XML – A Primer

By Helmut Reimann
Sales Engineering Manager

November, 2005

GUPTA™

Abstract

This white paper introduces the reader to the programming language known as XML.

Introduction

This technical white paper will provide you with a general XML overview. It will familiarize you with the syntax and structure of XML. Additionally, it provides you with the insight into the importance of XML in the future of application development.

What Is XML?

XML (eXtensible Markup Language) is a standard to produce documents in the form of a tree structure and it defines the rules on how to create these types of documents.

With XML it is possible to create every kind of document such as text, pictures, complex structures, etc.

XML splits the data from the application. If an application can read the XML document it could be shown as a graphic or as a Microsoft Excel spreadsheet, or as a table window in a Team Developer application. XML splits the presentation of the data from the data itself but the application type dictates how the document can be manipulated.

The XML document structure is always the same every time.

XML helps independent applications work together because they use the same data. That is the main reason why Enterprise Application Integration (EAI) uses XML for data integration.

Document Object Model (DOM)

Document Object Model (DOM)

DOM is a form of representation for structured documents as an object oriented model. DOM is the official World Wide Web Consortium (W3C) standard for representing structured documents in a platform and language neutral manner. The DOM is also the basis for a wide range of application programming interfaces, some of which are standardized by the W3C.

DOM was initially supported by a web browser to manipulate elements in an HTML document. DOM was a way of dynamically accessing and updating the content, structure and style of documents. Owing to incompatibilities in the DOM implementation between different browsers, the W3C came up with standard specifications for DOM.

DOM puts no restrictions on the document's underlying data structure. A well-structured document can take the free form using DOM. Most XML parsers like Xerces and XSL processors like Xalan have been developed to make use of the tree structure.

Such an implementation requires the entire content of a document be parsed and stored in memory. Hence, DOM is best used for applications where the document elements have to be randomly accessed and manipulated. For XML-based applications which involve a one-time selective read/write per parse, DOM presents a considerable overhead on memory.

DOM level specifications

The current DOM specification is Level 2, but some of the Level 3 specifications are now W3C recommendations.

Level 0:

This includes all the vendor-specific DOMs that existed before creation of DOM Level 1, e.g. document.images, document.forms, document.layers, and document.all.

Note: This is not a formal specification published by the W3C, but rather a reference to what existed before the standardization process.

Level 1:

This involves the navigation of DOM documents and to manipulate content.

Level 2:

This involves XML Nanmespace support, filtered views, and events

Level 3:

Level 3 Consists of six different specifications:

1. The DOM Level 3 Core,
2. The DOM Level 3 Load and Save,
3. The DOM Level 3 XPath,
4. The DOM Level 3 Views and Formatting,
5. DOM Level 3 Requirements,
6. The DOM Level 3 Validation which further enhances the DOM.

A basic XML document

What an XML Documents Looks Like

XML documents can be read by applications or by humans if the document has readable character information.

Sample

```
<?xml version="1.0"?>
<pricelist>
   <entry>
      <product>Team Developer </product>
      <Version>2005.1 for Linux </Version>
   </entry>
</pricelist>
```

Everyone can read this XML document without any XML knowledge.

The sample means that this is a *pricelist* for a *product* and the specified *version* of this product.

An EAI environment utilizes this kind of structure to communicate between different business applications over different operating platforms (e.g. Microsoft Windows and Linux), databases and in the future release of Microsoft Office because all Microsoft Office products will be able to save the data into XML files making communication between platforms, databases and MS-Office products easier and more efficient.

XML Documents Have a Physical and Logical Structure

The physical structure is the file itself and the first line of the document. This first line is the XML declaration. It is an optional line stating what version of XML is in use (normally version 1.0) and may also contain information about character encoding and external dependencies. The logical structure describes the data itself.

The remainder of this document consists of nested elements, some of which have attributes and content. An element typically consists of two

tags: a *start tag* and an *end tag*. And possibly surrounding text and other elements is also included. The start tag consists of a name surrounded by angle brackets, such as <product>; the end tag consists of the same name surrounded by angle brackets but with a forward slash preceding the name, for instance </product>. The element's content is everything that appears between the start tag and the end tag, including text and other (child) elements. The following is a complete XML element, with a start tag, text content, and an end tag:

<step>Knead again, place in a tin, and then bake in the oven.</step>

In addition to content, an element can contain attributes such as name-value pairs which are included in the start tag after the element name. Attribute values must always be quoted, using single or double quotes, and each attribute name should appear only once in any element.

