



XML

Backgrounder

TECHNOLOGY AND APPLICATIONS

Content

Introduction	4
XML and Software AG	4
From SGML to HTML to XML	4
The Limitations of HTML	4
SGML - Flexible but Complex	4
From SGML to XML	5
XML - The Simple Meta Language	5
The Shakespeare Example	7
Implementation of XML	7
The Document Type Definition - DTD	7
XML Schema - XSD	7
The eXtensible Stylesheet Language - XSL	7
XPath and XLink	8
XPath - For XML Document Queries	8
The Document Object Model - DOM	8
XQuery	8
XML - A Universal Document Standard	9
Standardization and W3C	9
XML Applications	9
EDI	10
XML and Databases	10
XML and SQL	11
XML and ODBMS	11
More examples of XML applications	12

Introduction

XML AND SOFTWARE AG

Early in 1997, Software AG recognized the importance of XML. Already at that stage we had the vision that the impact of XML would be extensive for any organization and therefore came up with the idea of an XML strategy. Today we have implemented this strategy with Tamino XML Server, offering next-generation XML services.

Tamino

XML Server

Tamino XML Server stores XML documents without converting them to other formats and helps with finding and managing any type of content across the enterprise. Accompanied by a broad range of services, Tamino enables rapid implementation of mission-critical electronic business applications based on XML standards.

It's no longer necessary to explain the meaning of XML to Web experts. After all, the W3C Recommendation for the eXtensible Markup Language is more than four years old - and in the Web world that's a very long time. The situation in the traditional IT world - that of large databases and even larger machines - is quite different. There, people are only just beginning to take note of XML. In our view, however, the long-term benefits of XML are likely to be even greater for IT users than for Web users.

XML is a meta language which can be used to describe the logical structure of a wide variety of documents and data in different ways according

to the application. This universal, flexible and extensible approach opens up an almost unlimited range of uses for XML, from word processing through electronic business to data archiving. XML can probably best be compared with SQL in terms of its impact. For decades this standard also provided a firm common ground on which manufacturers could meet, regardless of their own particular platforms or system limitations. SQL was one of the kingpins on which the IT world hinged as it set out on its overwhelmingly successful campaign. XML will play a similar role in the coming years.

From SGML to HTML to XML

THE LIMITATIONS OF HTML

The history of XML is closely linked to the evolution of the World Wide Web. One of the mainstays of the World Wide Web is HTML (Hypertext Markup Language), which serves to describe the layout of Web pages. HTML is extremely simple to use: the Web designer employs a series of codes (known as 'tags') to define what a document should look like in the browser, create links to other Web pages, integrate any required applets, etc. HTML is a very simple page description language. Its great advantage is that Web pages have much the same appearance everywhere and that they also adapt automatically to different screen sizes. HTML pages can now be generated easily using a graphic editor, and the previously obligatory chore of entering tags manually is now more or less obsolete.

Thanks to the rapid advances in Web technology, however, HTML has now come up against certain limitations:

- HTML does not allow individual elements on a page to be marked up semantically. It is not possible to reproduce specifications of the data structures - something that is essential for databases or object hierarchies, for instance.
- HTML only describes the appearance of documents and cannot cover any content aspects. It is therefore unsuitable for explicit queries.
- HTML is not extensible; it is thus not possible to define your own tags for specific requirements, or to use data formats within documents.

If the requirements that have to be satisfied by a Web page are more complex, they can only be met by indulging in time-consuming programming work, either with applets or with special applications. Finally, the restrictions inherent to HTML have led to a series of proprietary extensions, which in turn have had a negative effect on the universality of HTML-formatted information and thus also of the Web in general.

SGML - FLEXIBLE BUT COMPLEX

The majority of HTML users are completely unaware that their language is in fact based on SGML (Standard Generalized Markup Language). SGML is a highly complex meta language used to define rules for handling different types of documents. Numerous organizations have developed their own standards on the basis of SGML. The spectrum ranges from ancient Sumerian writings to technical documentation for the aircraft construction industry and also includes special formats such as musical notes.

HTML is one application of SGML.

It was "invented" by Tim Berners-Lee in 1989 specifically for the purpose of disseminating documents over the Internet. Berners-Lee concentrated solely on the formal aspects of document representation and excluded everything that - from the point of view the time - was not absolutely essential for the new Web medium. Although there are plenty of possible objections to HTML, it should not be forgotten that without its straightforwardness and limitations, the widespread use and acceptance now enjoyed by the Web would have been impossible. Nine to ten years ago, for example, nobody would have understood the point of document content classifications on the Internet.

Since SGML is HTML's "mother tongue," the most obvious approach would appear to be to make use of SGML to overcome the limitations of HTML. In actual fact, SGML does have a few crucial advantages. Firstly, it is a universal standard that is supported by a large number of software manufacturers. A document database founded on the SGML standard is therefore much more likely to survive the test of time than one built around more ephemeral, manufacturer-specific standards. Secondly, its documents can be read by anybody, and at the same time syntactically dissected and analyzed by suitable programs. Thirdly, SGML documents describe the data itself and not just the way in which it is represented.

