XML

XML

http://en.wikipedia.org/wiki/XML

This Book Is Generated By WikiType

using

RenderX DiType, XML to PDF XSL-FO Formatter

29 September 2008

Table of Contents

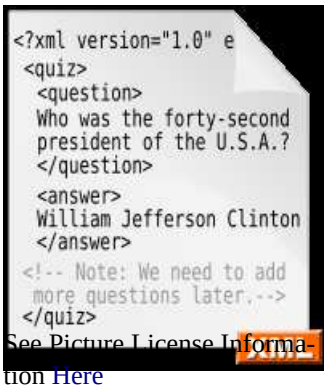
XML

See Picture License Information Here

This article may be too long.

Please discuss this issue on the talk page; if necessary, split the content into subarticles and keep this article in a **summary style**.

Extensible Markup Language



See Picture License Information Here

Filename extension	.xml
Internet media type	application/xml, text/xml (deprecated)
Uniform Type Identifier	public.xml
Developed by	World Wide Web Consortium
Type of format	Markup language
Extended from	SGML
Extended to	XHTML, RSS, Atom, ...
Standard(s)	1.0 (Fourth Edition) 1.1 (Second Edition)

The **Extensible Markup Language** (**XML**) is a general-purpose *specification* for creating custom markup languages.[1] It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to help information systems share structured data, particularly via the Internet,[2] and it is used both to encode documents and to serialize data. In the latter context, it is comparable with other text-based serialization languages such as JSON and YAML.[3]

It started as a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML,[4] RSS, MathML, GraphML, Scalable Vector Graphics, MusicXML, and thousands of others. Moreover, XML is sometimes used as the specification language for such application languages.

XML is recommended by the World Wide Web Consortium (W3C). It is a fee-free open standard. The recommendation specifies both the lexical grammar and the requirements for parsing.

Well-formed and valid XML documents

There are two levels of correctness of an XML document:

- **Well-formed**. A well-formed document conforms to all of XML's syntax rules. For example, if a start-tag appears without a corresponding end-tag, it is not *well-formed*. A document that is not well-formed is not considered to be XML; a *conforming parser* is not allowed to process it.
- **Valid**. A valid document additionally conforms to some semantic rules. These rules are either user-defined, or included as an XML schema, especially DTD. For example, if a document contains an undefined element, then it is not *valid*; a *validating parser* is not allowed to process it.

Well-formed documents: XML syntax

As long as only well-formedness is required, XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree. The only indispensable syntactical requirement is that the document has exactly one **root element** (alternatively called the **document element**). This means that the text must be enclosed between a root start-tag and a corresponding end-tag. The following is a "well-formed" XML document:

```
<book>This is a book... </book>
```

The root element can be preceded by an optional **XML declaration**. This element states what version of XML is in use (normally 1.0); it may also contain information about character encoding and external dependencies.

```
<?xml version="1.0" encoding="UTF-8"?>
```

The specification *requires* that processors of XML support the pan-Unicode character encodings UTF-8 and UTF-16 (UTF-32 is not mandatory). The use of more limited encodings, such as those based on ISO/IEC 8859, is acknowledged and is widely used and supported.

Comments can be placed anywhere in the tree, including in the text if the content of the element is text or #PCDATA.

XML comments start with **<!--** and end with **-->**. Two consecutive dashes (--) may not appear anywhere in the text of the comment.

```
<!-- This is a comment. -->
```

In any meaningful application, additional markup is used to structure the contents of the XML document. The text enclosed by the root tags may contain an arbitrary number of XML elements. The basic syntax for one **element** is:

```
<element_name attribute_name="attribute_value">Element Content</element_name>
```

The two instances of »element_name« are referred to as the **start-tag** and **end-tag**, respectively. Here, »Element Content« is some text which may again contain XML elements. So, a generic XML document contains a tree-based data structure. Here is an example of a structured XML document:

```
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="8" unit="dL">Flour</ingredient>
  <ingredient amount="10" unit="grams">Yeast</ingredient>
  <ingredient amount="4" unit="dL" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together.</step>
    <step>Knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again.</step>
    <step>Place in a bread baking tin.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Bake in the oven at 180(degrees)C for 30 minutes.</step>
  </instructions>
</recipe>
```

Attribute values must always be quoted, using single or double quotes; and each attribute name must appear only once in any element.

