

XF Rendering Server 2007 SDK Tutorial and QuickStarts

Copyright© 2002-2006 ECRION Software. All Rights Reserved.

This document is designed to quickly acquaint COM+, VB and ASP programmers with the programming model of XF Rendering Server 2007 SDK (Software Development Kit). Please contact Technical Support at support@ecrion.com if you need additional information about this product, or visit our Web Site at www.ecrion.com.

Table of Contents

About XF Rendering Server 2007	1
Product Features	1
Before You Get Started	1
Platform Requirements	1
Feedback and Support	1
XF Rendering Server 2007 SDK Tutorial	1
Hello World!	4
Web Access	6
XSL Transformations	7
In Memory Operations	10

Last updated: August 2007

Important Notice: This document and the information within is furnished "as is" and is subject to change without notice. In no event shall the author be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if the author has been advised of the possibility of such damages.

This document was generated using XF Rendering Server 2007. For the latest version, visit [Technical Resources](#) section on our web site.

About XF Rendering Server 2007

XF Rendering Server 2007 is a highly scalable, enterprise class rendering product. It can be used to automate the creation of electronic documents like technical manuals, brochures, proposals, business reports containing charts and graphs, by dynamically generating them from XML.

XF Rendering Server 2007 supports two major industry standards: XSL-FO (Extensible Style Language Formatting Objects) describing how an XML document should be formatted for a variety of media as well as SVG (Scalable Vector Graphics) used to describe two-dimensional vector and mixed vector/raster graphics in XML.

In addition high quality, all-purpose charts can be generated directly from XML using **xChart** XML language. More information can be found in [xChart 1.0 Language Reference](#).

Product Features

- Supports XSL-FO, SVG, xChart as input.
- Produces PDF, Postscript, HTML, GIF, JPEG, PNG, BMP and other formats.
- Supports TrueType and Postscript Type 1 font embedding.
- Scalable server architecture that can run across multiple CPUs, meeting the high-performance needs of your applications.
- Is accessible from a multitude of development environments: C++, VB, ASP, .NET, Java.
- Includes XF Designer 2004 XSL-FO authoring tool.

Before You Get Started ...

To load and run the tutorial samples, you must have the server product installed.

Platform Requirements

XF Rendering Server 2007 can be used in the Microsoft Windows® 2000 Professional and Server, Windows XP and Windows Server 2003 environments.

- Minimum Intel Pentium III, AMD Athlon 500 MHz or better. Intel Pentium IV 2.4 GHz recommended for development computers, dual XEON 3.0 GHz for production servers.
- Minimum 128 Mb RAM, 512 Mb recommended for development computers, 1Gb for productions servers.

Feedback and Support

Send error reports, feature requests, and comments about the SDK documentation or samples directly to the [Technical Support team](#).

Further information about support options can be found on the [Ecrion Web site](#).

XF Rendering Server 2007 SDK Tutorial

The tutorial provides a step-by-step introduction to fundamental areas of programming using the XF Rendering Server 2007 SDK.

SDK samples are included when you install the product. The location of the samples has a path similar to the following. Here, it is assumed that C: is the XF Rendering Server 2007 installation drive:

```
C:\Program Files\Ecrion Software\XF Rendering Server 2007\Samples\
```

The tutorial includes the following sections:

Hello World!	Demonstrates the basic code needed to create a console application using the SDK and generate a PDF document.
Web Access	Describes how to render PDF documents directly into IIS (Internet Information Server) output stream.
XSL Transformations	Demonstrates how to transform XML using a stylesheet and then feed this input into the rendering engine.
In Memory Operations	Demonstrats how to receive rendering output in a memory stream.

Hello World!

The following console program is the PDF version of the traditional "Hello World!" program, which displays the string Hello World!.

```
#import <XFCom.tlb> no_namespace

int main(int argc, char* argv[])
{
    ::CoInitialize(NULL);
    try
    {
        IXFRendererPtr    engine(__uuidof(XFRenderer));

        engine->render("<fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'>"
            " <fo:layout-master-set>"
            "   <fo:simple-page-master master-name='all-pages'>"
            "     <fo:region-body region-name='xsl-region-body' />"
            "   </fo:simple-page-master>"
            " </fo:layout-master-set>"
            " <fo:page-sequence master-reference='all-pages'>"
            "   <fo:flow flow-name='xsl-region-body'>"
            "     <fo:block>Hello World!</fo:block>"
            "   </fo:flow>"
            " </fo:page-sequence>"
            "</fo:root>", "C:\\HelloWorld.pdf");
    }
    catch (_com_error& e)
    {
        //Report any errors that may occur
    }

    ::CoUninitialize();
    return 0;
}
```



The complete source code for this example is located in the installation folder, under Samples/COM+/HelloWorld.