SGML has weaknesses of its own, however, that prevent it from gaining acceptance as a universal standard. It originated at a time when most documents were still processed in batch mode. SGML is therefore extremely general-purpose

as well as complex and its specification is comprised of around 500 pages, most of which are not relevant for practical Web purposes. As SGML permits such a large number of alternatives for every single aspect, interoperability between different firms is in practice very limited despite standardization.

FROM SGML TO XML

While SGML is too complicated to meet present-day requirements, HTML is not complicated enough. An HTML extension would also be limited and in need of constant adaptation, each new addition bringing with it a further risk of fragmentation. A simplification of SGML is consequently the answer, not a broader-based HTML. A new meta language, in other words, which allows users to implement their own applications on the basis of a simple standard. Its name: XML.

XML, the eXtensible Markup Language, is - unlike HTML - a subset of SGML. XML is not really an independent markup language at all, but rather a meta language - a definition tool that enables users to build their own markup languages. XML merely provides a linguistic framework that permits the extremely diverse applications constructed with it to understand one another. Compared with SGML, XML is above all very simple: its specification by the World Wide Web Consortium (W3C) is restricted to a much more manageable twenty-six pages.

For the sake of compatibility, XML was constructed in such a way that HTML fits into the new framework. The successor version of HTML 4 bears the name XHTML for this reason and is in reality no more than a special XML application. Of course,

not every HTML page ever written is actually a valid XML page: Web browsers have been extended over the years so that they also understand syntactically incorrect HTML pages - something expressly forbidden by the XML standard.

XML - THE SIMPLE META LANGUAGE

The main difference between HTML and XML is that XML allows users to define their own tags. As a result, data can be structured not only according to formal criteria (such as header, body text, etc.), but also by referring to its content. The content criteria are not defined in XML, however, but on the basis of XML, because XML is a meta language. XML allows content-based structuring because it functions on a different level to HTML. Similarly, it would permit a layout description beyond the scope considered valid by HTML.

XML therefore needs to be fed XML-based input by users in order to thrive. Users can define their own, new tags to suit their specific requirements with the aid of XML - "DateOfBirth," "ShoeSize" or "CookingTime," for instance - providing they also use XML to exchange their recipes. XML does not define these tags itself, but essentially lays down the procedures for defining them. Of course, XML tags do exist that apply to all documents (XML Processing Instructions), as well as tags that are needed to save the definitions in the respective documents (Document Type Definitions).

XML thus provides a set of grammar rules for describing document content. For example:

<identifier> content </identifier>, which the user then has to fill with

actual content, for instance:

```
<DateOfBirth> 25.10.78  
</DateOfBirth>
```

It would be equally conceivable, for example, for meteorologists to define their own tags for exchanging data about the weather, such as <Temperature>, <BarometricPressure>, <WindForce>, etc., and then to save these definitions in appropriate schemas. XML-capable applications could then process a Web page of this kind directly, in other words by evaluating meteorological data automatically via the Web for instance.

Despite this capability, XML doesn't truly 'understand' the content of a document. What the tags actually conceal is entirely up to the user. Instead of a 'meaningful' identifier such as <DateOfBirth>, it would be perfectly feasible to use something like <XYZ56789ABC> without any objection from the XML parser¹. However this would mean forfeiting an important advantage of XML, as it is precisely this option of user-definable tags that allows them to be 'understandable' in the first place.

Even understandable, application-specific tags only make sense if they are known to all the users who are likely to need them. This is the case when user groups agree on certain document types. They can do so using DTDs (Document Type Definitions)² - or XML schema definitions - document schemas which specify the valid tags and the valid structure of a document class. Even then, well formed XML documents can still cope with unknown tags: they are automatically identified as such and left unresolved, whereby the possi-

bility of incorrect interpretation is excluded. The architecture is thus very flexible and has been designed bearing in mind that users who have access to all information need to be served as well. And since the definitions are entered in plain text, not using cryptic control characters, XML documents remain readable by humans. Regardless of the author's particular DTD or XML schema, all XML-coded documents can be processed, saved and distributed.

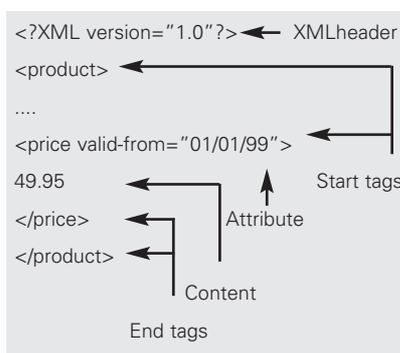


Figure 1: The basic syntax of XML

XML can thus be adapted extremely flexibly for every conceivable application. The only limit is the user's imagination: all information that is available in text form can be entered with XML. It is possible to define prices, authors' names, time and date information, keywords, stock values and so on. If content of this kind is involved, it is advisable to define special document types for specific application families (such as realtors, stock exchange services, publishing houses) with the aid of a DTD or an XML schema. Many professional associations and industrial groups have already reached agreement on suitable XML document types. It is then possible on this basis to issue explicit queries according to the defined logical criteria. The results of an analysis of documents or Web pages that conform to an XML standard can also

be processed directly in application programs. An application can, for instance, automatically extract and utilize prices or stock values it finds on Web pages. And in the same way, meta data can be generated for CAD information or X-ray photographs.