XML requires that elements be properly nested — elements may never overlap, and so must be closed in the opposite order to which they are opened. For example, this fragment of code below cannot be part of a well-formed XML document because the *title* and *author* elements are closed in the wrong order:

```
<!-- WRONG! NOT WELL-FORMED XML! -->
<title>Book on Logic<author>Aristotle</title></author>
```

One way of writing the same information in a way which could be incorporated into a well-formed XML document is as follows:

```
<!-- Correct: well-formed XML fragment. -->
<title>Book on Logic</title> <author>Aristotle</author>
```

XML provides special syntax for representing an element with empty content. Instead of writing a start-tag followed immediately by an end-tag, a document may contain an empty-element tag. An empty-element tag resembles a start-tag but contains a slash just before the closing angle bracket. The following three examples are equivalent in XML:

```
<foo></foo>
<foo />
<foo/>
```

An empty-element may contain attributes:

```
<info author="John Smith" genre="science-fiction" date="2009-Jan-01" />
```

Entity references

An entity in XML is a named body of data, usually text. Entities are often used to represent single characters that cannot easily be entered on the keyboard; they are also used to represent

pieces of standard ("boilerplate") text that occur in many documents, especially if there is a need to allow such text to be changed in one place only.

Special characters can be represented either using entity references, or by means of numeric character references. An example of a numeric character reference is "€", which refers to the Euro symbol by means of its Unicode codepoint in hexadecimal.

An entity reference is a placeholder that represents that entity. It consists of the entity's name preceded by an ampersand ("&") and followed by a semicolon (";"). XML has five predeclared entities:

- & (& or "ampersand")
- < (< or "less than")
- > (> or "greater than")
- ' (' or "apostrophe")
- " (" or "quotation mark")

Here is an example using a predeclared XML entity to represent the ampersand in the name "AT&T":

```
<company_name>AT&amp;T</company_name>
```

Additional entities (beyond the predefined ones) can be declared in the document's Document Type Definition (DTD). A basic example of doing so in a minimal internal DTD follows. Declared entities can describe single characters or pieces of text, and can reference each other.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE example [
    <!ENTITY copy "&#xA9;">
    <!ENTITY copyright-notice "Copyright &copy; 2006, XYZ Enterprises">
]>
<example>
    &copyright-notice;
</example>
```

When viewed in a suitable browser, the XML document above appears as:

Copyright © 2006, XYZ Enterprises

Numeric character references

Numeric character references look like entity references, but instead of a name, they contain the "#" character followed by a number. The number (in decimal or "x"-prefixed hexadecimal) represents a Unicode code point. Unlike entity references, they are neither predeclared nor do they need to be declared in the document's DTD. They have typically been used to represent characters that are not easily encodable, such as an Arabic character in a document produced on a European computer. The ampersand in the "AT&T" example could also be escaped like this (decimal 38 and hexadecimal 26 both represent the Unicode code point for the "&" character):

