

So was there an answer that addressed the root workflow problems and still offered a viable alternative to the status quo? Fortunately, XML (eXtensible Markup Language) seemed to address precisely these problems.

But what is XML? Here's a definition from the W3C Web site (www.w3c.org):

XML is a set of rules (you may also think of them as guidelines or conventions) for designing text formats that let you structure your data. XML is not a programming language, and you don't have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous.

"Structure your data." That sounded like a step in the right direction. But let's stop for a moment to examine what "structured data" means within an XML context. Put more simply:

XML is a markup language for documents containing structured information.

"Markup" means that all text within a document is enclosed within tags that describe that content. Most people are familiar with markup through HTML where an `<h1></h1>` tag indicates a first-level heading and `<p></p>` indicates a new paragraph. However, notice that those two sample HTML tags do not really describe the data enclosed within them. They identify the enclosed data as headings or paragraphs, but do not describe the data itself. Is it a chapter heading? A paragraph within a list? While it tells a Web browser (such as Netscape or Internet Explorer) how to display the data, it does not define the type of data it contains. The same `<h1></h1>` tag can be used for the title of a book, the author, and the chapter titles. These are three very different types of data, but nothing within the HTML tags differentiates them.

XML employs markup in a fundamentally different way. If used properly, XML tags do not just identify data, they describe it. For example, a book title would be tagged with a `<title></title>` tag within a `<book></book>` tag. The author would be tagged within a `<author></author>` tag and even more precisely with `<firstname></firstname><lastname></lastname>` tags. The chapter title would be tagged with a `<title></title>` tag within a `<chapter></chapter>` tag. Notice that unlike HTML, the XML tags do not contain any information about how the data will be displayed, thus making it possible to separate content from format. XML describes the data more thoroughly in a richly structured document. Figure 1 provides a simple example of a structured XML document. Notice that the example contains no formatting information; that will be provided in a linked stylesheet file that contains instructions on how to display the XML content.

Figure 1. Sample XML file (all code simplified for example)

```
<?xml version="1.0" ?> <!-- Specify the media type and the corresponding stylesheet --> <?xml-stylesheet
href="gateways_ss.xml" type="text/xml" media="netscape"?> <?xml-stylesheet href="gateways_ss_ie.xml"
type="text/xml" media="explorer"?> <?xml-stylesheet href="gateways_ss_lynx.xml" type="text/xml" media="lynx"?>
<book> <title>Opening Gateways:</title> <title>A Practical Guide for</title> <title>Designing Electronic Records
Access Programs</title> <para> <bookinfo> <authorgroup> <author> <firstname>Theresa A.</firstname>
<lastname>Pardo</lastname> </author> <author> <firstname>Sharon S.</firstname>
<lastname>Dawes</lastname> </author> <author> <firstname>Anthony M.</firstname>
<lastname>Cresswell</lastname> </author> </authorgroup> <pubdate>December 2000</pubdate> <address>
Center for Technology in Government University at Albany, SUNY <street>1535 Western Avenue</street>
<city>Albany</city> <state>NY</state> <postcode>12203</postcode> <phone>(518) 442-3892</phone>
<fax>(518) 442-3886</fax> <email>info@ctg.albany.edu</email>
<otheraddr>http://www.ctg.albany.edu</otheraddr> </address> <copyright> <year>2002</year>
<year>2000</year> <holder>Center for Technology in Government</holder> </copyright> <legalnotice>
<para>The Center grants permission to reprint this document provided this cover page is included.</para>
</legalnotice> <contractsponsor> This material is based upon work supported in part by the National Historical
Publications and Records Commission under Grant No. 98027. </contractsponsor> </bookinfo> </book>
```

Now that you have a nicely structured document, how can you display it if there's nothing in the XML file to do that? Well, actually there is. As stated previously in the definition, XML is not a programming language; it's a set of rules for designing text formats to structure data. And XML is not just one thing; it is defined by several related specifications, including:

- **XML** (eXtensible Markup Language) which defines the syntax of XML (as seen in the sample in Figure 1)
- **XSL** (eXtensible Stylesheet Language) which is a language for expressing stylesheets. An XSL stylesheet is a file that describes how to display an XML document of a given type. It consists of three parts:

- **XSLT** (Transformations), a language for transforming one XML document into another. It is used, for example, to generate HTML web pages from XML data.
- **XPath**, (XML Path Language), a language used by XSLT to access or refer to parts of an XML document.
- **XSL-FO** (Formatting Objects), an XML vocabulary for specifying formatting semantics and advanced styling features. It is used to produce a PDF document from an XML file, for example.

In short, XSL enables you to take your structured XML data and selectively format it for your various presentation possibilities.

Figure 2 contains a sample XSL file (simplified) that takes the content from the sample XML file in Figure 1 and prepares it for presentation as an HTML page, as shown in Figure 3. Note how the XSL file references the tags from the XML file (title, author, year, etc.) to retrieve the actual content and gives instructions on how to present the content within those XML tags as an HTML page (<h2 align="center">).

Now we have one place to create, control, and maintain our content (XML file) and another mechanism to present that content (XSL file). We have effectively separated content from style and begun to address our root problem. By comparison, HTML tags contain both content and style information (such as <h2></h2> which identifies a second-level heading and defines how to present it) or contain just style information (such as which instructs a browser to show enclosed text as bold).

Figure 2. Sample XSL File (all code simplified for example)

```
<?xml version="1.0"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <!--  
Specify the XHTML layout for the root tag --> <xsl:template match="book"> <html> <head> </head> <body> <!--  
Specify to look for the styling for title element--> <xsl:apply-templates select="title"/> <br/><br/>  
<xsl:apply-templates select="author"/> <xsl:apply-templates select="year"/> <br/><br/> <p align="center">  
<xsl:apply-templates select="etc"/> </p> <xsl:apply-templates select="footer1"/> <br/> <xsl:apply-templates  
select="footer2"/> </body> </html> </xsl:template> <!-- Specify the xhtml style for the 'title' in xml file -->  
<xsl:template match="title"> <h2 align="center"><xsl:value-of select="."/></h2> </xsl:template> <!-- Specify the  
xhtml style for the other templates --> <xsl:template match="author"> <h3 align="center"><xsl:value-of  
select="."/></h3> </xsl:template> <xsl:template match="year"> <h4 align="center"><xsl:value-of select="."/></h4>  
</xsl:template> <xsl:template match="etc"> <font align="center" size="2pt"><xsl:value-of select="."/> <br/></font>  
</xsl:template> <xsl:template match="footer1"> <h5 align="center"><xsl:value-of select="."/></h5>  
</xsl:template> <xsl:template match="footer2"> <h5 align="center"><xsl:value-of select="."/></h5>  
</xsl:template> </xsl:stylesheet>
```

Figure 3. Formatted Page Produced from Sample XML File using XSL