names are mapped to actual font names by the UserAgent. To change the mapping call the UserAgent.setFontSubstitution() API.

For instance to set the "serif" font to "arial", use this code:

```
UserAgent.setFontSubstitution( "serif", "arial", false );
```

Default value

The value of UserAgent.DefaultFontFamily or inherited from parent

C.100. font-size

Description

This sets the font size of this element and the elements it contains.

Typical values are show here:

```
font-size='12pt'
```

```
font-size='1.2em'
```

Values which set the font size relative to the font size of the containing element can also be used like this:

```
font-size='smaller'
```

Percentage sizes refer to the font size of the containing element.

Default value

medium

Values

xx-small	Set the font size to the value of UserAgent.XX_Small which defaults to "7pt"
x-small	Set the font size to the value of UserAgent.X_Small which defaults to "8.3pt"
small	Set the font size to the value of UserAgent.Small which defaults to "10pt"
medium	Set the font size to the value of UserAgent.Medium which defaults to "12pt"
large	Set the font size to the value of UserAgent.Large which defaults to "14.4pt"
x-large	Set the font size to the value of UserAgent.X_Large which defaults to "17.4pt"
xx-large	Set the font size to the value of UserAgent.XX_Small which defaults to "20.7pt"
smaller	Set the font size to the parent font size multipled by the value of UserAgent.Smaller which defaults to "0.8em"
larger	Set the font size to the parent font size multipled by the value of UserAgent.Larger which defaults to "1.2em"
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.101. font-style

Description

This sets the font style of this element and elements it contains.

Typical values are show here:

```
font-style='italic'
```

Default value

normal

Values

normal	The style is the same as the style of the parent element.
italic	The style is italic.
oblique	The style is italic.
inherit	

C.102. font-weight

Description

This sets the font weight of this element and elements it contains.

The specification supports numeric values. These are mapped as follows:

900	bold
800	bold
700	bold
600	normal
500	normal
400	normal
300	normal
200	normal
100	normal

Default value

normal

Values

100-900	See the table above
normal	The weight is inherited from the parent element.
bold	The text will be bold.
inherit	

C.103. format

Description

In conjuction with grouping-separator and grouping-size this attribute sets the format to be used when formatting page numbers contained within this fo:page-sequence.

Default value

1

Values

1	Use numeric formatting so page numbers will be 1,2,3 ..
i	Use roman formatting so page numbers will be i, ii, iii, iv, v ..

C.104. height

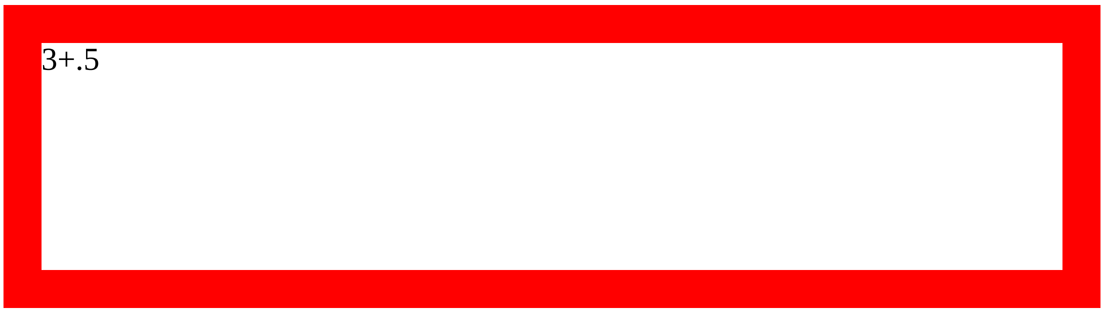
Description

Sets the height of the content of an element excluding padding and borders. This means an element with height='3cm' and border='.25cm' will have a height including borders and padding of 3.5cm.

This example shows the effect of the height attribute on the content of the block:

```
<fo:block space-before='1cm' height='3cm'
        border='.5cm solid red'>3+.5</fo:block>
```

This produces this output:


3+.5

By pressing Control-U in Acrobat Reader you can measure the content and see that the area *within the borders* is 3cm high.

To set minumum and maximum height values use the block-progression-dimension attribute.

Default value

auto

C.105. id

Description

This attribute is set on elements which need to be referenced from somewhere else in the document.

An example of this is the fo:page-number-citation element which inserts into the document the page number some other content appears on. The content whose page number we want to retrieve is given an id attribute, and the fo:page-number-citation sets its ref-id attribute to the same value.