```
<company_name>AT&#38;T</company_name>
<company_name>AT&#x26;T</company_name>
```

Similarly, in the previous example, notice that "©" is used to generate the “©” symbol.

See also numeric character references.

Well-formed documents

In XML, a well-formed document must conform to the following rules, among others:

- 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 with either single (') or double (") quotes. Single quotes close a single quote and double quotes close a double quote.
- Tags may be nested but must not overlap. Each non-root element must be completely contained in another element.
- The document complies with its declared character encoding. The encoding may be declared or implied externally, such as in "Content-Type" headers when a document is transported via HTTP, or internally, using explicit markup at the very beginning of the document. When no such declaration exists, a Unicode encoding is assumed, as defined by a Unicode Byte Order Mark before the document's first character. If the mark does not exist, UTF-8 encoding is assumed.

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

<Step> ... </Step>

whereas this is not

<Step> ... </step>
<STEP> ... </step>

By carefully choosing the names of the XML elements one may 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.

Automatic verification

It is relatively simple to verify that a document is well-formed or validated XML, because the rules of well-formedness and validation of XML are designed for portability of tools. The idea is that any tool designed to work with XML files will be able to work with XML files written in any XML language (or XML application). Here are some examples of ways to verify XML documents:

- load it into an XML-capable browser, such as Firefox or Internet Explorer
- use a tool like xmlwf (usually bundled with expat)
- parse the document, for instance in Ruby:

```
irb> require "rexml/document"
irb> include REXML
irb> doc = Document.new(File.new("test.xml")).root
```

Valid documents: XML semantics

By leaving the names, allowable hierarchy, and meanings of the elements and attributes open and definable by a customizable *schema* or *DTD*, XML provides a syntactic foundation for the creation of purpose-specific, XML-based markup languages. The general syntax of such lan-

guages is rigid — documents must adhere to the general rules of XML, ensuring that all XML-aware software can at least read and understand the relative arrangement of information within them. The schema merely supplements the syntax rules with a set of constraints. Schemas typically restrict element and attribute names and their allowable containment hierarchies, such as only allowing an element named 'birthday' to contain one element named 'month' and one element named 'day', each of which has to contain only character data. The constraints in a schema may also include data type assignments that affect how information is processed; for example, the 'month' element's character data may be defined as being a month according to a particular schema language's conventions, perhaps meaning that it must not only be formatted a certain way, but also must not be processed as if it were some other type of data.

An XML document that complies with a particular schema/DTD, in addition to being well-formed, is said to be **valid**.

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such schemas, and some of these languages are XML-based, themselves.

Before the advent of generalised data description languages such as SGML and XML, software designers had to define special file formats or small languages to share data between programs. This required writing detailed specifications and special-purpose parsers and writers.

XML's regular structure and strict parsing rules allow software designers to leave parsing to standard tools, and since XML provides a general, data model-oriented framework for the development of application-specific languages, software designers need only concentrate on the development of rules for their data, at relatively high levels of abstraction.

Well-tested tools exist to validate an XML document "against" a schema: the tool automatically verifies whether the document conforms to constraints expressed in the schema. Some of these validation tools are included in XML parsers, and some are packaged separately.

Other usages of schemas exist: XML editors, for instance, can use schemas to support the editing process (by suggesting valid elements and attributes names, etc).

DTD

Main article: Document Type Definition

The oldest schema format for XML is the Document Type Definition (DTD), inherited from SGML. While DTD support is ubiquitous due to its inclusion in the XML 1.0 standard, it is seen as limited for the following reasons:

- It has no support for newer features of XML, most importantly namespaces.
- It lacks expressiveness. Certain formal aspects of an XML document cannot be captured in a DTD.
- It uses a custom non-XML syntax, inherited from SGML, to describe the schema.

DTD is still used in many applications because it is considered the easiest to read and write.

XML Schema

Main article: XML Schema (W3C)

A newer XML schema language, described by the W3C as the successor of DTDs, is XML Schema, or more informally referred to by the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system, allow for more detailed constraints on an XML document's logical structure, and must be processed in a more robust validation framework. XSDs also use an XML-based format, which makes it possible to use ordinary XML tools to help process them, although XSD implementations require much more than just the ability to read XML.

RELAX NG

Main article: RELAX NG

Another popular schema language for XML is RELAX NG. Initially specified by OASIS, RELAX NG is now also an ISO international standard (as part of DSDL). It has two formats: an XML based syntax and a non-XML compact syntax. The compact syntax aims to increase read-ability and writability but, since there is a well-defined way to translate the compact syntax to the XML syntax and back again by means of James Clark's Trang conversion tool, the advantage of using standard XML tools is not lost. RELAX NG has a simpler definition and validation framework than XML Schema, making it easier to use and implement. It also has the ability to use datatype framework plug-ins; a RELAX NG schema author, for example, can require values in an XML document to conform to definitions in XML Schema Datatypes.

ISO DSDL and other schema languages

The ISO DSDL (Document Schema Description Languages) standard brings together a comprehensive set of small schema languages, each targeted at specific problems. DSDL includes RELAX NG full and compact syntax, Schematron assertion language, and languages for defining datatypes, character repertoire constraints, renaming and entity expansion, and namespace-based routing of document fragments to different validators. DSDL schema languages do not have the vendor support of XML Schemas yet, and are to some extent a grassroots reaction of industrial publishers to the lack of utility of XML Schemas for publishing.

Some schema languages not only describe the structure of a particular XML format but also offer limited facilities to influence processing of individual XML files that conform to this format. DTDs and XSDs both have this ability; they can for instance provide attribute defaults. RELAX NG and Schematron intentionally do not provide these; for example the infoset augmentation facility.

International use

XML supports the direct use of almost any Unicode character in element names, attributes, comments, character data, and processing instructions (other than the ones that have special symbolic meaning in XML itself, such as the open corner bracket, "<"). Therefore, the following is a well-formed XML document, even though it includes both Chinese and Cyrillic characters:

