XML—Coming to an Application Near You

By Greg Flurry

What is XML? How does XML relate to HTML? What are DTD, XSL, OSD, CDF, WIDL, and XLL? How does XML fit into the future of network computing? This article answers these questions and more.

The eXtensible Markup Language (XML) is a new kind of markup language from the World Wide Web Consortium (W3C). It represents a much simplified subset of Standard Generalized Markup Language (SGML) ISO 8897, a standard for defining documents used for the last decade. The W3C also defines HyperText Markup Language (HTML). Since HTML is also derived from SGML, how do XML and HTML differ?

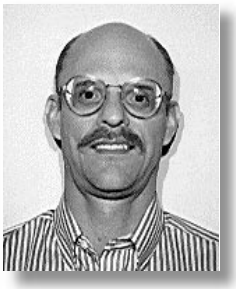
HTML, a key part of today's network computing, is a simple markup language for describing the format of small and simple documents containing text and other media. HTML, as defined by the W3C, uses a fixed vocabulary of tags to describe the document or data. Although a fixed set of tags make it easier to apply HTML, they also limit extensibility, the ability to support rich information structures, and the ability to determine syntactic and structural validity.

To address these issues, the W3C designed XML to fill a much broader role, which is evident in the design goals from the XML specification (see *http://www.w3.org/TR/REC-xml* or *http://www.xml.com/axml/testaxml.htm*):

- XML should be straightforward and usable on the Internet.
- XML should support a wide variety of applications.
- XML should be compatible with SGML.
- Programs that process XML documents should be easy to write.
- The number of optional features in XML should be kept to the absolute minimum.
- The design of XML should be formal and concise.
- XML documents should be easy to create.

To meet these goals, the W3C defined XML as a *meta-language*—a language used to describe other languages. As such, XML allows the definition of custom vocabularies to describe the abstract structure of a document or data.

XML also overcomes the three main limitations of HTML. First, XML supports extensibility. The ability to define custom tags enables its use in a much wider range of applications and with a broad range of clients including RS/6000s and PCs, and servers, such as RS/6000®, AS/400®, S/390®, Netfinity®, and others. Second, XML supports complex data structures that can be nested to any level of complexity. In addition, these structures can be efficiently created, consumed, manipulated, searched, and interchanged. Finally, XML, because of its



Greg Flurry

```
<H3>Moe Howard</H3>
<UL>
  <LI>address</LI>
    <ADDRESS>1212 Nevada
Street</ADDRESS>
    <ADDRESS>Podunk, CA</ADDRESS>
  <LI>phones</LI>
    <UL>
      <LI>555-555-5555</LI>
      <LI>555-555-4444</LI>
    </UL>
</UL>
```

Figure 1. HTML address book entry example

```
<person first="Moe" last="Howard">
  <address>
   <street>1212 Nevada
     Street</street>
   <city>Podunk</city>
   <state>CA</state>
  </address>
  <phones>
   <work>555-555-5555</work>
   <home>555-555-4444</home>
  </phones>
</person>
```

Figure 2. XML address book entry example

crisp rules for defining documents, supports the validation of a document's syntax and structure. These three enhancements have created much excitement about XML. In short, XML can do almost anything HTML can do, plus much more.

The next section presents a simple example that clarifies some XML advantages. Later sections describe how XML can define additional document types for new or existing applications, some of the current or proposed uses for XML, and activities and tools required for complete solutions.

An HTML versus XML Address Book

Figure 1 shows an example HTML used to describe an entry for an address book that might be found in Lotus Notes® or on an IBM WorkPad™. Figure 2 shows the XML equivalent.

Both HTML and XML use tags for markup. Tags in the HTML version can only identify the expected format. More abstract information, such as the person's state, typically would not be part of the tag, but would be part of the content of the tag. This abstract information could also perhaps be assumed from positioning within the `<h3>` tag. The tags in the XML markup, however, clearly identify the pieces of the address.

Consider which example—HTML or XML—is easier to process. Suppose an application needed to identify all the people from California within an address book. The application in the HTML implementation must search the entire entry, because only de facto rules exist about the structure of an entry. Additionally, there is no way to guarantee that Moe Howard is from Nevada or California, since both are valid states. The XML implementation clearly identifies the address and the state, making the search much easier and more accurate.

XML Advantages

XML has significant advantages for today's network computing applications, given the possibility of limited bandwidth and clients of variable capabilities. Assume that an XML-based address book database is stored on a server, such as an RS/6000 (see Figure 2). Then suppose the owner wants to access the address book from a wired network-attached PC, from a wireless network-attached device such as an IBM WorkPad, from an intelligent telephone and a regular (dumb) telephone. See Figure 3.