An example of this is:

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">
         <fo:region-body margin="2.5cm" region-name="body"/>
      </fo:simple-page-master>
   </fo:layout-master-set>

  <fo:page-sequence master-reference="simple">
      <fo:flow flow-name="body">
         <fo:block id='22'>
         Hello
         </fo:block>
```

```
        </fo:flow>
    </fo:page-sequence>

    <fo:page-sequence master-reference="simple">
        <fo:flow flow-name="body">
           <fo:block>
             The block with id='22' is on page
             <fo:page-number-citation ref-id='22'/>
           </fo:block>
        </fo:flow>
    </fo:page-sequence>

  </fo:root>
```

Default value

a unique value generated by Ibex

Values

<id>	a unique string

C.106. initial-page-number

Description

This attribute sets the page number of the first page create by a fo:page-sequence element.

Default value

auto

C.107. inline-progression-dimension

Description

Sets the dimension of content in the inline progression direction which for an unrotated page is across the page.

The content of an element excludes padding and borders. This means an element with inline-progression-dimension='3cm' and border='.25cm' will have a width including borders and padding of 3.5cm.

Can be set as a single value such as:

```
inline-progression-dimension='20cm'
```

or you can specify minimum and maximum values like this:

```
inline-progression-dimension.minimum='5cm'
inline-progression-dimension.maximum='25cm'
```

Default value

auto

Values

auto

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
<length-range>	The value has three sub-components, namely minimim, optimum and maximum. Each of these can be set to a <length> value.
inherit	

C.108. internal-destination

Description

This sets the destination of a fo:basic-link element.

This should be set a value used as the id attribute of the element to be linked to.

Default value

""

C.109. keep-together

Description

Set this attribute to 'always' to keep content together on one page. If content with keep-together='always' will not fit on what remains of a page it will be moved to the next page. If it is larger than the region in which it is being placed it will be split.

Default value

auto

C.110. keep-with-next

Description

Set this attribute to 'always' to keep the content with the next element in the FO. If both elements do not fit on a page they will both be moved to the next page.

This is typically used to keep a heading together with the content which follows.

Any number of elements can be kept together by having keep-with-next='always' set on each one. If the list to be kept together exceeds the size of the region in which they are being placed they will not be kept together.

Default value

auto

C.111. keep-with-previous

Description

Set this attribute to 'always' to keep the content with the previous element in the FO. If both elements do not fit on a page they will both be moved to the next page.

This is typically used to keep a the last two rows of a table together so that a single row is never displayed by itself.

Any number of elements can be kept together by having keep-with-previous='always' set on each one. If the list to be kept together exceeds the size of the region in which they are being placed they will not be kept together.

Default value

auto

C.112. leader-length

Description

This sets the length of a fo:leader element.

This can be set as three components for the minimum, optimum and maximum values like this:

```
      leader-length.mimimum="10pt"
      leader-length.optimum="50%"
      leader-length.maximum="100%"
```

Or alternatively all three components can be set to the same value like this:

```
      leader-length="100%"
```

The default values for each component are:

leader-length.mimimum="0pt"

leader-length.optimum="12pt"

leader-length.maximum="100%"

Default value

see description

C.113. leader-pattern

Description

This sets the appearance of a leader element.

Default value

space

Values

space	The leader will be blank. This is useful for justification of text.
dots	The leader will be a dotted line.
rule	The leader will be a solid line.

C.114. left

Description

When used on an absolutely or relatively positioned element this sets the offset from the left edge of the container to the left edge of this element.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.115. linefeed-treatment

Description

This sets the way in which linefeeds in the FO appear in the output. By default linefeeds are treated as spaces only. See page 53 for a detailed example of the effects of this attribute.

Default value

treat-as-space

Values

treat-as-space	Linefeeds in the FO become spaces in the output.
preserve	Linefeeds in the FO become linefeeds in the output.

C.116. line-height

Description

Sets the height of a line.

Percentage values refer to the current font size.

Default value

normal

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
normal	Line height is 1.2 times the font size.

C.117. margin

Description

This is a shorthand way of setting margin-top, margin-bottom, margin-right and margin-left.

To set all margins to the same size use a single value like this:

```
margin='1pt'
```

If there are two values the top and bottom margins are set to the first value and the side margins are set to the second, like this:

```
margin='1pt 3pt'
```

If there are three values the top margin is set to the first value, the side margins are set to the second, and the bottom is set to the third like this:

```
margin='1pt 2pt 3pt'
```

If there are four values the top margin is set to the first value, the right margin is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

```
margin='1pt 2pt 3pt 4pt'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.118. margin-bottom

Description

Sets the bottom margin on an element. For example:

```
margin-bottom='1pt'
```

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.119. margin-left

Description

Sets the left margin on an element. For example:

```
margin-left='1pt'
```

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.120. margin-right

Description

Sets the right margin on an element. For example:

```
margin-right='1pt'
```

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.121. margin-top

Description

Sets the top margin on an element. For example:

```
margin-top='1pt'
```

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.122. marker-class-name

Description

Sets the name for a fo:marker which is then used on a fo:retrieve-marker element to retrieve the content contained in that fo:marker.