```
<?xml version="1.0" encoding="UTF-8"?>
<  >Данные</  >
```

Displaying XML on the web

Generally, generic XML documents do not carry information about how to display the data.[5] Without using CSS or XSLT, a generic XML document is rendered as raw XML text by most web browsers. Some display it with 'handles' (e.g. + and - signs in the margin) that allow parts of the structure to be expanded or collapsed with mouse-clicks.

In order to style the rendering in a browser with CSS, the XML document must include a reference to the stylesheet:

```
<?xml-stylesheet type="text/css" href="myStyleSheet.css"?>
```

Note that this is different from specifying such a stylesheet in HTML, which uses the <link> element.

XSLT (XSL Transformations) can be used to alter the format of XML data, either into HTML or other formats that are suitable for a browser to display.

To specify client-side XSLT, the following processing instruction is required in the XML:

```
<?xml-stylesheet type="text/xsl" href="myTransform.xslt"?>
```

Client-side XSLT is supported by many web browsers. Alternatively, one may use XSLT to convert XML into a displayable format *on the server* rather than being dependent on the end-user's browser capabilities. The end-user is not aware of what has gone on 'behind the scenes'; all they see is well-formatted, displayable data.

See the XSLT article for an example of server-side XSLT in action.

XML extensions

- **XPath** makes it possible to refer to individual parts of an XML document. This provides random access to XML data for other technologies, including XSLT, XSL-FO, XQuery etc. XPath expressions can refer to all or part of the text, data and values in XML elements, attributes, processing instructions, comments etc. They can also access the names of elements and attributes. XPaths can be used in both valid and well-formed XML, with and without defined namespaces.
- **XInclude** defines the ability for XML files to include all or part of an external file. When processing is complete, the final XML infoset has no XInclude elements, but instead has copied the documents or parts thereof into the final infoset. It uses XPath to refer to a portion of the document for partial inclusions.
- **XQuery** is to XML and XML Databases what SQL and PL/SQL are to relational databases: ways to access, manipulate and return XML.
- **XML Namespaces** enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring.
- **XML Signature** defines the syntax and processing rules for creating digital signatures on XML content.
- **XML Encryption** defines the syntax and processing rules for encrypting XML content.
- **XPointer** is a system for addressing components of XML-based internet media.

XML files may be served with a variety of Media types. RFC 3023 defines the types "application/xml" and "text/xml", which say only that the data is in XML, and nothing about its semantics. The use of "text/xml" has been criticized as a potential source of encoding problems but is now in the process of being deprecated.[6] RFC 3023 also recommends that XML-based languages be given media types beginning in "application/" and ending in "+xml"; for example "application/atom+xml" for Atom. This page discusses further XML and MIME.

Processing XML files

Three traditional techniques for processing XML files are:

- Using a programming language and the SAX API.
- Using a programming language and the DOM API.
- Using a transformation engine and a filter

More recent and emerging techniques for processing XML files are:

- Pull Parsing
- Non-Extractive Parsing (i.e. in-situ parsing)
- Data binding

Simple API for XML (SAX)

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

DOM

DOM is an interface-oriented Application Programming Interface that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed. DOM is supported in Java by several packages that usually come with the standard libraries. As the DOM specification is regulated by the World Wide Web Consortium, the main interfaces (Node, Document, etc.) are in the package org.w3c.dom.*, as well as some of the events and interfaces for other capabilities like serialization (output). The package com.sun.org.apache.xml.internal.serialize.* provides the serialization (output capacities) by implementing the appropriate interfaces, while the javax.xml.parsers.* package parses data to create DOM XML documents for manipulation.[7]

Transformation engines and filters

