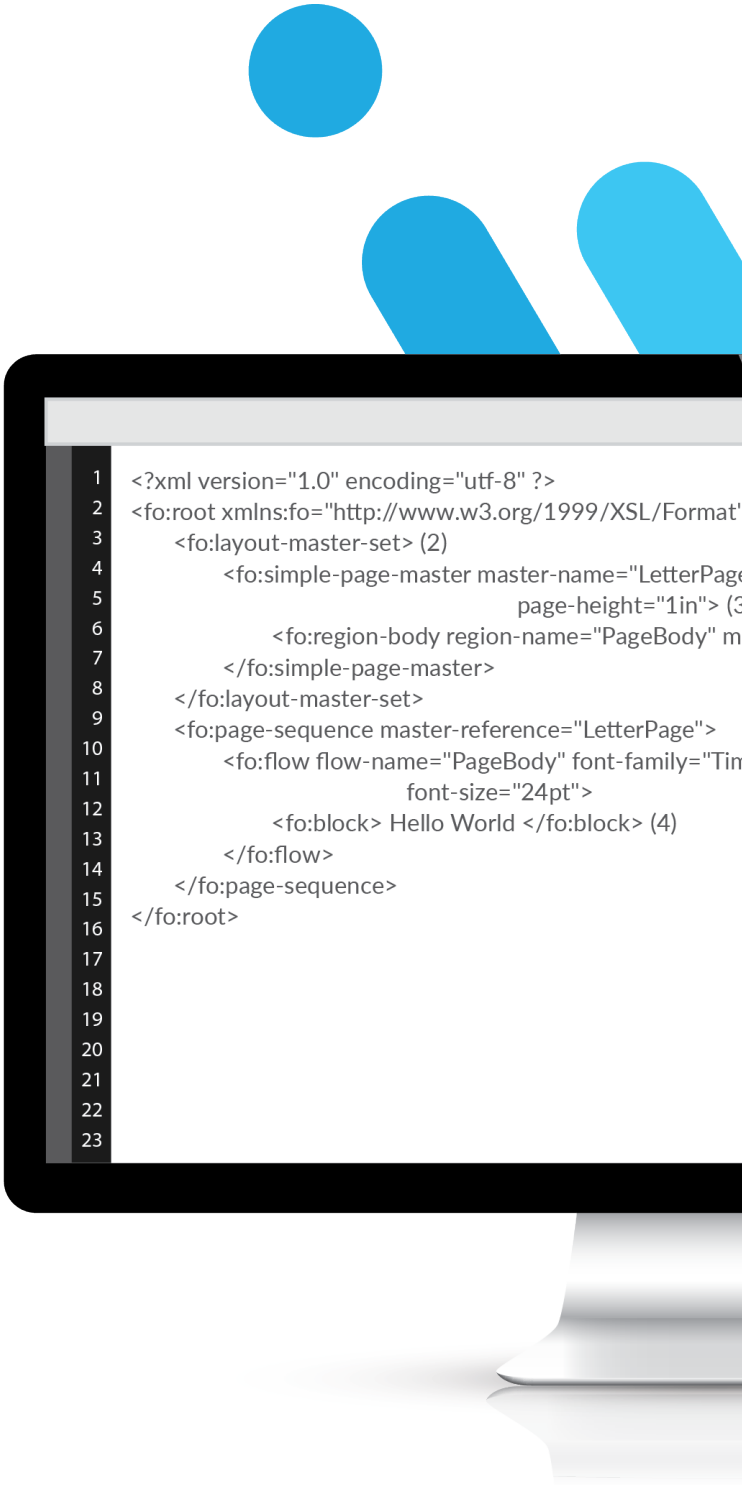
XML to PDF TUTORIAL

This document is designed to help XML programmers better understand how pure data can be converted to PDF using the Ecrion Solution.



ABOUT THE ECRION SOLUTION

The Ecrion Solution is a highly scalable, enterprise class document automation and customer engagement platform. It can be used to automate the creation of electronic documents like invoices, letters, notifications, brochures, proposals and business reports containing charts and graphs, by dynamically generating them from XML or any other supported data source.