XML is widely used for data interchange between heterogeneous systems. It is far more flexible than the rigid field concept of relational data and enhances the performance of interface standards such as CORBA or DCOM. Even though the current debate about XML technology applies mainly to the Web, XML's range of applications extends well beyond this area, indeed embracing any situation in which complex data needs to be stored or transmitted.

Once agreement has been reached on an XML format, for example for representing molecular structures, it is not only possible to search explicitly in the Web for specific chemical compounds, but also to store and retrieve this kind of information in a database in a similar manner. To map this type of structure in the fields of a relational database in such a way that individual components or compounds could be searched for explicitly would be nearly impossible. The main stumbling block of the relational approach is the complexity of information. What usually happens in the end is that the information is saved as unformatted text or graphics, in which case it can only be represented and not analyzed. Since an estimated 60 percent of all information is only available in the IT system unformatted - the remainder is made up of documents, photographs, graphics, spreadsheets, etc. The potential of a language can be gauged by its ability to handle this data.

¹ The parser analyzes XML documents for syntactic correctness.

² See page 7.

THE SHAKESPEARE

EXAMPLE

The example reproduced below³ demonstrates the underlying logic of XML as compared with an “ordinary”-text. In other words one which is not IT-compatible, on the one hand, and with HTML on the other. It is based on an XML application for representing Shakespeare’s dramas. Once all “Shakespeare users” have agreed on a set of standardized rules for their XML tags (in this case: <ACT>, <TITLE>, etc.), that is with regard to the meaning and usage of keywords such as ACT, SCENE or SPEECH, it will be a straightforward matter to program an application that is suitable for printing out these dramas or, for instance, transferring them to a voice output algorithm. Extensions can be attached seamlessly.

Implementation of XML

The XML world is made up of a series of separate “sub-standards” describing various aspects of document representation and reproduction.

THE DOCUMENT TYPE DEFINITION - DTD

DTDs are schemas for defining document types. They can, for instance, be used to specify that the documents concerned should always contain tags with a fixed structure.⁴ DTDs are laid down at the time the XML applications are developed. XML documents can also be processed without a DTD, but in this case the structure information stored in them is lost. DTDs are nor-

³ Source: <http://metalab.unc.edu/pub/sun-info/standards/xml/eg/shakespeare.1.10.xml.zip>; Shakespeare’s complete work was retrieved from this source in XML format. Also, see example “XML in Helsingør,” page 12.

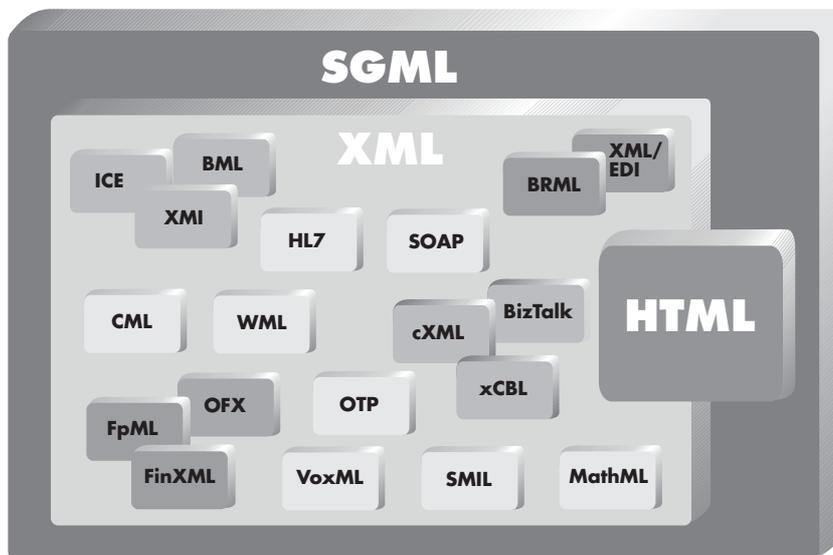


Figure 2: The structure of SGML and XML: XML is a subset of SGML, while HTML is an application of SGML (and of XML)

mally only used to control XML tools and to verify the structural validity of XML documents - they are not necessary to understand these documents.

EXAMPLE OF A DTD:

```
<!ELEMENT product (name, category?,
color?, description?, product-number,
availability?, price?)>
<!ELEMENT name (#PCDATA)*>
<!ELEMENT description (#PCDATA)*>
<!ELEMENT product-number (#PCDATA)*>
<!ELEMENT availability (#PCDATA)*>
<!ELEMENT price (#PCDATA)*>
<!ATTLIST product price valid-from #required>
```

Further information⁵

XML SCHEMA - XSD

The XML Schema 1.0 recommendation addresses one of the most serious drawbacks of XML, namely the absence of data types. Although XML DTDs (Document Type Definitions) allow the tags and structure of

⁴ XML distinguishes “well formed” documents that are syntactically correct from those which are “valid” (structurally correct in accordance with a DTD).

a document class to be specified, the content of document elements and the values of attributes are plain text (strings). The XML schema introduces types such as number, date, time, etc. into XML and also permits user-defined data types. The XML schema additionally supports modularization, which makes a schema easier to reuse. XML schemas are written in XML and can thus be processed using XML tools.