A filter in the Extensible Stylesheet Language (XSL) family can transform an XML file for displaying or printing.

- **XSL-FO** is a declarative, XML-based page layout language. An XSL-FO processor can be used to convert an XSL-FO document into another non-XML format, such as PDF.
- **XSLT** is a declarative, XML-based document transformation language. An XSLT processor can use an XSLT *stylesheet* as a guide for the conversion of the data tree represented by one XML document into another tree that can then be serialized as XML, HTML, plain text, or any other format supported by the processor.
- **XQuery** is a W3C language for querying, constructing and transforming XML data.
- **XPath** is a DOM-like node tree data model and path expression language for selecting data within XML documents. XSL-FO, XSLT and XQuery all make use of XPath. XPath also includes a useful function library.

Pull parsing

Pull parsing[8] treats the document as a series of items which are read in sequence using the Iterator design pattern. This allows for writing of recursive-descent parsers in which the structure of the code performing the parsing mirrors the structure of the XML being parsed, and intermediate parsed results can be used and accessed as local variables within the methods performing the parsing, or passed down (as method parameters) into lower-level methods, or returned (as method return values) to higher-level methods. Examples of pull parsers include StAX in the Java programming language, SimpleXML in PHP and System.Xml.XmlReader in .NET.

A pull parser creates an iterator that sequentially visits the various elements, attributes, and data in an XML document. Code which uses this 'iterator' can test the current item (to tell, for example, whether it is a start or end element, or text), and inspect its attributes (local name, namespace, values of XML attributes, value of text, etc.), and can also move the iterator to the 'next' item. The code can thus extract information from the document as it traverses it. The recursive-descent approach tends to lend itself to keeping data as typed local variables in the code doing the parsing, while SAX, for instance, typically requires a parser to manually maintain intermediate data within a stack of elements which are parent elements of the element being parsed. Pull-parsing code can be more straightforward to understand and maintain than SAX parsing code.

Non-extractive XML Processing API

Non-extractive XML Processing API is a new and emerging category of parsers that aim to overcome the fundamental limitations of DOM and SAX. The most representative is VTD-XML, which abolishes the object-oriented modeling of XML hierarchy and instead uses 64-bit Virtual Token Descriptors (encoding offsets, lengths, depths, and types) of XML tokens. VTD-XML's approach enables a number of interesting features/enhancements, such as high performance, low memory usage [9], ASIC implementation [10], incremental update [11], and native XML indexing [12] [13].

Data binding

Another form of XML Processing API is data binding, where XML data is made available as a custom, strongly typed programming language data structure, in contrast to the interface-oriented DOM. Example data binding systems include the Java Architecture for XML Binding (JAXB)[14] or C++ CodeSynthesis XSD[15][16].

Specific XML applications and editors

The native file format of OpenOffice.org, AbiWord, and Apple's iWork applications is XML. Some parts of Microsoft Office 2007 are also able to edit XML files with a user-supplied schema (but not a DTD), and Microsoft has released a file format compatibility kit for Office 2003 that allows previous versions of Office to save in the new XML based format. There are dozens of other XML editors available.

History

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet.[17][18] By the mid-1990s some practitioners of SGML had gained experience with the then-new World Wide Web, and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. Dan Connolly added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when Sun Microsystems engineer Jon Bosak developed a charter and recruited

collaborators. Bosak was well connected in the small community of people who had experience both in SGML and the Web.