The server can send the same data to the first three devices. A browser on the PC can render the XML directly to Windows™ display orders, or convert first to HTML to leverage the existing browser HTML renderer. The WorkPad can convert the XML directly to its own unique display orders. An intelligent telephone can convert the XML to speech. For the regular telephone, the server itself, or more likely some intermediary, can convert the XML to speech.

The key point in this example is that the content is the same in all cases; only the rendering is different. XML's ability to

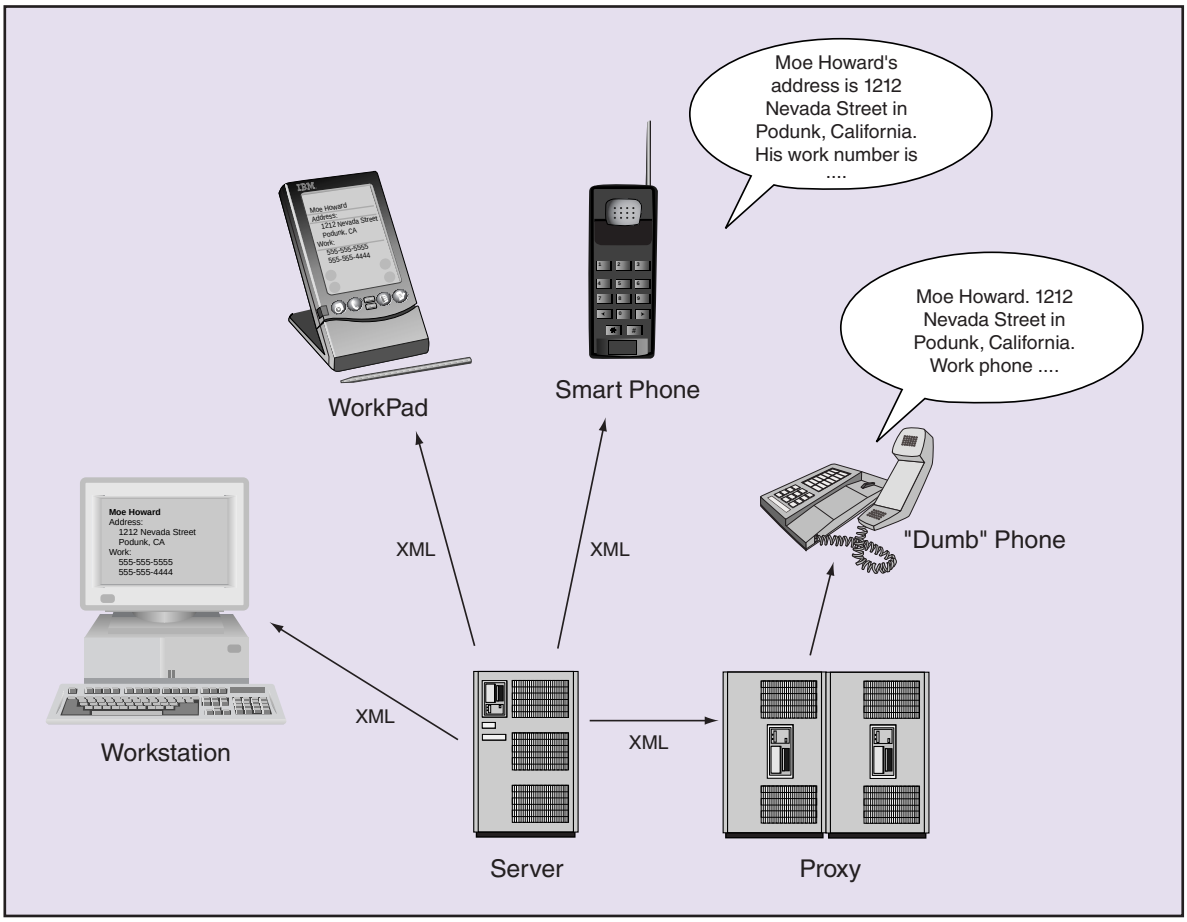


Figure 3. Accessing the address book

explicitly define the structure of the address book entry makes this possible.

Using XML for New Applications

The question arises about how to use XML in a new application. The first step is to define a new XML dialect or document type.

A new XML document type is defined using yet another concept from SGML called the Document Type Definition (DTD). The DTD defines the vocabulary and grammar for an XML-compliant document, that is, the set of tags that can appear in a document and how the tags relate to each other. This aspect of XML provides the extensibility and the ability to describe complex data structures. The DTD and additional rules defining XML also provide XML's ability to validate or prove the "correctness" of XML document syntax and structure.