Further information^{6, 7, 8}

THE EXTENSIBLE STYLESHEET LANGUAGE - XSL

The layout of an XML document is not defined in the document itself or in its DTD. It is one of the fundamental principles of XML that content should be absolutely separate from presentation. How a document is represented is defined in a stylesheet that is created with XSL (or possibly CSS). One document can also have several different XSL stylesheets, resulting in different docu-

⁵ <http://www.w3.org/XML/1998/06/xmlspec-19980910.dtd>
⁶ <http://www.w3.org/TR/NOTE-xml-schema-req>
⁷ <http://www.xml.com/schemas/>
⁸ <http://www.xml.com/pub/1999/07/schemas/syntax.html>

ment presentations. XSL likewise supports a variety of output media, such as screen display, printouts, etc. In addition, XSL enables XML documents to be translated into HTML documents. If this takes place on a server, terminal devices that understand HTML but not XML can be supported as well.

EXAMPLE OF AN XSL STYLE-SHEET:

```
<xsl:template match="product-catalog">
<ul>
<xsl:apply-templates select="product">
<xsl:sort select="name"/>
</xsl:apply-templates>
</ul>
</xsl:template>
<xsl:template match="product">
<li>
<xsl:value-of select="name"/>
<xsl:text> </xsl:text>
<xsl:value-of select="product-number"/>
<xsl:text> </xsl:text>
<xsl:value-of select="price"/>
</li>
</xsl:template>
```

In this example a stylesheet sorts all products from a product catalog by "name" and outputs the list as an HTML document. As we can see, XSL style sheets are themselves XML documents and can therefore also be processed using the same tools as XML. XSLT is a subset of XSL designed for transformation of XML documents into other XML documents.

Further information⁹

XSL attained W3C recommendation status on October 15, 2001.

Further information¹⁰

XPOINTER AND XLINK

The conventions for linking objects to XML documents are defined by means of XLink and XPointer. The following link types are supported in addition to "classic" HTML links:

- Bidirectional links
- Extended 1:n links (links to several versions of the same document, for example)
- Indirect links
- Pointer addressing (links can also pinpoint certain locations within a document)

Further information^{11, 12, 13}

XPATH - FOR XML DOCUMENT QUERIES

XPath is a W3C recommendation that is becoming the favored language for querying and addressing parts of XML documents. As such, it replaces the initially proposed XQL query language. XPath was created to provide a common syntax and semantics for querying and addressing the contents of XML documents. In the same way as SQL with relational databases, XPath serves to retrieve XML documents or parts of documents from an XML data source, such as a database. XPath is also used by XSLT (XSL Transformation Language), XLink and XPointer. It operates on the abstract, logical structure of an XML document as a tree of nodes, allowing certain elements or fragments to be picked out of a document.

Further information^{14, 15}

THE DOCUMENT OBJECT MODEL - DOM

The Document Object Model is an API (Application Programming Interface) for HTML and XML documents. It allows application programs to navigate within the structures of documents and to retrieve, add, modify or delete individual elements or attributes. DOM is designed to be used with practically any programming language. The model is supported, for instance, by Microsoft's Office2000 and by other vendors.

Further information¹⁶

XQUERY

Currently in a working draft status, XQuery is W3C's attempt to propose an XML query language that allows the combining of data from multiple sources. While these sources can be transient or persistently stored in traditional databases, native XML databases, XML programming libraries or other XML repositories, XQuery is a flexible functional language that allows one to generate query expressions which are often composed of many other expressions. Its expressive power gives users in any environment the possibility to intelligently query the many kinds of data contained in XML documents through a single query language. Similarly, XQuery now allows developers to implement their DOM, XPath and XSLT applications in just one language.

Adding to this, XQuery also covers the major functionality of former query language proposals like XML-QL, XQL, OQL or the SQL standard, thus making it a viable candidate for replacing these query languages before long.

QUIP is Software AG's first implementation of XQuery which developers can download and try out free of charge through the Tamino Developer Community.

Further information about XQuery¹⁷

Further information about QUIP¹⁸

9 <http://www.w3.org/TR/xslt>

10 <http://www.w3.org/TR/2001/REC-xsl-20011015/>

11 <http://www.w3.org/TR/WD-xptr>

12 <http://www.w3.org/TR/xlink/>

13 <http://www.xml.com/pub/a/2000/09/xlink/index.html>

14 <http://www.w3.org/TR/xpath>

15 <http://www.w3.org/TR/xpath20/>

16 <http://www.w3.org/DOM/>

17 <http://www.w3.org/TR/xquery/>

18 <http://www.softwareag.com/developer/quip/default.htm>

XML - A universal document standard

Since XML is a meta language, it is specified on two different levels:

- The XML standard itself is supported by the World Wide Web Consortium (W3C), which is also responsible for developing it further.
- Specific XML applications are developed by independent user groups.