XML was compiled by a working group of eleven members,[19] supported by an (approximately) 150-member Interest Group. Technical debate took place on the Interest Group mailing list and issues were resolved by consensus or, when that failed, majority vote of the Working Group. A record of design decisions and their rationales was compiled by Michael Sperberg-McQueen on December 4, 1997.[20] James Clark served as Technical Lead of the Working Group, notably contributing the empty-element "<empty/>" syntax and the name "XML". Other names that had been put forward for consideration included "MAGMA" (Minimal Architecture for Generalized Markup Applications), "SLIM" (Structured Language for Internet Markup) and "MGML" (Minimal Generalized Markup Language). The co-editors of the specification were originally Tim Bray and Michael Sperberg-McQueen. Halfway through the project Bray accepted a consulting engagement with Netscape, provoking vociferous protests from Microsoft. Bray was temporarily asked to resign the editorship. This led to intense dispute in the Working Group, eventually solved by the appointment of Microsoft's Jean Paoli as a third co-editor.

The XML Working Group never met face-to-face; the design was accomplished using a combination of email and weekly teleconferences. The major design decisions were reached in twenty weeks of intense work between July and November 1996, when the first Working Draft of an XML specification was published.[21] Further design work continued through 1997, and XML 1.0 became a W3C Recommendation on February 10, 1998.

XML 1.0 achieved the Working Group's goals of Internet usability, general-purpose usability, SGML compatibility, facilitation of easy development of processing software, minimization of optional features, legibility, formality, conciseness, and ease of authoring. Like its antecedent SGML, XML allows for some redundant syntactic constructs and includes repetition of element identifiers. In these respects, terseness was not considered essential in its structure.

Sources

XML is a profile of an ISO standard SGML, and most of XML comes from SGML unchanged. From SGML comes the separation of logical and physical structures (elements and entities), the availability of grammar-based validation (DTDs), the separation of data and metadata (elements and attributes), mixed content, the separation of processing from representation (processing instructions), and the default angle-bracket syntax. Removed were the SGML Declaration (XML has a fixed delimiter set and adopts Unicode as the document character set).

Other sources of technology for XML were the Text Encoding Initiative (TEI), which defined a profile of SGML for use as a 'transfer syntax'; HTML, in which elements were synchronous with their resource, the separation of document character set from resource encoding, the xml:lang attribute, and the HTTP notion that metadata accompanied the resource rather than being needed at the declaration of a link; and the Extended Reference Concrete Syntax (ERCS), from which XML 1.0's naming rules were taken, and which had introduced hexadecimal numeric character references and the concept of references to make available all Unicode characters.

Ideas that developed during discussion which were novel in XML, were the algorithm for encoding detection and the encoding header, the processing instruction target, the xml:space attribute, and the new close delimiter for empty-element tags.

Versions

There are two current versions of XML. The first, *XML 1.0*, was initially defined in 1998. It has undergone minor revisions since then, without being given a new version number, and is currently in its fourth edition, as published on August 16, 2006. It is widely implemented and still recommended for general use. The second, *XML 1.1*, was initially published on February 4, 2004, the same day as XML 1.0 Third Edition, and is currently in its second edition, as published on August 16, 2006. It contains features — some contentious — that are intended to make XML easier to use in certain cases[22] - mainly enabling the use of line-ending characters used on EBCDIC platforms, and the use of scripts and characters absent from Unicode 2.0. XML 1.1 is not very widely implemented and is recommended for use only by those who need its unique features. [23]

XML 1.0 and XML 1.1 differ in the requirements of characters used for element and attribute names: XML 1.0 only allows characters which are defined in Unicode 2.0, which includes most world scripts, but excludes those which were added in later Unicode versions. Among the excluded scripts are Mongolian, Cambodian, Amharic, Burmese, and others.