Formally, an XML document contains a prolog and a body. The prolog contains XML version and character encoding information, the DTD, and miscellaneous markup; all parts of the prolog are optional. The body contains elements, entity references, and miscellaneous markup.

An XML document has both a logical and a physical structure. Elements, which describe the valid tags in a document, define the logical structure. Elements also have attributes that provide additional information about the elements. Entities define the physical structure of an XML document.

The DTD contains declarations that define the elements, element attributes, and entities. The DTD also contains notation declarations, conditional section declarations, and miscellaneous markup. Finally, miscellaneous markup, which is optional no matter where it appears, consists of com-

ments and processing instructions.

Parts of an XML Document

This section offers additional definitions and examples of the various XML document parts.

Element type declarations: Contain the name of the element type and a content specification for the data that can appear within that element. When used for markup, the elements are identified by start and end *tags*. For instance, to define the address element in the above example that contains street, city, and state elements, use the following:

```
<!ELEMENT address (street, city, state)>
```

Attribute declarations: Declared by associating the element name with an attribute list. Each attribute consists of an attribute name, attribute type, and default value. For example, the "person" element from Figure 4 has attributes "first" and "last" defining the name.

Elements and attributes: Used as described above:

```
<!ATTLIST person
  first CDATA #REQUIRED
  last  CDATA #REQUIRED>
```

Figure 4. Element attributes

```
<person first="Moe" last="Howard" ...
</person>
```

Entities: Provide "macro" type capability for XML. A full description of entities is beyond the scope of this article, but a few examples give a hint of their usefulness. For example, suppose you want to use the phrase "Howard, Howard, Fine & Sons" in many places within a document. The document could contain the following internal general entity declaration:

```
<!ENTITY theCompany "Howard, Howard,
Fine & Sons">
```

A document could use "`&theCompany;`" as a representation for the phrase. Next, suppose that a DTD contains several element type declarations that have the same content specification. It is possible to define an internal parameter entity as follows:

```
<!ENTITY  % genContentSpec "#PCDATA |
link | image">
```

and use it as follows:

```
<!ENTITY xyz (%genContentSpec)>
<!ENTITY mno (%genContentSpec)>
```

A final example shows additional flexibility of XML. Entities also can define the physical structure of a document. For example, an external general entity might point to some set of boilerplate for a business that is used in many documents; therefore it is kept in an external file. The entity declaration is shown in Figure 5 and the entity is referenced using "`&boilerplate;`".

```
<!ENTITY boilerPlate
      SYSTEM
"http://www.hhfs.com/internal/boilerplate.xml">
```

Figure 5. Declaration referencing frequently used entity

Notation declarations: Identify non-XML content, which allows XML parsers or applications that cannot handle such content to pass it to the appropriate processor.

Conditional sections: Allow XML parsers to include or ignore such sections depending on a fixed keyword or a parameter.

Processing instructions: Identify directions and information for XML parsers and applications. Processing instructions are not considered part of a document, but are passed on to an application identified in the declaration. The following is an example:

```
<?XML> version= "1.0"?>
```

Comments: Intended primarily as notes for a person to read and not considered part of a document. Comments are declared as follows:

```
<!– this is a comment –>
```

XML documents come in two major flavors: well-formed and valid. A well-formed XML document has the following characteristics that make the document easy to process:

- All beginning tags have matching end tags; for example, <person> ... </person> are properly nested; that is, <person> ... <address>...</address>... </person>, not <person> ... <address>... </person>...</address>.
- Tags with no content use the special XML syntax; for example, a page break could be indicated with a <pagebreak/> tag.
- All attributes are correctly quoted as in the <person first="Moe"> tag.
- All entities are declared.

```
<?xml version="1.0"?>
<!– address book –>
<!DOCTYPE addressbook SYSTEM
"addressbook.dtd">
<addressbook>
<person first="Moe" last="Howard">
  <address>
    <street>1212 Nevada
Street</street>
    <city>Podunk</city>
    <state>CA</state>
  </address>
  <phones>
    <work>555-555-5555</work>
    <home>555-555-4444</home>
  </phones>
</person>

<!– here could be other persons –>
</addressbook>
```

Figure 6. A valid address book example

Being well-formed implies nothing about having a DTD. It simply means that the document is syntactically correct. For example, the <person> element might require both a first and last parameter, but that could not be detected in a well-formed document. A valid XML document does contain or reference a DTD and complies with all the constraints expressed by the declarations in the DTD. XML allows a document to contain a DTD in a prolog, or reference an external DTD or DTDs. Using a set of DTDs, an XML parser or application can determine if a document is not only well-formed, but also logically correct, that is, valid.