<ingredient amount="3" unit="cups">Flour</ingredient>

In this example, the ingredient element has two attributes: *amount* (having a value of 3) and *units* (having a value of cups). In both cases at the markup level, the names and values of the attributes, like the names and content of the elements, are just textual data. The "3" and "cups" are not a quantity and unit of measure, respectively, but rather they are just character sequences that the document author may be using to represent those items.

In addition to text, elements may contain other elements:

<Instructions>
<step>Mix all ingredients together, and knead thoroughly.</step>
<step>Cover with a cloth, and leave for one hour in warm room.</step>
<step>Knead again, place in a tin, and then bake in the oven.</step>
</Instructions>

In this case, the **Instructions** element contains three step elements. XML requires that elements to be properly nested. Elements may never overlap.

Additionally to the data, you can describe in an XML document what you want to do with this data (*Processing Instruction*).

<?Target-Name Parameter ?>

You can use this syntax if you want to write comments into your document:

<! --- this is a comment ---- >

You can include free text into an XML document by using the CDATA area of the document:

<! [CDATA[This is a free text ,...]] >

Well-Formed Documents

An XML document must be well-formed

An XML document is text-based sequence of characters. The specification requires support for Unicode encodings UTF-8 and UTF-16. The use of other non-Unicode based encodings, such as ISO-8859, is admitted and is indeed widely used and supported.

A well-formed document must conform to a set of rules. The following list contains the most important of those rules:

- One and only one root element exists for the document. However, the XML declaration, processing instructions, and comments can precede the root element.
- Non-empty elements are delimited by both a start-tag and an end-tag.
- Empty elements may be marked with an *empty-element* (*self-closing*) tag, such as <IAmEmpty/>. This is equal to <IAmEmpty></IAmEmpty>.
- All attribute values are quoted, either single (') or double (") quotes. Single quotes close a single quote and double quotes close a double quote.
- Tags may be nested but may not overlap. Each non-root element must be completely contained in another element.
- The document complies with its character set definition. The charset is usually defined in the XML declaration but it can be provided by the transport protocol, such as HTTP. If no charset is defined, usage of a Unicode encoding is assumed, defined by the *Unicode Byte Order Mark*. If the mark does not exist, UTF-8 is the default.

Element names are case-sensitive, so the following example is a well-formed matching pair:

<Step> ... </Step>

The careful choice of names for XML elements will convey the meaning of the data in the markup. This increases human readability while retaining the rigor needed for software parsing.

Choosing meaningful names implies the semantics of elements and attributes to a human reader without reference to external documentation. However, this can lead to verbosity, which complicates authoring and increases file size.

The XML Meta Language / XML Schema

XML schema

XML Schema, published as a W3C Recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve recommendation status by the W3C.

Since other XML Schema languages (such as *Reflaxing*) now exist it is important to cite this language as either XML Schema or W3C XML Schema, and always with the word *Schema* capitalized. An XML Schema instance is known as an XML Schema Definition (XSD) and it typically has the filename extension .xsd. The language itself is sometimes informally referenced as XSD, even though WXS (W3C XML Schema) is the more appropriate initialization.

In its appendix of references, XML Schema acknowledges the influence of DTD and other early XML Schema efforts such as DDML, SOX, XML-Data, and XDR. It appears to have borrowed pieces from each of these proposals, but is also a compromise between them. Of those languages, two are still actively used and developed: XDR and SOX. Their sponsors, Microsoft and Commerce One, respectively, have both announced that they would support XML Schema for their new developments, so W3C XML Schema should become the only surviving member of this family.

After XML Schema-based validation it is possible to express an XML document's structure and content in terms of the data model that was implicit during validation. The XML Schema data model includes:

- The vocabulary (Element/Attribute names),
- The content model (Relationships/Structure),
- And data types.

This collection of information is called the Post-Schema Validation Infoset (PSVI). The PSVI gives a valid XML document its *type* and facilitates treating the document as an object, using object-oriented programming (OOP) paradigms. This particular OOP approach to XML data access was primarily advocated by Microsoft, a major contributor to the development of XML Schema. Converting an XML document to a data type-aware object can be beneficial in some parts of computer software design, but critics contend that it also undermines openness, a key feature of XML, and that it is biased toward compatibility with the data types native to Microsoft's favored programming languages.

The XML Meta language describes the structure of an XML document. Generally there are two Meta languages used: DTD (Document Type Definition) and XML-Schema.

A DTD (Document Type Definition) is a description of an XML document. DTD was standardized with XML. It is not really a strict definition and, more importantly, it has its own language.