At the core of the solution, Ecrion has implemented a high-performance **XML to PDF** software component called a rendering engine. The rendering engine supports two major industry standards: XSL-FO (Extensible Style Language Formatting Objects) describing how an XML document should be formatted for various of media, as well as SVG (Scalable Vector Graphics) used to describe bi-dimensional vector and mixed vector/raster graphics in XML.

FEATURES AND SPECIFICATIONS

INPUT

The solution supports the following data formats:

- XML
- JSON
- SQL databases (SQL Server, Oracle, etc.)
- Other data sources (text, csv, etc.)
- CRM (Salesforce, Microsoft Dynamics, etc.)
-

Formatted input can be included in the document production process as well. The following formats are supported natively without the need for third party software:

- Word, RTF
- HTML, XHTML
- PDF
- XSL-FO
- SVG (and SVG Zipped)
- XChart

OUTPUT

The following print-ready output formats can be produced:

- PDF
- AFP (IBM MO:DCA and IBM:IOCA)
- PostScript

For digital distribution the following formats can be produced as well:

- HTML, HTML5
- Word, PowerPoint, Excel
- Raster Image Files (TIFF, JPEG, GIF, PNG)
- SVG
- Text
- Direct printing

OTHER FEATURES:

- Supports Type1, Type 1 with Postscript Glyphs (OTF), TrueType, AFM font embedding and private fonts.
- Scalable server architecture that can run across multiple CPUs, meeting the high-performance needs of the user's applications.
- It is accessible from a multitude of development environments: REST, .NET and Java Application Programming Interfaces (API's), Python, PHP, etc.
- Multiprocessor/Multithreaded (multiple documents can be processed at the same time; threads can be processed on different CPUs, if available).
- It is available in 32-bit and 64-bit editions.
- Supports 128-bit strength encryption for generated PDF files.
- Has advanced pagination and layout capabilities.
- Advanced SVG support, with additional support for charts and barcodes.
- Can achieve significant speed improvements by enabling the use of additional system memory.
- Generates print-ready PDF documents, compliant with PDF/X, PDF/A standards and with full color management support (ICC profiles).
- Includes a visual design environment