The important points of this program are the following:

- XFCom Type Library is imported using *#import* directive. This is a feature of Microsoft Visual C++ compiler (version 6.0 and up) that will generate C++ wrapper classes for the types described in a TLB.
- COM is initialized using *CoInitialize*.
- A XFRenderer COM+ object is created.
- We call *render*, supplying an XSL-FO text and an output file name.

Remarks

When rendering in a file, the output format can be derived from the file's extension (in this case is PDF) if it is not specified explicitly using *outputFormat* property.

The method *render* accepts a string or IStream object as input. For output, use a string (denoting a file name), IStream or IResponse. There is also another method called *renderUrl* if the document you want converted to PDF resides in a file.

The following code shows how to instantiate and call the engine from Visual Basic:

```
Sub Main()
    Dim engine
    Set engine = CreateObject("XFCom.XFRenderer")

    engine.Render "<fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'>" & _
        " <fo:layout-master-set>" & _
        "   <fo:simple-page-master master-name='all-pages'>" & _
        "     <fo:region-body region-name='xsl-region-body' />" & _
        "   </fo:simple-page-master>" & _
```

```
    " </fo:layout-master-set>" &
    " <fo:page-sequence master-reference='all-pages'>" & _
    "   <fo:flow flow-name='xsl-region-body'>" & _
    "     <fo:block>Hello World!</fo:block>" & _
    "   </fo:flow>" &
    " </fo:page-sequence>" &
    "</fo:root>", "C:\\HelloWorld.pdf"
```

End Sub



The complete source code for this example is located in the installation folder, under Samples/VB/HelloWorld.

Web Access

The following sample describes how to render PDF documents directly into IIS (Internet Information Server) output stream.

```
<%@ Language=VBScript %>
<%
Set engine = CreateObject("XFCom.XFRenderer")
engine.OutputFormat = 5 'XOF PDF
engine.Render "<fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'>" & _
    "<fo:layout-master-set>" & _
    "<fo:simple-page-master master-name='all-pages'>" & _
    "<fo:region-body region-name='xsl-region-body' margin='0.7in'/>" & _
    "</fo:simple-page-master>" & _
    "</fo:layout-master-set>" & _
    "<fo:page-sequence master-reference='all-pages'>" & _
    "<fo:flow flow-name='xsl-region-body'>" & _
    "<fo:block>Hello World!</fo:block>" & _
    "</fo:flow>" & _
    "</fo:page-sequence>" & _
    "</fo:root>", Response
%>
```



The complete source code for this example is located in the installation folder, under Samples/ASP/HelloWord.asp.

The important points of this program are the following:

- A XFRenderer COM+ object is created.
- We have to set the output format explicitly.
- Call *render* passing the ASP's intrinsic *Response* objects as the second parameter.

Remarks

Other output formats are possible as well, including Postscript, HTML, PNG, JPG, etc. For the complete list, please see [XF Rendering Server Programmers Reference](#).

XSL Transformations

One typical usage scenario is when you have a stylesheet (XSL) file, and you want to use it to transform input XML into XSL-FO or XCHART and then generate PDF documents or JPG images.

In the example below, the stylesheet is hardcoded in C++ for demonstration purposes. In real life you probably have the stylesheet in a file. **XF Designer 2004** can be used to test transformation results and preview the generated PDFs (just open the XML, and hit F5).

```
#import <XFCom.tlb> no_namespace
#import <msxml3.dll> rename_namespace("XML")

int main(int argc, char* argv[])
{
    ::CoInitialize(NULL);

    try
    {
        XML::IXMLDOMDocumentPtr xml(__uuidof(XML::FreeThreadedDOMDocument));
        XML::IXMLDOMDocumentPtr xsl(__uuidof(XML::FreeThreadedDOMDocument));

        xml->loadXML("<person name='Joe Doe' age='65'/>");

        xsl->loadXML("<xsl:stylesheet version='1.0' "
            " xmlns:xsl='http://www.w3.org/1999/XSL/Transform'"
            " <xsl:template match='/'>"
            "   <fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'"
            "     <fo:layout-master-set"
            "       <fo:simple-page-master master-name='all-pages'"
            "         <fo:region-body region-name='xsl-region-body' />"
            "       </fo:simple-page-master"
            "     </fo:layout-master-set"
            "     <fo:page-sequence master-reference='all-pages'"
            "       <fo:flow flow-name='xsl-region-body'"
            "         <fo:block>Name: "
            "           <xsl:value-of select='person/@name' />"
            "         </fo:block"
            "         <fo:block>Age: "
            "           <xsl:value-of select='person/@age' />"
            "         </fo:block"
            "       </fo:flow"
            "     </fo:page-sequence"
            "   </fo:root"
            " </xsl:template"
            ">";

        variant_t xslFo = Transform(xml, xsl);

        // Create a new instance of the engine
        IXFRendererPtr engine(__uuidof(XFRenderer));

        // Render transformed data and place the output into a file.
        engine->render(xslFo, "C:\\XSLTransformations.pdf");
    }
    catch (_com_error& e)
    { // Report any errors that may occur
    }

    ::CoUninitialize();
    return 0;
}

variant_t Transform(XML::IXMLDOMDocument* xml, XML::IXMLDOMDocument* xsl)
{
    // Compile the stylesheet into a XSL template.
    XML::IXSLTemplatePtr xslTemplate(__uuidof(XML::XSLTemplate));

    xslTemplate->stylesheet = xsl;

    // Tranform the xml using the compiled stylesheet.
}
```

```
XML::IXSLProcessorPtr processor = xslTemplate->createProcessor();

processor->input = (IUnknown*)xml;
processor->transform();
return processor->output;
}
```



The complete source code for this example is located in the installation folder, under Samples/COM+/XSLTransformations.