Almost any Unicode character can be used in the character data and attribute values of an XML 1.1 document, even if the character is not defined, aside from having a code point, in the current version of Unicode. The approach in XML 1.1 is that only certain characters are forbidden, and everything else is allowed, whereas in XML 1.0, only certain characters are explicitly allowed, thus XML 1.0 cannot accommodate the addition of characters in future versions of Unicode.

In character data and attribute values, XML 1.1 allows the use of more control characters than XML 1.0, but, for "robustness", most of the control characters introduced in XML 1.1 must

be expressed as numeric character references. Among the supported control characters in XML 1.1 are two line break codes that must be treated as whitespace. Whitespace characters are the only control codes that can be written directly.

There are also discussions on an XML 2.0, although it remains to be seen[*vague*] if such will ever come about. XML-SW (SW for skunk works), written by one of the original developers of XML, contains some proposals for what an XML 2.0 might look like: elimination of DTDs from syntax, integration of namespaces, XML Base and XML Information Set (*infoset*) into the base standard.

The World Wide Web Consortium also has an XML Binary Characterization Working Group doing preliminary research into use cases and properties for a binary encoding of the XML infoset. The working group is not chartered to produce any official standards. Since XML is by definition text-based, ITU-T and ISO are using the name *Fast Infoset[2]* for their own binary infoset to avoid confusion (see ITU-T Rec. X.891 | ISO/IEC 24824-1).

Patent claims

In October 2005 the small company Scientigo publicly asserted that two of its patents, U.S. Patent 5,842,213 and U.S. Patent 6,393,426 , apply to the use of XML. The patents cover the "modeling, storage and transfer [of data] in a particular *non-hierarchical*, non-integrated neutral form", according to their applications, which were filed in 1997 and 1999. Scientigo CEO Doyal Bryant expressed a desire to "monetize" the patents but stated that the company was "not interested in having us against the world." He said that Scientigo was discussing the patents with several large corporations.[24]

XML users and independent experts responded to Scientigo's claims with widespread skepticism and criticism. Some derided the company as a patent troll. Tim Bray described any claims that the patents covered XML as "ridiculous on the face of it".[25]

Critique of XML

Commentators have offered various critiques of XML, suggesting circumstances where XML provides both advantages and potential disadvantages.[26]

Advantages of XML

- It supports Unicode, allowing almost any information in any written human language to be communicated.
- It can represent common computer science data structures: records, lists and trees.
- Its self-documenting format describes structure and field names as well as specific values.
- The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.
- XML is heavily used as a format for document storage and processing, both online and offline.
- It is based on international standards.
- It can be updated incrementally.
- It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier.
- The hierarchical structure is suitable for most (but not all) types of documents.
- It is platform-independent, thus relatively immune to changes in technology.
- Forward and backward compatibility are relatively easy to maintain despite changes in DTD or Schema.
- Its predecessor, SGML, has been in use since 1986, so there is extensive experience and software available.

Disadvantages of XML

- XML syntax is redundant or large relative to binary representations of similar data,[27] especially with tabular data.
- The redundancy may affect application efficiency through higher storage, transmission and processing costs.[28][29]
- XML syntax is verbose, especially for human readers, relative to other alternative 'text-based' data transmission formats.[30][31]
- The hierarchical model for representation is limited in comparison to an object oriented graph.[32][33]
- Expressing overlapping (non-hierarchical) node relationships requires extra effort.[34]