For the latest features and updates, please visit our website at www.ecrion.com

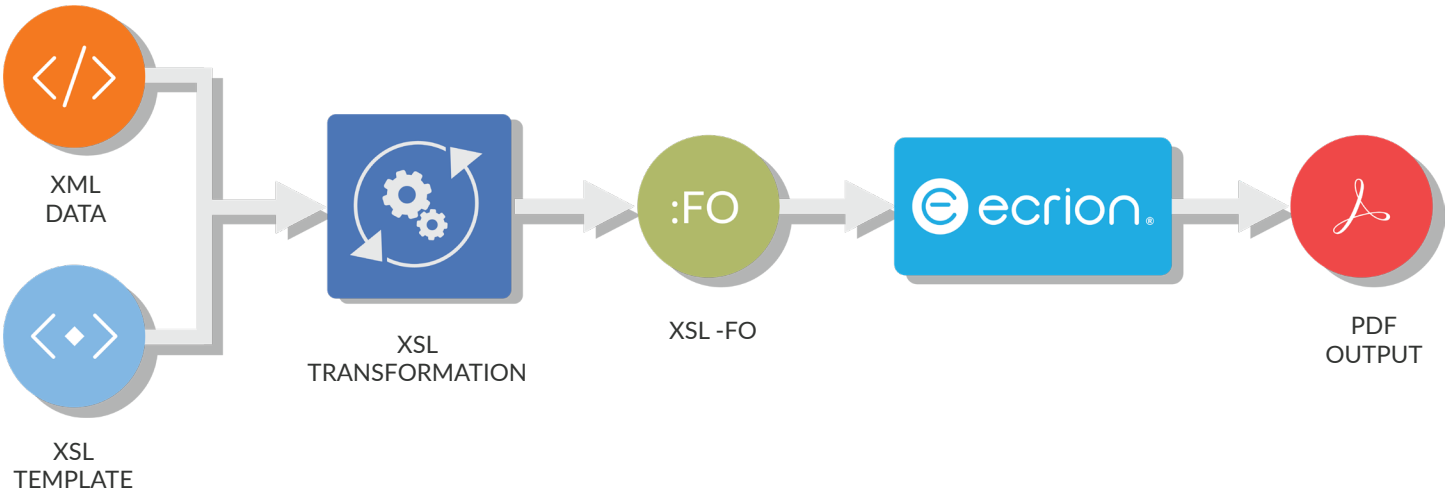
GETTING STARTED

XML is converted to PDF (or transformed) using a language called XSL-FO. XSL-FO has been designed for describing all visual aspects of paginated documents. The well-known HTML is another language used for specifying formatting semantics, but is more usable for documents that are presented on screen, and less for materials created for printing, because it does not support pagination elements like headers and footers, page size specifications, footnotes, etc.

XSL-FO is part of XSL language family:

XSLT (XSL Transformations) deals with transforming XML.
XSL-FO (XSL Formatting Objects) a language that can be used in XSLT for the purpose of presenting the XML.

The following image depicts the steps required to produce a PDF document (or any other supported output format) using XSL:



As you can notice, XML data is transformed together with the XSL stylesheet to produce an intermediary XSL-FO document which in turn is then converted to PDF.

The XSL-FO language uses CSS (Cascading Style Sheets) to describe formatting attributes like fonts, colors and borders, therefore it can be easily learned by HTML and XML developers. This tutorial will help you get accustomed with a few concepts.

Here is the traditional Hello World, XSL-FO style:

```
<?xml version="1.0" encoding="utf-8" ?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"> (1)
    <fo:layout-master-set> (2)
        <fo:simple-page-master master-name="LetterPage" page-width="6.5in"
                               page-height="1in"> (3)
            <fo:region-body region-name="PageBody" margin="0.2in"/>
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="LetterPage">
        <fo:flow flow-name="PageBody" font-family="Times New Roman"
                  font-size="24pt">
            <fo:block> Hello World </fo:block> (4)
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

(1) Any XML document must have only one root, and XSL- FO makes no exception. The root element for an XSL-FO document is fo:root. The word "fo" before column character ":" is called a namespace prefix. An XML namespace is collection of names identified by an unique URL. Its main role is to avoid collisions when a single XML document contains elements and attributes defined by multiple software modules. The "fo" namespace prefix is linked with an unique URL, in this case "http://www.w3.org/1999/XSL/Format" using xmlns attribute. This syntax is based on W3C XML Namespace Spec.

(2)(3) The page's structure is defined using fo:layout-master-set; more information regarding the structure of page sequences can be found in the Pagination chapter. For now, it is enough to say that it declares one type of page, 11.5 x 8 inches (US Letter).

(4) The "Hello World" paragraph is added to the page.

To convert this document into PDF, use the Ecrion Studio or the Ecrion API.

If using the Ecrion API, you would follow the following main steps:

(1) Create a new renderer instance
(2) Create a new request
(3) Send the request and get a response back

```
1  DirectRenderService renderSvc = new DirectRenderService(eosUrl); // (1)
2
3  RenderRequestEntity request = new RenderRequestEntity() // (2)
4  {
5      Input = new Input()
6      {
7          InputFormat = "xslfo",
8          Source = "<fo:root xmlns:fo=\"http://www.w3.org/1999/XSL/Format\">" +
9                   "<fo:layout-master-set>" +
10                    "<fo:simple-page-master master-name=\"all-pages\">" +
11                      "<fo:region-body region-name=\"xsl-region-body\" margin=\"0.7in\"/>" +
12                    "</fo:simple-page-master>" +
13                  "</fo:layout-master-set>" +
14                  "<fo:page-sequence master-reference=\"all-pages\">" +
15                    "<fo:flow flow-name=\"xsl-region-body\">" +
16                      "<fo:block>Hello World!</fo:block>" +
17                    "</fo:flow>" +
18                  "</fo:page-sequence>" +
19              "</fo:root>"
20     },
21
22     PdfOutput = new PdfOutput()
23 };

Stream response = renderSvc.Render(sessionToken, request); // (3)
```