See fo:marker for an example.

Default value

This attribute has no default value, you must provide a value.

C.123. master-name

Description

This is a unique name given to a fo:simple-page-master or fo:page-sequence-master and then used as the master-reference attribute of a fo:page-sequence to specify which page master will be used to lay out the content of the fo:page-sequence.

Default value

This attribute has no default value, you must provide a value.

C.124. master-reference

Description

Both fo:simple-page-master and fo:page-sequence-master have a unique master-name attribute which is used as the master-reference attribute of a fo:page-sequence to specify which page master will be used to lay out the content of the fo:page-sequence.

Default value

This attribute has no default value, you must provide a value.

C.125. number-columns-repeated

Description

This is used on a fo:table-column element to indicate how many columns the fo:table-column element applies to.

Default value

1

C.126. number-columns-spanned

Description

This is used on a fo:table-cell element to specify how many columns the cell spans. This is functionally similar to the HTML colspan attribute.

Default value

1

C.127. number-rows-spanned

Description

This is used on a fo:table-cell element to specify how many rows the cell spans. This is functionally similar to the HTML rowspan attribute.

Default value

1

C.128. orphans

Description

This specifies the number of lines of text which must appear in a paragraph before a page break. At the default setting of '2' a single line will never appear by itself at the bottom of a page. If there is room for only a single line on a page (and the paragraph has more than one line) the whole paragraph will be shifted to the next page.

Increasing the value increases the number of lines which must appear on a page before a page break.

See also widows.

Default value

2

C.129. overflow

Description

This attribute determines whether content which exceeds the size of an element should be displayed or not.

An example of this is the fixed size region elements such as fo:region-before which have their size set by the extent attribute. If content is placed in the region using a fo:static-content element the content may be too large for the region. If this happens and the overflow attribute is set to 'hidden' the content will not appear.

Default value

auto

Values

hidden	Content which exceeds the elements boundaries will be discarded.
auto	Content which exceeds the elements boundaries will be displayed.
visible	Content which exceeds the elements boundaries will be displayed.

C.130. padding

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This is a shorthand way of setting padding-top, padding-bottom, padding-right and padding-left.

To set all paddings to the same size use a single value like this:

```
padding='1pt'
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.131. padding-after

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the after edge of an element which for an unrotated area is the bottom edge.

```
padding-after='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.132. padding-before

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the before edge of an element which for an unrotated area is the top edge.

```
padding-before='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.133. padding-bottom

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the bottom edge of an element.

```
padding-bottom='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.134. padding-end

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the end edge of an element which for an unrotated area is the right edge.

```
padding-end='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.135. padding-left

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the left edge of an element.

```
padding-left='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.136. padding-right

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the right edge of an element.

```
padding-right='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.137. padding-start

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the start edge of an element which for an unrotated area is the left edge.

```
padding-start='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.138. padding-top

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the top edge of an element.

```
padding-top='1pt'
```

Default value

0pt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.139. page-height

Description

This is used on a fo:simple-page-master element to set the height of a page.

If not set or if set to 'auto' the page height is determined from the UserAgent.PageHeight property.

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.140. page-width

Description

This is used on a fo:simple-page-master element to set the width of a page.

If not set or if set to 'auto' the page width is determined from the UserAgent.PageWidth property.

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.141. precedence

Description

This is used on fo:region-before and fo:region-after elements to control whether the top and bottom regions take precedence over (i.e. extend into the corners over) the side regions.

Default value

false

Values

false
true

C.142. provisional-distance-between-starts

Description

This applies to the fo:list-block element and sets the distance (in each fo:list-item) between the start of the label element and the start of the body element.

This is not the same as the width of the label element because the width of the label element is reduced by the provisional-label-separation value.

See fo:list-block for an example.

Default value

24pt

C.143. provisional-label-separation

Description

This applies to the fo:list-block element and sets the distance between the end of the label element and the start of the body element.

See fo:list-block for an example and also provisional-distance-between-starts.

Default value

6pt

C.144. reference-orientation

Description

This attribute is used to set the rotation of whole pages (when used on fo:simple-page-master), regions (when used on region element), blocks (when used on fo:block-container) and fo:table elements.

Rotation is counter-clockwise.

See fo:block-container for an example.