Further information¹⁹

STANDARDIZATION AND W3C

XML is an independent standard which is maintained by the World Wide Web Consortium.

W3C was set up by the Massachusetts Institute of Technology (MIT) in collaboration with CERN in Geneva and with the backing of the European Commission. W3C enjoys widespread support within the IT industry. Software AG is also an active member of W3C.

Further information²⁰

XML APPLICATIONS

There are already a number of industrial initiatives aimed at establishing XML as a data interchange standard. Firms and organizations in all branches have joined forces to develop XML applications:

- **HL7** (Kona proposal), a coalition of health organizations which develops standards for exchanging hospital, financial and administrative data electronically between various independent computer systems used in the health service.^{21, 22}
- **CML** The Chemical Markup Language, developed in the United Kingdom to enable chemists to exchange descriptions of molecules, formulas and other chemical data.²³
- **OFX** Open Financial Exchange the format used by Intuit Quicken and Microsoft Money to communicate with banks.

- **OSD** Open Software Distribution from Marimba and Microsoft.

Other activists, to name but a few:

- **aecXML** for Architecture, Engineering and Construction
- **AIML** - Artificial Intelligence Markup Language
- **BioML** - Biopolymer Markup Language
- **BizTalk** - Framework for exchange of business documents²⁴
- **BML** - Bean Markup Language
- **BRML** - Common Rules and Business Rules Markup Language
- **BSML** - Bioinformatic Sequence Markup Language
- **CFML** - Cold Fusion Markup Language
- **CPML** - Call Policy Markup Language
- **CPL** - Call Processing Language
- **cXML** - Commerce XML protocol²⁵
- **DTD** for patent documents - ST32 US Patent Grant
- **EAD** - Encoded Archival Description
- **ebXML** - Electronic Business XML Initiative²⁶
- **ECMdata** - Electronic Component Manufacturer Data Sheet Library Specification
- **FinXML** - XML for Capital Markets (Financial ML)
- **FLBC** - Formal Language for Business Communication
- **FpML** - Financial Product Markup Language
- **ICE** - Information and Content Exchange
- **IOTP** - Internet Open Trading Protocol
- **JSML** - Java Speech Markup Language
- **KBML** - Koala Bean Markup Language
- **LitML** - Liturgical Markup Language
- **MathML** - Mathematical Markup Language²⁷
- **MoDL** - Molecular Dynamics (Markup) Language

- **NAA** Standard for Classified Advertising Data
- **NVML** - Navigation Markup Language
- **OTP** - Open Trading Protocol
- **PDX** - Product Definition Exchange
- **PML** - Markup Language for Paper and Printing - Procedural ML - Portal ML
- **PMML** - Predictive Model Markup Language
- **RDF** - Resource Description Framework²⁸
- **RosettaNet PIPs** - Partner Interface Processes²⁹
- **SMIL** - Synchronized Multimedia Integration Language³⁰
- **SOAP** - Simple Object Access protocol³¹
- **SVG** - Scalable Vector Graphics³²
- **TML** - Tutorial Markup Language
- **TMX** - Translation Memory Exchange
- **VoxML** - Voice Recognition ML
- **WIDL** - Web Interface Definition Language
- **WML** - Wireless Markup Language for WAP³³
- **XBEL** - XML Bookmark Exchange Language
- **XBRL** - Extensible Business Reporting Language
- **xCBL** - XML Common Business Library³⁴
- **XFRML** - Extensible Financial Reporting Markup Language
- **XMI** - XML Metadata Interchange

Detailed information available at: <http://www.xml.com/>

19 <http://www.w3.org/XML/>

20 <http://www.xml.com/pub/q/stdlist>

21 <http://xml.coverpages.org/gen-apps.html#HL7-SGML>

22 <http://xml.coverpages.org/konaProp970718.html>

23 <http://www.xml-cml.org/>

24 <http://www.biztalk.org>

25 <http://www.cxml.org/home/>

26 <http://www.ebxml.org/>

27 <http://www.w3.org/TR/MathML2>

28 <http://www.w3.org/TR/PR-rdf-schema>

29 <http://www.rosettanet.org>

30 <http://www.w3.org/TR/REC-smil/>

31 <http://www.w3.org/TR/SOAP/>

32 <http://www.w3.org/TR/2001/PR-SVG-20010719/index.html>

33 <http://www.wapforum.org/what/technical.htm>

34 <http://www.commerceone.com/xml/cbl/index.html>

EDI

EDI (Electronic Data Interchange) is one of the most important applications of XML. For several years now, attempts have been made to establish a standard for exchanging information between corporations on the basis of EDI. The underlying principle is both simple and convincing: if it were possible to exchange data of the kind that is contained, for instance, in delivery documents, orders or invoices in a compatible format, we could cut out the time-consuming, error-prone process of entering it manually. In practice, however, this approach has always failed to live up to expectations on account of the overwhelming abundance of data, data types and formats. In the USA, EDI is used by a mere 2-3 percent of all companies.