The important points of this program are the following:

- In addition to XFCOM.tlb, we have to import msxml3.dll.
- First, we create and load two documents: the input XML and the stylesheet.
- Transform the xml data using the stylesheet.
- Use the transformation result to generate a PDF document.

Remarks

You should cache the *IXSLTemplate* objects and reuse them. This object is free-threaded and stateless, so it can be stored in shared Active Server Pages (ASP) application state.

The same program in VB language:

```
Sub main()

Set xml = CreateObject("MSXML2.FreeThreadedDOMDocument")
Set xsl = CreateObject("MSXML2.FreeThreadedDOMDocument")

xml.loadXML "<person name='Joe Doe' age='65' />"

xsl.loadXML "<xsl:stylesheet version='1.0' "
" xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>" & _
" <xsl:template match='/'>" & _
" <fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'>" & _
" <fo:layout-master-set>" & _
" <fo:simple-page-master master-name='all-pages'>" & _
" <fo:region-body region-name='xsl-region-body' />" & _
" </fo:simple-page-master>" & _
" </fo:layout-master-set>" & _
" <fo:page-sequence master-reference='all-pages'>" & _
" <fo:flow flow-name='xsl-region-body'>" & _
" <fo:block>Name: " & _
" <xsl:value-of select='person/@name' />" & _
" </fo:block>" & _
" <fo:block>Age: " & _
" <xsl:value-of select='person/@age' />" & _
" </fo:block>" & _
" </fo:flow>" & _
" </fo:page-sequence>" & _
" </fo:root>" & _
" </xsl:template>" & _
"</xsl:stylesheet>"

Set template = CreateObject("MSXML2.XSLTemplate")
template.stylesheet = xsl
Set processor = template.createProcessor()
processor.input = xml
processor.Transform

Set engine = CreateObject("XFCOM.XFRenderer")
engine.Render processor.output, "C:\XSLTransformations.pdf"

End Sub
```



The complete source code for this example is located in the installation folder, under Samples/VB/XSLTransformations.

In Memory Operations

In memory only operations are useful in certain special situations like storing the generated PDFs in a database BLOB, performing additional postprocessing, etc.

The example below shows one way to create an in memory stream and then use it as output destination for the rendering process.

```
#import <XFCom.tlb> no_namespace

int main(int argc, char* argv[])
{
    ::CoInitialize(NULL);

    try
    {
        IStreamPtr      stream = CreateMemoryStream();

        IXFRendererPtr  engine(__uuidof(XFRenderer));

        engine->outputFormat = XOF_PDF;
        engine->render("<fo:root xmlns:fo='http://www.w3.org/1999/XSL/Format'>"
            "  <fo:layout-master-set>"
            "    <fo:simple-page-master master-name='all-pages'>"
            "      <fo:region-body region-name='xsl-region-body' />"
            "    </fo:simple-page-master>"
            "  </fo:layout-master-set>"
            "  <fo:page-sequence master-reference='all-pages'>"
            "    <fo:flow flow-name='xsl-region-body'>"
            "      <fo:block>Hello World!</fo:block>"
            "    </fo:flow>"
            "  </fo:page-sequence>"
            "</fo:root>", (IUnknown*)stream);

        // Do something usefull with the stream
        // ...
    }
    catch (_com_error& e)
    { // Report any errors that may occur
    }

    ::CoUninitialize();
    return 0;
}

IStreamPtr CreateMemoryStream(void)
{
    HRESULT  hr;
    IStreamPtr  stream;

    if (FAILED(hr = CreateStreamOnHGlobal(NULL, TRUE, &stream)))
        _com_issue_error(hr);

    return stream;
}
```



The complete source code for this example is located in the installation folder, under Samples/COM+/InMemoryRendering.

The important points of this program are the following:

- We create a memory stream using `CreateStreamOnHGlobal` Windows API function. The memory will be allocated incrementally, as more data is added to this stream, and it will be freed when the `IStream` object is released.
- Output format must be set before calling `render`.
- Call `render` passing the stream as the second parameter.

Remarks

Be aware that using this feature for large documents (>1MB) can result in poor performance if the sever doesn't have enough memory - the operating system will swap memory pages to disk in order to accommodate your memory requests.