Default value

0

Values

0
90
180
270
-90
-180
-270

inherit

C.145. ref-id

Description

This attribute is used on the fo:page-number-citation to identify which the element for which we want to retrieve the page number. This should match the value of the id attribute on the other element.

See fo:page-number-citation for an example.

Default value

This attribute has no default value, you must provide a value.

C.146. retrieve-boundary

Description

This attribute is used on a fo:retrieve-marker to specify limits on which markers should be retrieved.

See fo:marker for a complete example.

Default value

page-sequence

Values

page
page-sequence
document

C.147. retrieve-class-name

Description

This attribute is used on a fo:retrieve-marker to specify which marker is to be retrieved. This attribute specifies which class of marker is retrieved and the retrieve-boundary and retrieve-position attributes are used to choose one of the markers in that class.

See fo:marker for a complete example.

Default value

This attribute has no default value, you must provide a value.

C.148. retrieve-position

Description

This attribute is used on a fo:retrieve-marker to specify which marker is to be retrieved. The retrieve-class-name attribute specifies which class of marker is retrieved and the retrieve-boundary and retrieve-position attributes are used to choose one of the markers in that class.

See fo:marker for a complete example.

Default value

first-starting-within-page

Values

first-starting-within-page	Use the first marker which appears starts on this page
first-including-carryover	Use the first marker which has any content on this page
last-starting-within-page	
last-ending-within-page	

C.149. right

Description

For an absolutely positioned element this specifies the distance between the right edge of the containing element and the right edge of this element.

Default value

auto

C.150. rule-thickness

Description

This is used on the fo:leader element to specify the thickness (i.e. height) of the line the leader creates.

Default value

1pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.151. scaling

Description

This is used on graphic elements fo:external-graphic and fo:instream-foreign-object to specify how the image should be scaled.

If the scaling is uniform a change to the image size using content-height or content-width will result in a corresponding change in the other dimension to preserve the aspect ratio. If scaling is non-uniform a change to height or width will not change the other dimension and the aspect ratio will be changed.

Default value

uniform

Values

uniform	See above.
non-uniform	See above.

C.152. space-after

Description

This attribute is used to define the amount of space which appears between this element and the next.

This attribute can be set as a single value like this:

```
space-after='3mm'
```

or individual components can be set like this:

```
space-after.minimum='3pt'
space-after.optimum='4pt'
space-after.maximum='5pt'
```

Space resolution in XSL-FO is complicated. If two elements have space after the first one and before the second one, usually the space is combined using a formula so that generally speaking the largest space will be used.

For example if there are two blocks A and B, and A has space-after='3cm' and B has space-before='2cm' the space between the blocks will not be the sum of the two spaces (ie. 5cm) it will be the largest of the two, ie. 3cm.

To prevent the two spaces from being merged, and get the sum of the two spaces you can use the precedence component like this:

```
space-after='3cm' space-after.precedence='force'
```

Precedence can also be assigned a number. If there are two spaces to be merged and they have different precedence values the one with the highest value will be used. For example:

```
<fo:block space-after='3cm' space-after.precedence='5'>
A
</fo:block>
<fo:block space-before='1cm' space-after.precedence='6'>
B
</fo:block>
```

In this case the space between the two blocks will be 1cm because the second block has the higher precedence value so its space value is the one which is used.

Space which appears before a block at the top of a region is usually discarded. To avoid this and make the space appear use the conditionality component like this:

```
space-before='3cm' space-before.conditionality='retain'
```

To make matters even more complex, the space after an element refers to the space between the last mark made by this element and the first mark made by the next element. This means we need to consider child elements of the two elements whose space is being merged.

For example the block A below has a child block A2 which has a space-after attribute. This means when Ibex merges the space between A and B, it also considers the space between A2 and B.

```
<fo:block space-after='3cm' >
   A
<fo:block space-after='4cm' >
```

```
    A2
  </fo:block>
  </fo:block>
  <fo:block space-before='1cm' >
    B
  </fo:block>
```

so the space between A and B will be 4cm because this is the largest value. If B had a child block this would also be considered.

And it gets worse. In the example shown above A2 makes the last mark on the page made by the A block and its children. If A had a bottom border, this border would then be the last mark made by the A block and its children (because the border of A is after A2) and the merging formula would not consider A2 (as it does not now make the last mark) and so the gap between A and B would now be 3cm.

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.153. space-before

Description

This attribute is used to define the amount of space which appears between this element and the previous one.

This attribute can be set as a single value like this:

```
space-before='3mm'
```

or individual components can be set like this:

```
space-before.minimum='3pt'
space-before.optimum='4pt'
space-before.maximum='5pt'
```

Space resolution in XSL-FO is complicated. See space-after for a detailed description of space resolution.