XML enables these difficulties to be overcome because it provides a common umbrella for all EDI applications, though one which still allows sufficient free scope to accommodate all possible requirements. XML/EDI, developed by the XML/EDI Group, represents a new framework for business-to-business data interchange and thus also for the important field of electronic business. Not only can documents be exchanged on the basis of XML, but 'proper' transactions can be effected. A high-performance XML infrastructure is therefore crucial for developing powerful applications.³⁵

³⁵ Further information:
XML/EDI Group Home Page URL:
<http://www.xmledi.net/>

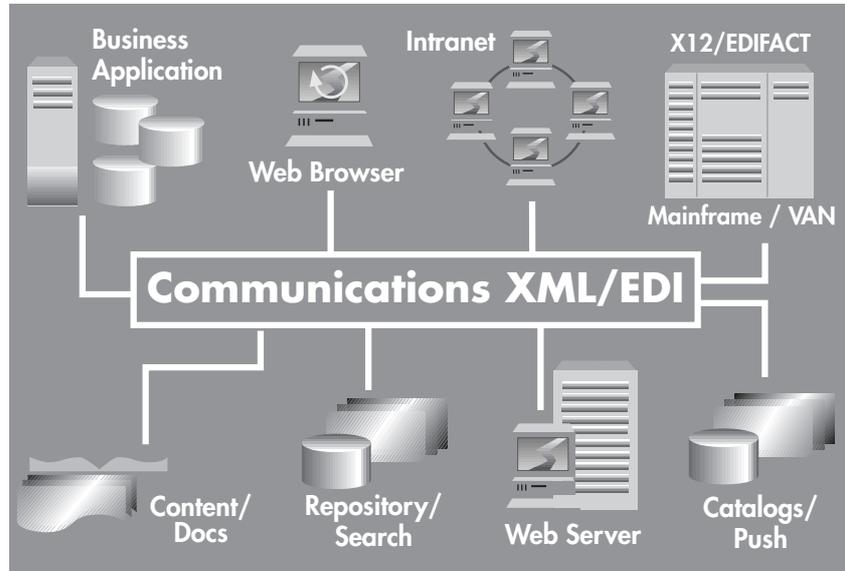


Figure 3: Business-to-business communication with XML/EDI

XML AND DATABASES

XML's great flexibility opens up a range of applications for this standard, far broader than that available to HTML. Not only can XML process complex, hierarchical information, it can also be used for commercial transactions. XML is to the application layer what TCP/IP is to the communication layer. These two standards together form a foundation that allows the Web to be utilized for electronic business.

Against this backdrop there are two requirements that must be satisfied by the XML infrastructure:

- XML information must be made available quickly and reliably - if necessary in large amounts and for transactions.
- XML information must be integrated with existing corporate data.

Whereas HTML pages can still be managed in a file system, more powerful concepts are needed for complex XML documents. The present state of the art is to use databases whose functionality - consistency, restart capability, data

security and recovery, etc. - can be utilized by the XML data.

The simplest way to accommodate XML objects in a database would be to save them as "character large objects." The tags in these objects would be interpreted as straightforward running text, which could be handled using the retrieval methods for full text. A database offers further possibilities for indexing the information objects and thus facilitates more flexible access paths. This would permit these objects to be accessed both via their structures (structure-based retrieval) and via their content (content-based retrieval). All relevant information would in this case have to be managed in separate fields or tables, however. On the other hand, it would not be possible to use this method to map the hierarchical structures of XML documents. Although a solution of this kind could be implemented relatively easily with the means currently available, it would certainly not be adequate to cope efficiently with large volumes of data or, above all, XML transaction data.

XML AND SQL

The most obvious approach for integrating XML into conventional database technologies would be to implement XML structures with the aid of data models that are in common use today, such as the relational model. Either SQL data structures could be used at an XML interface or any desired XML object could be converted to a relational data structure.

It is never problematic to convert existing SQL data into XML objects. Although the absence of data types in XML still constitutes an obstacle at the moment, a remedy is seen in the form of the XML schema (see above). Since, with relational data, complex data objects are composed of "flat" tables in the application logic, either additional definitions are required in the form of data maps or different data records must be joined together with a JOIN operator integrated in an XQL query.

On the other hand, implementing every possible tree structure that is supported by XML in a relational data model is a surmountable problem, one which in practice is extremely difficult to assimilate, though. XML allows information to be graded hierarchically to any required depth. In order to reproduce such hierarchies in a relational database, complex table links would have to be not simply created but also maintained. Practice has shown that even a comparatively low number of levels leads to massive problems with regard to performance. This kind of model would therefore only be suitable for managing very simple XML documents.

The locking mechanisms of RDBMSs represent a further serious obstacle. Locks on the document level are not

supported, since documents are irrelevant to the RDBMS methodology. In order to modify an XML document mapped in an RDBMS, a large number of locks would have to be checked in several different tables - yet another cause of deteriorations in performance which would also entail an enormous amount of internal administration.