Figure 6 represents a more complete example of the address book from Figure 2. The example above was well-formed, but not valid; the version in Figure 6 is valid.

The XML document in Figure 6 indicates that the document uses XML Version 1.0 and references a DTD in the file called addressbook.dtd. The rest of the markup is the same as in Figure 2. Figure 7 shows the DTD for the address book document type.

The address book DTD says that the <addressbook> element consists of one or more (+) <person> elements. The attributes of the <person> are the first and last name. Each <person> element contains one

```
<?xml encoding="US-ASCII"?>
<!– dtd for the address book –>
<!ELEMENT addressbook (person)+>
<!ELEMENT person (address, phones)>
<!ATTLIST person
  first CDATA #REQUIRED
  last  CDATA #REQUIRED>
<!ELEMENT address (street, city,
state)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT state (#PCDATA)>
<!ELEMENT phones (work*,home*)>
  <!ELEMENT work (#PCDATA)>
  <!ELEMENT home (#PCDATA)>
```

Figure 7. DTD for the address book

<address> and one <phones> element. An <address> element contains one each of <street>, <city>, and <state> elements. The <phones> element contains zero or more (*) <work> phone numbers and zero or more <home> phone numbers.

For additional details about XML documents, see the XML 1.0 specification by visiting the following Web sites: *http://www.w3.org/TR/REC-xml* or *http://www.xml.com/axml/testaxml.htm*.

Uses of XML in Applications

XML is useful and relevant in many applications. More than 20 XML dialects have been proposed or are currently in use, and the number is growing daily. The following list represents some selected uses of XML:

1. The Channel Definition Format (CDF), created by Microsoft® in partnership with DataChannel™, describes Web-based "channels" used for application and data distribution. CDF is built into Internet Explorer 4.0. See *http://www.w3.org/Submission/1997/2/*.

2. The Open Software Description (OSD), related to CDF, was defined by Marimba™ and Microsoft. The objective of OSD is to describe software packages in a way that makes it possible to distribute applications on a timely basis with a minimum of user involvement. See *http://www.w3.org/TR/NOTE-OSD*.

3. The Chemical Markup Language (CML) and the Mathematical Markup Language (MML) have proved effective at representing the arcane symbols, complex characters, and demanding markup that chemists and mathematicians require to disseminate important ideas. See *http://www.venus.co.uk/omf/cml/doc/* and *http://www.w3.org/TR/WD-math/*.

4. The Resource Description Framework (RDF) is a foundation for processing metadata that describes different types of Web resources. RDF represents the W3C's attempt to unify three separate, but similar, XML efforts:
   - W3C's work on describing data using the Platform for Internet Content Selection (PICS) dialect
   - Netscape's™ Meta Content Framework (MCF) for describing Web pages
   - Microsoft's XML-Data, intended for representing schema

   See *http://www.w3.org/RDF/Overview.html*.

5. The Extensible Linking Language (XLL) defines a rich syntax and semantics for hyperlinks. XLL promises to expand the capabilities inherent in hypertext well above and beyond current HTML implementations. It defines mechanisms such as bi-directional links, multi-way links, aggregate links (for multiple sources), and link types (links with attributes). See *http://www.w3.org/TR/WD-xml-link*.

6. The Synchronized Multimedia Interface Definition Language (SMIL) is a recent effort intended to help content developers better synchronize media for Web delivery, without having to depend upon specialized and expensive tools. See *http://www.whatis.com/smil.htm*.

7. Open Financial Exchange (OFX), a collaborative effort among Microsoft, Intuit®, and CheckFree®, is intended "for the electronic exchange of financial data between financial institutions, business, and consumers." Applications such as Microsoft Money and Intuit Quicken® use OFX. Although currently SGML based, the collaborators are converting OFX to XML so XML-capable Web browsers can use it. See *http://www.ofx.net/ofx/default.asp*.

8. OpenTag™ from the International Language Engineering® Corporation (ILE®) is designed to permit a single XML document to deliver content in different languages. International companies wanting to conduct e-business outside the English-speaking world obviously have a great interest in OpenTag. See *http://www.opentag.org/otdownld.htm*.

9. The Web Interface Definition Language (WIDL) from webMethods, but submitted to the W3C, is targeted at Web automation. For example, WIDL helps integrate information retrieved from the Web and disseminate that information throughout all business processes of an enterprise. See *http://www.webmethods.com*.