Default value

0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.154. span

Description

This attribute is used on block-level (fo:block,fo:table,fo:list-block) elements whose immediate parent element is a fo:flow. The span indicates if the block element should span all the columns of a multi-column fo:region-body. The only options are 'none' and 'all'. It is not possible to span some but not all columns.

Default value

none

Values

none	Span one column
all	Span all columns
inherit	

C.155. src

Description

This specifies the source for a fo:external-graphic element.

The src attribute is called a *uri-specification* and must follow the following rules:

> A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or both be absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

This means the following are all valid values for the src attribute:

> uri(ibex.jpg)
>
> uri("ibex.jpg")
>
> uri('ibex.jpg')
>
> url(http://www.xmlpdf.com/images/download2.gif)

Default value

This attribute has no default value, you must provide a value.

C.156. start-indent

Description

This attribute sets indentation of content from the start edge of the containing area. For unrotated content the start edge is the left edge.

This attribute sets the indentation of the content contained in the element. The content will be positioned the required distance from the right edge of the containing area, and any padding and border will then be placed outside the content.

For CSS style alignment of nested elements use the margin-left and margin-right attributes instead of start-indent and end-indent.

Default value

0pt

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

C.157. starts-row

Description

Within a fo:table-body (or fo:table-header and fo:table-footer) element a table has fo:table-cell elements. Normally cells are placed inside a fo:table-row element, but it is possible to place the cells directly below the fo:table-body element and not have any fo:table-row elements. In this case the formatter determines formation of rows by looking for ends-row and starts-row attributes on each fo:table-cell. If a fo:table-cell ends the row then the ends-row attribute should be set to "true", otherwise it should be set to "false" or not used at all.

A table which has two rows of three cells each and is created without row elements looks like this:

```
<fo:table>
 <fo:table-body>
  <fo:table-cell starts-row='true'>col 1</fo:table-cell>
   <fo:table-cell>col 2</fo:table-cell>
    <fo:table-cell ends-row='true'>col 3</fo:table-cell>
    <fo:table-cell starts-row='true'>col 1</fo:table-cell>
    <fo:table-cell>col 2</fo:table-cell>
    <fo:table-cell ends-row='true'>col 3</fo:table-cell>
  </fo:table-body>
</fo:table>
```

Default value

false

Values

false	This cell does not start a new row.
true	This cell starts a new the row.

C.158. table-layout

Description

This attribute controls whether the layout of a fo:table (which means the column widths) is calculated from the content of the cells or from fo:table-column elements.

Use 'fixed' to calculate column widths from fo:table-column elements. This is the recommended approach. It is faster and makes the output file look consistent when using different data.

Default value

auto

Values

auto	
fixed	see above

C.159. table-omit-footer-at-break

Description

By default a fo:table-footer element is repeated before every table break. If you set this attribute to 'true' the table footer will appear only once, at the end of the table.

Default value

false

Values

true	Footer appears once at end of table
false	Footer appears at every page break

C.160. table-omit-header-at-break

Description

By default a fo:table-header element is repeated after every table break. If you set this attribute to 'true' the table header will appear only once, at the beginning of the table.

Default value

false

Values

true	header appears once at start of table
false	header appears after every page break

C.161. text-align

Description

This sets the text alignment of text contained within the element. This does not align the last (or only) line of a paragraph - see the text-align-last attribute for aligning the last line.

Default value

start

Values

start	same as left for unrotated content
left	

center	
end	same as right for unrotated content
right	
justify	

C.162. text-align-last

Description

This sets the text alignment of text last line of a paragraph - see the text-align attribute for aligning lines other than the last one.

Default value

start

Values

start	same as left for unrotated content
left	
center	
end	same as right for unrotated content
right	
justify	

C.163. white-space-collapse

Description

This controls how multiple contiguous whitespace in the FO is treated by Ibex. By default after processing of linefeeds all remaining runs of two or more consecutive spaces are replaced by a single space, then any remaining space immediately adjacent to a remaining linefeed is also discarded.

See page 53 for a detailed example of the effects of this attribute.

Default value

true

C.164. white-space-treatment

Description

This controls how whitespace characters in the FO are treated by Ibex.

See page 53 for a detailed example of the effects of this attribute.

Default value

ignore-if-surrounding-linefeed

C.165. widows

Description

This specifies the number of lines of text which must appear in a paragraph at the top of a page. At the default setting of '2' a single line will never appear by itself at the top of a page. If possible a line from the previous page will be moved to the the current page so that 2 lines of text appear at the top of the page. If this is not possible (perhaps because of the orphans setting) the whole paragraph will be moved to the current page.

Increasing the value increases the number of lines which must appear on a page.

See also orphans.