Here is a basic sample of XML-Schema:

```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="country">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="population" type="xs:decimal"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

This XML Schema describes the following XML document:

```
<country
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="country.xsd">
 <name>France</name>
 <population>59.7</population>
</country>
```

Now we can see that an XML document has its own XML Schema which can be shared.

Basic XML Technologies

Two main technologies are defined to work with XML documents:

- **SAX** (Simple API for XML) is a standardized possibility to parse XML-documents which is used as a stream of events. The application has to react to these events and all work is done in a sequenced way.
- **DOM** (Document Object Model) loads the whole file into the memory. All data is loaded into an object tree which makes it easier to work in an object-based mode. With Team Developer 2005.1 you can use XML-Serialization based on the XML class to deal directly with the document via User Defined Variables (UDV).

XML Families

XML is only a basic definition of different XML families which are certified by the W3 institution:

- Transformation of XML documents: XSLT
- Addressing from XML tree nodes: xPath
- Standardized attributes: XML Base and xml:id
- Combination of XML resources: XPointer, XLInk and XInclude
- Selection of data out of an XML record: XQuery
- XML data structure: XML Schema or...
- ...XML Meta Language: XSD XML Schema Definition Language
- Signature and encryption of XML nodes: XML Signature and XML Encryption

Binary XML

Binary XML, or Binary eXtensible Markup Language, refers to any specification which attempts to encode an XML document in a binary data format, rather than plain text. While there are several competing formats, none has been widely adopted by a standards organization or accepted as a *de facto* standard.

Binary XML

Using a binary XML format generally reduces the verbosity of XML documents and cost of parsing, while hindering the use of ordinary text editors to view and edit the document. Other advantages include enabling random access and indexing of XML documents.

The major challenge for binary XML is to create a single, widely adopted standard. The W3C has a XML Binary Working Group, while Sun Microsystems is pushing to have Fast Infoset adopted as an ISO standard.

Alternatives to binary XML include using traditional file compression methods on XML documents (for example *gzip*); or using an existing standard such as ASN.1.

Traditional compression methods, however, offer only the advantage of compression, without the advantages of decreased parsing time and do not enable random access. ASN.1 is being used as the basis of Fast Infoset, which is one binary XML standard.

GUPTA Team Developer 2005.1 and XML

GUPTA Team Developer 2005.1 and XML

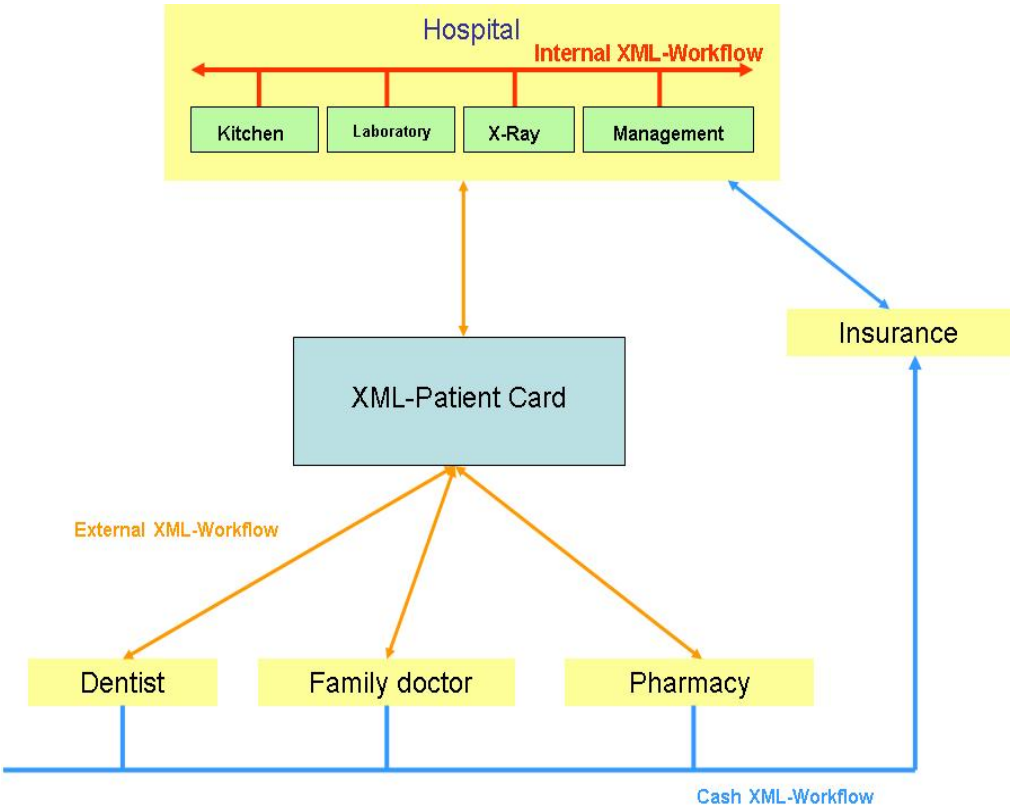