Moreover, the typical document structure of many information objects modeled with the aid of XML, such as long text elements, images and complex lateral links, resists direct representation with relational tools, quite apart from the fact that XML also allows the use of non-predefined tags. After all, the great advantage of XML is that is capable of mapping such objects better than "pure" SQL.

This problem furthermore depends on the location within the database at which the conversion from XML to the relational model takes place - as an external interface in other words or close to the kernel.

XML AND ODBMS

At first glance, the world of hierarchically structured XML objects suggests a direct relationship with object-oriented databases (ODBMS). The latter do indeed offer the almost generic option for XML of implementing the structures of persistent objects in a server. Persistent objects could thus be made available with XML at a transparent interface and accessed by a large number of users either locally or via the Web. On the other hand, the familiar drawbacks of ODBMSs cannot be avoided if the data passes via an XML interface. In practice, ODBMSs are inadequate for high throughput or large volumes of data, as well as

being particularly unsuitable for transaction tasks of the kind essential for electronic business. In addition, ODBMSs are complex to implement and prone to an especially serious deterioration in performance if required to integrate "old data" from an RDBMS or - worse still - a file system. In the past, ODBMSs have only proved successful for special tasks - and XML most definitely cannot be considered "special." An XML server provides an ideal basis for storing and exchanging XML objects of different types. Thanks to the option of implicit data structuring, it offers an enormous degree of flexibility, which is particularly important for electronic business applications. Database queries and instructions are not formulated as a string of SQL queries, but are sent to the server as URL. This means that a single query can be applied not just to the documents actually on the XML server, but also to other data sources such as remote XML servers, relational databases and multimedia elements installed on special multimedia servers. As far as the user is concerned, the data appears to come from one server only - XML is the glue that holds the different elements together.

In a hospital system, for instance, patient records might be available as XML documents, their master data might be stored in a relational database and their CAT-scan images could exist in the form of volume image files. All this information could be combined by XML and an XML server in to a single document.

Using XML shields a concept of this kind from the problems associated with proprietary approaches.

In order to provide the highest access speed to stored information, Software AG has realized this concept with Tamino XML Server. The product stores XML documents natively, that is, without converting them in to other formats.

More examples of XML applications

The examples below are not simplifications of the kind customarily used to explain the basic principles of XML, but rather genuine XML applications. Tamino has been designed for managing precisely this type of information.

Example 1: XML in Helsingör Castle

Plain text	HTML	XML
<p>Act One SCENE ONE. Helsingör. A terrace in front of the castle. Francisco is on sentinel duty. Enter Bernardo.</p> <p>BERNARDO: Who's there? FRANCISCO: Nay, answer me. Stand and unfold yourself. BERNARDO: Long live the King!</p>	<pre><H1>ACT ONE</H1> <P><I>SCENE ONE. Helsingör. A terrace in front of the castle.</I></P> <P><I>FRANCISCO is on sentinel duty. Enter BERNARDO.</I></P> <P>BERNARDO:Who's there?</P> <P>FRANCISCO:Nay, answer me. Stand and unfold yourself.</P> <P>BERNARDO:Long live the King!</P></pre>	<pre><ACT><TITLE>ACT ONE</TITLE> <SCENE> <TITLE>SCENE ONE. Helsingör. A terrace in front of the castle.</TITLE> <STAGEDIR>FRANCISCO is on sentinel duty. Enter BERNARDO.</STAGEDIR> <SPEECH> <SPEAKER>BERNARDO</SPEAKER> <LINE>Who's there?</LINE> </SPEECH> <SPEECH> <SPEAKER>FRANCISCO</SPEAKER> <LINE>Nay, answer me. Stand and unfold yourself.</LINE> </SPEECH> <SPEECH> <SPEAKER>BERNARDO</SPEAKER> <LINE>Long live the King!</LINE> </SPEECH> </SCENE> </ACT></pre>
<p>Human beings don't need any text codes or tags in order to be able to read this excerpt from Shakespeare's Hamlet. Thanks to what they learned in school many moons ago, they are able to identify or derive the meaning of individual text passages.</p>	<p>The same text in HTML format. The HTML tags merely define how the text is represented in a browser. They do not add any semantic value to it and are thus unable to provide any information with regard to content (for example: "How much text does Bernardo have to recite?").</p>	<p>This time, XML tags have been added to the text. A computer is not able to derive meaning from a context, just to analyze character strings on the basis of defined rules. It therefore needs text codes to infer information about the content. The XML text could now be loaded in a voice computer, for instance, or used to draw up a cast list.</p>

Example 2:

BIOML – Polymer sequences, insulin (considerably shortened)