10. The Wireless Markup Language (WML) is part of the Wireless Application Protocol. WML is intended for use with narrow band devices, such as cellular phones and pagers, to specify content and user interface. See *http://www.wapforum.org/docs/technical/wml-30-apr-98.pdf*.

11. The Real Estate Listing Markup Language (RELML) is a relatively new dialect from OpenMLS and 4thWORLD Telecom. The purpose of RELML is to describe real estate listings that can be distributed across the Web, but still allow searching of the distributed listings that match a customer's criteria. See *http://www.xml.com/xml/pub/98/08/real/tech.html* and *http://www.4thworldtele.com/public/Client.html*.

You can find additional information about XML dialects and XML in general by visiting *http://www.xmlinfo.com* and *http://www.xml.com* on the Web.

What More is Needed?

The definition of an XML document does not describe how a document might be formatted when it is either displayed, printed, or otherwise rendered. Today, most active developments use the Cascading Style Sheets (CSS1 and CSS2) defined by the W3C and used with HTML documents.

Efforts are under way to define the Extensible Style Language (XSL) as a more dynamic and powerful notation for defining document style. Just as XML is derived from SGML, the work on XSL is derived, in large part, from the Document Style, Semantics, and Specification Language (DSSSL) used in the SGML community. See *http://www.w3.org/Submission/1997/13/ArborText*.

An application must be able to process the content of an XML document, otherwise it has no value. Each application can process a document in its own unique way. However, XML eliminates this chaos. The rules for the construction of well-formed and valid XML documents bring a regularity to such documents. In recognition of this, the W3C is developing the Document Object Model (DOM), "a platform- and language-neutral program interface that will allow programs and scripts to access every element in a document and update the content and structure of documents in the standard way." See *http://www.w3.org/DOM*.

XML parsers are one of the key targets for DOM. Parsers are tools that can determine whether an XML document is valid or well-formed, and produce normalized versions of the document for easier processing by applications. A good example of such a parser is XML4J, a Java™-based parser from IBM. Since XML4J is written in Java, it can run without modification on AIX® as well as other operating systems. XML4J is freely available on the IBM alphaWorks™ Web site. See *http://www.alphaWorks.ibm.com/formula.nsf/system/technologies/7BC35F3E4E69996A882565A700035C56*.

XML Tools

One strong indicator of an important technology is tools for using the technology. XML tools exist and more are coming. The major Web browsers (Netscape Communicator 5.0 and Microsoft Internet Explorer 4.0) support or will support various uses of XML. Because of its SGML heritage, XML can be handled by hundreds of existing SGML tools, such as GRIF Symposia and SP. Some existing authoring tools like FrontPage and HomeSite also support XML, and others are expected to incorporate support rapidly. For information about XML tools, see *http://www.sil.org/sgml/*, *http://www.infotek.no/sgmltool/guide.html*, *http://www.infotek.no/sgmltolguide.htm* and *http://www.xmlinfo.com*.

What Does It All Mean?

From the beginning, XML was designed to serve a wide range of network computing

applications. XML certainly overcomes the disadvantages of HTML. The sample of XML dialects above show that XML can serve in almost any application, extending far beyond HTML, in roles such as defining complex content and facilitating processing of that content. XML can provide an answer for just about any question concerning document or data processing in network computing applications.

The companies promoting some aspect of XML (IBM, Netscape®, Sun®, Microsoft, and many others) guarantee that XML will have an excellent chance of playing a role in virtually all new network computing applications. While XML will not necessarily replace HTML, XML's power, flexibility, extensibility, and openness should make it a major factor in the network computing environment of the next decade. XML will come to an application near you—and soon.

Additional References

Leventhal, Michael; Lewis, David; and Fuchs, Matthew. *Designing XML Internet Applications*. Upper Saddle River, NJ: Prentice-Hall, Inc. 1998. ISBN 0-13-616822-1.

Megginson, David. *Structuring XML Documents*. Upper Saddle River, NJ: Prentice-Hall, Inc. 1998. ISBN 0-13-642299-3.

Tittle, Ed; Mikula, Norbert; and Chandak, Ramesh. *XML for Dummies*; Foster City, CA: IDG Books Worldwide, Inc. ISBN 0-7645-0360-X.

Greg Flurry, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Flurry is a senior technical staff member in the Server Group Division. His responsibilities, as part of the Systems Architecture & Technical Strategy team, include network computing and Java. He has a BS in Electrical Engineering from Vanderbilt University and an MS in the same field from the University of Kentucky.