GUPTA Team Developer 2005.1 gives you the possibility to deal with XML documents in a high level fourth generation programming language (4GL).

You can create or read XML Documents (with XML Schema files) by coding only one line of source code. This is a basic functionality of a table window.

With Team Developer 2005.1 it is possible to implement XML serialization & de-serialization. UDV variables can be translated into XML documents and back again. A powerful class library supports developer-to-build EAI (Enterprise Application Integration) solutions with a high level 4GL language.

Sample XML Workflow: Digital Patient Folder

The following diagram shows a sample XML workflow chart entitled *Digital Patient Folder*.



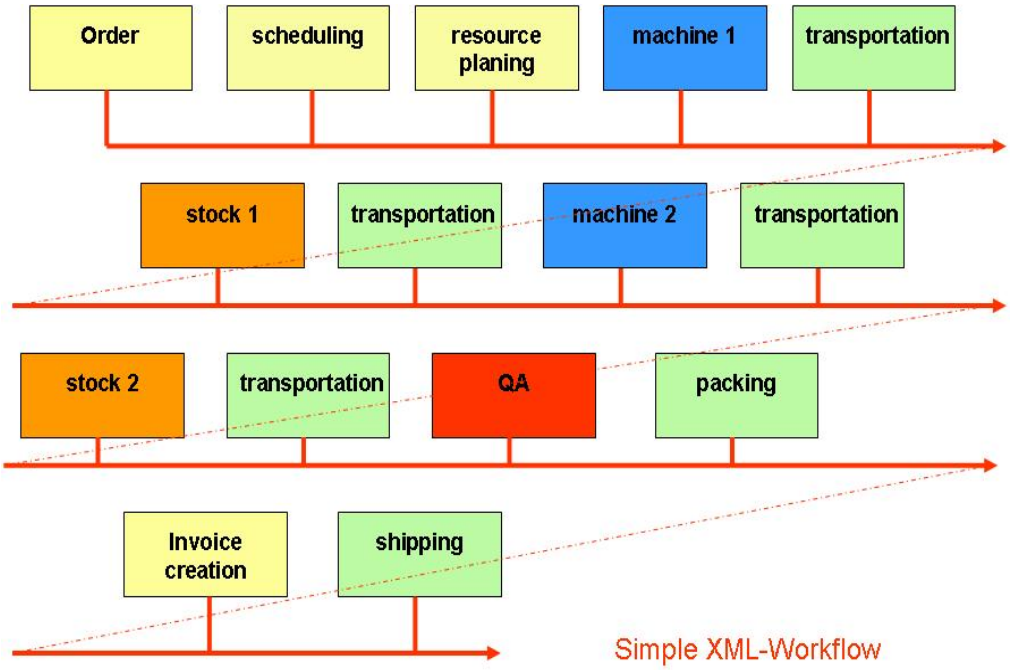
This is a small sample of how XML documents can be used. Every hospital, insurance company or doctor has its own internal IT-structure with different expert systems (e.g. X-Ray department or management). For internal communication XML is used. With each patient there exists a digital XML document and the different departments add their information this document. The XML document flows easily between internal IT systems. In addition, the XML document is tied back to the profit center.

The external workflow is based on a smart card. This card stores all relevant information of a person; such needed medicaments, allergy, conditions and health risks to the patient. Different doctors receive all the information from this card and they can send then requests to another hospital or to colleagues for more information, such as an x-ray image.

The invoice for the insurance company is sent as an electronic XML document. All different doctors, hospitals and insurance firms can communicate with XML because XML transports only the data without the presentation layer.

Sample XML Workflow: Production

The following diagram shows a sample XML workflow chart entitled *Production*.



The above diagram shows an incoming order which opens an XML document. This document receives information from different departments, like Quality Assurance (QA), data entry from a machine, or from different offices. All information is without the presentation layer (for instance Microsoft Office), or without different expert systems like QA software.

Completed documents from the workflow can be used to get a better performance of the manufacturing process because they can read back into the expert systems. Finally, the customer receives all necessary information of his products, like a QA protocol or scheduling information. This helps him to get the right quality and the shipment just in time.

Conclusions

XML will become the newest standard to save and transfer information within a business environment. XML is easy to handle and platform independent. GUPTA gives you the technology to be prepared for the future, to build integrated, modern standard based applications.

You can find more information on XML at:

- www.w3.org/XML
- www.w3schools.com
- www.wikipedia.org