Default value

2

C.166. width

Description

This sets the desired width for an element. This is shorthand way of setting all three components of the inline-progression-dimension attribute.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

C.167. wrap-option

Description

This option controls word wrapping within an element.

Default behaviour for text within an fo:block is for words which do not fit on one line to wrap to the next line and the height of the block to increase. If wrap-option="no-wrap" then words which do not fit on the first line are discarded if overflow='hidden'.

See also overflow

Default value

wrap

Values

inherit	
wrap	words wrap to the next line
no-wrap	words do not wrap to the next line

C.168. z-index

Description

This attribute controls the positioning of one element over another. By default a more deeply nested element will appear over its container element but by changing the z-order you can change which elemnets appear over which other elements.

Default value

auto

Appendix D.

Formatting Object Summary

This chapter describes which extent to which Ibex supports the XSL W3C Recommendation dated 15 October 2001.

Objects are listed grouped together in the same way in which they appear in the specification.

D.1. Declaration, pagination and layout formatting objects

Element	Status
root	implemented
page-sequence	implemented
page-sequence-master	implemented
single-page-master-reference	implemented
repeatable-page-master-reference	implemented
repeatable-page-master-alternatives	implemented
conditional-page-master-reference	implemented
layout-master-set	implemented
simple-page-master	implemented
region-body	implemented
region-before	implemented
region-after	implemented
region-start	implemented
region-end	implemented
declarations	implemented to contain color-space
color-profile	implemented
flow	implemented

Element	Status
static-content	implemented
title	not implemented

D.2. Block formatting objects

Element	Status
block	implemented
block-container	implemented

D.3. Inline formatting objects

Element	Status
bidi-override	not implemented
character	implemented
initial-property-set	not implemented
external-graphic	implemented
instream-foreign-object	implemented
inline	implemented
inline-container	not implemented
leader	implemented
page-number	implemented
page-number-citation	implemented

D.4. Table formatting objects

Element	Status
table	implemented
table-and-caption	implemented
table-column	implemented
table-caption	implemented
table-header	implemented
table-footer	implemented
table-body	implemented
table-row	implemented

Element	Status
table-cell	implemented

D.5. List formatting objects

Element	Status
list-block	implemented
list-item	implemented
list-item-table-body	implemented
list-item-label	implemented

D.6. Link and multi formatting objects

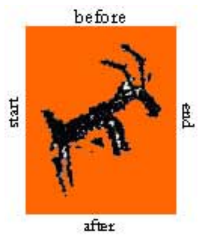
Element	Status
basic-link	implemented
multi-switch	not implemented
multi-case	not implemented
multi-toggle	not implemented
multi-properties	not implemented
multi-property-set	not implemented

D.7. Out-of-line formatting objects

Element	Status
float	left side only
footnote	implemented
footnote-body	implemented

D.8. Other formatting objects

Element	Status
wrapper	implemented
marker	implemented
retrieve-marker	implemented



Appendix E.

Formatting Property Summary

This chapter describes which extent to which Ibex supports the XSL W3C Recommendation dated 15 October 2001.

Properties are listed grouped together in the same way in which they appear in the specification.

Property	Status
absolute-position	implemented
active-state	not implemented
alignment-adjust	implemented
alignment-baseline	implemented
auto-restore	not implemented
azimuth	not implemented
background	implemented
background-attachment	not implemented
background-color	implemented
background-image	implemented
background-position	implemented
background-position-horizontal	implemented
background-position-vertical	implemented
background-repeat	implemented
baseline-shift	implemented
blank-or-not-blank	implemented
block-progression-dimension	implemented
border	implemented

Property	**Status**
border-after-color	implemented
border-after-precedence	implemented
border-after-style	implemented
border-after-width	implemented
border-before-color	implemented
border-before-precedence	implemented
border-before-style	implemented
border-before-width	implemented
border-bottom	implemented
border-bottom-color	implemented
border-bottom-precedence	implemented
border-bottom-style	implemented
border-bottom-width	implemented
border-collapse	implemented
border-color	implemented
border-end-color	implemented
border-end-precedence	implemented
border-end-style	implemented
border-end-width	implemented
border-left	implemented
border-left-color	implemented
border-left-precedence	implemented
border-left-style	implemented
border-left-width	implemented
border-right	implemented
border-right-color	implemented
border-right-precedence	implemented
border-right-style	implemented
border-right-width	implemented
border-separation	implemented