BIOML was developed with XML for annotating biopolymer structures. The aim was to facilitate an exchange of experiences between scientists using the Web. This example describes an insulin protein.

```
<?xml version="1.0"?>
<!DOCTYPE bioml SYSTEM="bioml.dtd">
<bioml label="Insulin, gene and protein structure">
  &paragraph;This BIOML file contains a simple set of information about the
  protein insulin. It is not meant to be an exhaustive study of insulin. Instead
  it is meant to be a demonstration of the organization of a BIOML file. To advance to the
  next item, either select it with the mouse or press the "DOWN" button.
  <organism label="Homo sapiens (human)">
    <chromosome label="Chromosome 11" number="11">
      &paragraph;
      The chromosome entry indicates that the locus that contains the insulin gene
      has been localized to the number 11 chromosome in humans. All of the
      entries linked to this entry are localized on this chromosome.
      <locus label="HUMINS locus">
        &paragraph;
        The HUMINS locus contains all of the sequence information necessary to code for
        insulin. A locus can be completely known, or only fragments may be known. In
        many BIOML files, the locus will contain a gene (or genes) of interest.
        ...
        <gene label="Insulin gene">
          <dna label="Complete HUMINS sequence" start="1" end="4992">
            1 ctcgaggggc ctgacattg cctccagag agagcaccca acaccctcca ggcttgaccg
            61 gccagggtgt cccttccta ccttgagag agcagcccca gggcatcctg cagggggtgc
            121 tgggacacca gctggcctc aaggctctg cctccctcca gccacccac tacacgctgc
            181 tgggatcctg gatctcagct cctggccga caaactggc aaactctac tcatccacga
            ...
            4861 ggccagggct gggcaggcgg gtggacggcc ggacactggc cccggaagag gagggaggcg
            4921 gtggctggga tcggcagcag cgtccatgg gaacaccag cggcccccac tcgcacgggt
            4981 agagacagcg gc
            ...
          </dna>
        </gene>
      </locus>
    </chromosome>
    ...
  </organism>
  ...
  <copyright label="1998 ProteoMetrics, LLC">
    &cr;&copyright; 1998 ProteoMetrics, LLC. All rights reserved.
  </copyright>
</bioml>
```

Source: <http://www.bioml.com/BIOML/Examples/insulin3.html>

Example 2 has been reproduced in part, courtesy of ProteoMetrics, LLC, New York.

Example 3

DETERMINATION OF VEGETATION DIFFERENCES IN AUSTRALIA

Source: <http://www.proteometrics.com/BIOML/Examples/insulin3.html>

```
<?xml version="1.0"?>
<!DOCTYPE anzmeta PUBLIC "-//ANZLIC//DTD ANZMETA 1.1//EN"
"http://www.environment.gov.au/net/anzmeta/anzmeta-1.1.dtd">
<anzmeta>
  <citeinfo>
    <uniqueid>
      ANZCW030100001
    </uniqueid>
    <title>
      AVHRR NDVI biweekly series covering
      continental Australia at full resolution.
    </title>
    <origin>
      <custod>
        Bureau of Meteorology
      </custod>
      <jurisdic>
        <keyword>
          Australia
        </keyword>
      </jurisdic>
    </origin>
  </citeinfo>
  <descript>
    <abstract>
      <p>
        AVHRR NDVI (Normalized Difference
        Vegetation Index) biweekly series covering
        continental Australia at a 1 kilometer
        resolution. NDVI is a measure of the absorp-
        tion of red light by plant chlorophyll and the
        reflection of infrared adiation by water-filled
        leaf cells. Its values broadly measure the
        density of active foliage.
      </p>
    </abstract>
    <theme>
      <keyword qualifier="biodiversity">
        AGRICULTURE
      </keyword>
      <keyword>
        ATMOSPHERE
      </keyword>
      <keyword qualifier="Monitoring">
        ATMOSPHERE Pressure
      </keyword>
      <keyword qualifier="Mapping">
        CLIMATE AND WEATHER
      </keyword>
      <keyword>
        HAZARDS Pests
      </keyword>
    </theme>
    <spdom>
      <place>
        <dsgpolyo>
          <long>112.5</long>
          <lat>-10</lat>
          <long>154</long>
          <lat>-10</lat>
          <long>154</long>
          <lat>-44</lat>
          <long>112.5</long>
          <lat>-44</lat>
          <long>112.5</long>
          <lat>-10</lat>
        </dsgpolyo>
      </place>
      <bounding>
        <northbc>-10</northbc>
        <southbc>-44</southbc>
        <eastbc>154</eastbc>
        <westbc>112.5</westbc>
      </bounding>
    </spdom>
  </descript>
  <timeperd>
    <begdate>
      <date>
        1991-04-01
      </date>
    </begdate>
  </timeperd>
  .
  .
  .
  .
  .
```

For more information on Tamino XML Server:
<http://www.softwareag.com/tamino>

Tamino Community:
<http://www.softwareag.com/developer>

Download XML Starter Kit:
<http://www.xmlstarterkit.com>

Tamino Demo Zone:
<http://tamino.demozone.softwareag.com/>

"XML - The Site" Resource Pages:
<http://www.softwareag.com/xml/>

Software AG
Corporate Headquarters

Umlandstraße 12
64297 Darmstadt/Germany
Tel.: +49-61 51-92-0
Fax: +49-61 51-92-11 91
www.softwareag.com/tamino

 **SOFTWARE AG**
THE XML COMPANY