Property	**Status**
border-spacing	implemented
border-start-color	implemented
border-start-precedence	implemented
border-start-style	implemented
border-start-width	implemented
border-style	implemented
border-top	implemented
border-top-color	implemented
border-top-precedence	implemented
border-top-style	implemented
border-top-width	implemented
border-width	implemented
bottom	implemented
break-after	implemented
break-before	implemented
caption-side	not implemented
character	implemented
clear	floats are not implemented yet
clip	not implemented
color	implemented
color-profile-name	implemented
column-count	implemented
column-gap	implemented
column-number	implemented
column-width	implemented
content-height	implemented
content-width	implemented
content-type	implemented
country	not implemented
cue	not implemented

Property	**Status**
cue-after	not implemented
cue-before	not implemented
destination-placement-offset	not implemented
direction	implemented
display-align	implemented
dominant-baseline	not implemented
elevation	implemented
empty-cells	not implemented
end-indent	implemented
ends-row	implemented
extent	implemented
external-destination	implemented
float	not implemented
flow-name	implemented
font	implemented
font-family	implemented
font-selection-strategy	not implemented
font-size	implemented
font-size-adjust	not implemented
font-stretch	not implemented
font-style	implemented
font-variant	implemented
font-weight	implemented
force-page-count	not implemented
format	implemented
glyph-orientation-horizontal	not implemented
glyph-orientation-vertical	not implemented
grouping-separator	implemented
grouping-size	implemented
height	implemented

Property	**Status**
hyphenate	not implemented
hyphenation-character	not implemented
hyphenation-keep	not implemented
hyphenation-ladder-count	not implemented
hyphenation-push-character-count	not implemented
hyphenation-remain-character-count	not implemented
id	implemented
indicate-destination	not implemented
initial-page-number	implemented
inline-progression-dimension	implemented
internal-destination	implemented
intrusion-displace	not implemented
keep-together	implemented
keep-with-next	implemented
keep-with-previous	implemented
language	not implemented
last-line-end-indent	implemented
leader-alignment	implemented
leader-length	implemented
leader-pattern	implemented
leader-pattern-width	implemented
left	implemented
letter-spacing	implemented
letter-value	not implemented
linefeed-treatment	implemented
line-height	implemented
line-height-shift-adjustment	not implemented
line-stacking-strategy	implemented
margin	implemented
margin-bottom	implemented

Property	Status
margin-left	implemented
margin-right	implemented
margin-top	implemented
marker-class-name	implemented
master-name	implemented
master-reference	implemented
max-height	implemented
maximum-repeats	implemented
max-width	implemented
media-usage	not implemented
min-height	implemented
min-width	implemented
number-columns-repeated	implemented
number-columns-spanned	implemented
odd-or-even	implemented
orphans	implemented
overflow	implemented
padding	implemented
padding-after	implemented
padding-before	implemented
padding-bottom	implemented
padding-end	implemented
padding-left	implemented
padding-right	implemented
padding-start	implemented
padding-top	implemented
page-break-after	implemented
page-break-before	implemented
page-break-inside	not implemented
page-height	implemented

Property	Status
page-position	implemented
page-width	implemented
pause	audio properties are not implemented
pause-after	audio properties are not implemented
pause-before	audio properties are not implemented
pitch	audio properties are not implemented
pitch-range	audio properties are not implemented
play-during	audio properties are not implemented
position	implemented
precedence	implemented
provisional-distance-between-starts	implemented
provisional-label-separation	implemented
reference-orientation	implemented
ref-id	implemented
region-name	implemented
relative-align	not implemented
relative-position	not implemented
rendering-intent	not implemented
retrieve-boundary	implemented
retrieve-class-name	implemented
retrieve-position	implemented
richness	audio properties are not implemented
right	implemented
role	not implemented
rule-style	implemented
rule-thickness	implemented
scaling	implemented
scaling-method	implemented
score-spaces	implemented
script	not implemented

Property	**Status**
show-destination	not implemented
size	not implemented
source-document	not implemented
space-after	implemented
space-before	implemented
space-end	implemented
space-start	implemented
span	implemented
speak	audio properties are not implemented
speak-header	audio properties are not implemented
speak-punctuation	audio properties are not implemented
speech-rate	audio properties are not implemented
src	implemented
start-indent	implemented
starting-state	not implemented
starts-row	implemented
stress	audio properties are not implemented
suppress-at-line-break	not implemented
switch-to	not implemented
table-layout	implemented
table-omit-footer-at-break	implemented
table-omit-header-at-break	implemented
target-presentation-context	not implemented
target-stylesheet	not implemented
text-align	implemented
text-align-last	implemented
text-altitude	implemented
text-depth	implemented
text-indent	implemented
text-shadow	not implemented

Property	Status
text-transform	not implemented
top	implemented
treat-as-word-space	not implemented
unicode-bidi	implemented
vertical-align	implemented
visibility	implemented
voice-family	audio properties are not implemented
volume	audio properties are not implemented
white-space	implemented
white-space-collapse	not implemented
white-space-treatment	implemented
widows	implemented
width	implemented
word-spacing	not implemented
wrap-option	implemented
writing-mode	only lr-tb and rl-tb are supported
xml:lang	not implemented
z-index	not implemented

Appendix F.

Licensing

F.1. License file location

Ibex is licensed on a per-developer basis. When you purchase a license for Ibex you will be sent a license file called 'xmlpdf.lic'. This file must be installed in a location where Ibex can locate and load it at runtime.

Ibex searches for the license file in a number of locations. There are:

- the location specified using xmlpdf.licensing.Generator.LicenseFileLocation (see below);
- the location from which the ibex10.dll or ibex11.dll assembly is loaded;
- the location from which the assembly which loaded ibex10.dll or ibex11.dll is loaded;
- the location from which the entry assembly is loaded (see Assembly.GetEntryAssembly());
- the location from which the executing assembly is loaded (see Assembly.GetExecutingAssembly());
- the current directory (see Environment.CurrentDirectory)
- the system directory (see Environment.SystemDirectory)

You can explicitly set a location for the license file by setting the xmlpdf.licensing.Generator.LicenseFileLocation property and passing the full path to the file like this:

```
xmlpdf.licensing.Generator.LicenseFileLocation = @"d:\xmlpdf\testlic\xmlpdf.lic";
```

This must be done before any FODocument objects are created.

F.2. Licensing with ASP.NET

If you are using ASP.NET it is often a problem that the ASP.NET process does not have sufficient access rights to read the xmlpdf.lic file. Just because a file is placed in a directory below the wwwroot directory does not mean that ASP.NET can access the file. Make sure when you install the license file that the ASPNET user has read access to the file.

Appendix G.

Page Layout Examples

The following pages show some examples of different page layouts and the fo:simple-page-master elements used to create them.

region-before

region-body

This page layout is created with the XML below. Note that by default the region-start and region-end regions extend the full height of the page and the region-before and region-after regions are narrowed so as not to overlap the side regions. See the following page for an example where the precedence attribute is used to change this.

```
<fo:simple-page-master master-name="region-example-1">

    <fo:region-body margin="2.5cm" region-name="body"
       background-color='#eeeeee'/>

    <fo:region-before extent="2.5cm" region-name="header"
       background-color='#dddddd'/>

    <fo:region-after extent="2.5cm" region-name="footer"
       background-color='#dddddd'/>

    <fo:region-start extent="2.5cm" region-name="start"
       background-color='#aaaaaa'/>

    <fo:region-end extent="2.5cm" region-name="end"
       background-color='#aaaaaa'/>

</fo:simple-page-master>
```

region-after

region-before

region-body

This page layout is created with the XML below. Note that by default the region-start and region-end regions extend the full height of the page and the region-before and region-after regions are narrowed so as not to overlap the side regions. See the following page for an example where the precedence attribute is used to change this.
This layout differs from the previous page in that the simple-page-master has the margin attribute set to '2.5cm'. This creates a margin of 2.5cm around the entire page, and regions are position with respect the the rectangle created by the margins, not with respect to the edges of the paper.

```
<fo:simple-page-master
master-name="region-example-1M" margin='2.5cm'>

    <fo:region-body margin="2.5cm"
region-name="body"
       background-color='#eeeeee'/>

    <fo:region-before extent="2.5cm"
region-name="header"
       background-color='#dddddd'/>

    <fo:region-after extent="2.5cm"
region-name="footer"
       background-color='#dddddd'/>

    <fo:region-start extent="2.5cm"
region-name="start"
       background-color='#aaaaaa'/>

    <fo:region-end extent="2.5cm"
region-name="end"
       background-color='#aaaaaa'/>

</fo:simple-page-master>
```

region-after

region-before

region-body

This page layout is created with the XML below. Note that the region-before and region-after regions have precedence='true' so they extend the full width of the page and the side regions are reduced in height to the regions do not overlap.

```
<fo:simple-page-master master-name="region-example-1">

    <fo:region-body margin="2.5cm" region-name="body"
       background-color='#eeeeee'/>

    <fo:region-before extent="2.5cm" region-name="header"
       precedence='true' background-color='#dddddd'/>

    <fo:region-after extent="2.5cm" region-name="footer"
       precedence='true' background-color='#dddddd'/>

    <fo:region-start extent="2.5cm" region-name="start"
       background-color='#aaaaaa'/>

    <fo:region-end extent="2.5cm" region-name="end"
       background-color='#aaaaaa'/>

</fo:simple-page-master>
```

region-after