- XML namespaces are problematic to use and namespace support can be difficult to correctly implement in an XML parser.[35]
- XML is commonly depicted as "self-documenting" but this depiction ignores critical ambiguities.[36][37]
- The distinction between content and attributes in XML seems unnatural to some and makes designing XML data structures harder.[38]
- Transformations, even identity transforms, result in changes to format (whitespace, attribute ordering, attribute quoting, whitespace around attributes, newlines). These problems can make diff-ing the XML source very difficult except via Canonical XML.
- Encourages non-relational data structures (data non-normalized)
- XML is very concrete and highly non-canonical. It introduces a very strong coupling between the actual representation chosen and the processing program (unlike relational storage and SQL)[*citation needed*]

XML in business world

- XBRL (eXtensible Business Reporting Language) based on XML is widely used in financial world. Since more and more investors and regulators call for financial transparency, XBRL technology could facilitate the process of transferring data, as well as for business reporting. Japanese banks are exemplified by using XBRL during their daily business.
- Web publishing uses XML to have one single source for creating and updating all content, which significantly saves time and reduce cost for companies when printing documents. In addition, governments in the world use XML for large documentation and printing maps. Airplane and Car manufactures may use XML technology to print maintenance booklets.
- Web searching- Since XML does not have fixed tags, the user is free to define the type of information in their work which means that other users will find their work easily when search on the Web.[39]

Standardization

In addition to the ISO standards mentioned above, other related document include

- ISO/IEC 8825-4:2002 *Information technology -- ASN.1 encoding rules: XML Encoding Rules (XER)*
- ISO/IEC 8825-5:2004 *Information technology -- ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1*
- ISO/IEC 9075-14:2006 *Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML)*

- ISO 10303-28:2007 *Industrial automation systems and integration -- Product data representation and exchange -- Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas*
- ISO/IEC 13250-3:2007 *Information technology -- Topic Maps -- Part 3: XML syntax*
- ISO/IEC 13522-5:1997 *Information technology -- Coding of multimedia and hypermedia information -- Part 5: Support for base-level interactive applications*
- ISO/IEC 13522-8:2001 *Information technology -- Coding of multimedia and hypermedia information -- Part 8: XML notation for ISO/IEC 13522-5*
- ISO/IEC 18056:2007 *Information technology -- Telecommunications and information exchange between systems -- XML Protocol for Computer Supported Telecommunications Applications (CSTA) Phase III*
- ISO/IEC 19503:2005 *Information technology -- XML Metadata Interchange (XMI)*
- ISO/IEC 19776-1:2005 *Information technology -- Computer graphics, image processing and environmental data representation -- Extensible 3D (X3D) encodings -- Part 1: Extensible Markup Language (XML) encoding*

- ISO/IEC 22537:2006 *Information technology -- ECMAScript for XML (E4X) specification*
- ISO 22643:2003 *Space data and information transfer systems -- Data entity dictionary specification language (DEDSL) -- XML/DTD Syntax*
- ISO/IEC 23001-1:2006 *Information technology -- MPEG systems technologies -- Part 1: Binary MPEG format for XML*
- ISO 24531:2007 *Intelligent transport systems -- System architecture, taxonomy and terminology -- Using XML in ITS standards, data registries and data dictionaries*

See also

- Ajax
- APML
- ASN.1
- asXML
- AutomationML
- CDATA section, the mechanism for including non-markup text in XML
- Comparison of layout engines (XML)
- DITA
- DocBook
- ebXML
- Binary XML
- Extensible Binary Meta Language
- Extensible Metadata Platform (XMP), used in graphics applications
- General purpose markup language
- JSON
- OGDL
- List of XML markup languages
- S-expression
- SAML
- Serialization
- Single source publishing
- SOAP
- Universal Business Language
- XBRL
- WBXML
- XML Catalog
- XML Data Binding
- XML/EDIFACT
- XML editor
- XML Information Set
- XML processing APIs:
  - DOM,
  - SAX,
- XML query language
- XML-RPC
- XML schema languages:
  - DTD,
  - RELAX NG,
  - Schematron,
  - DSDL
  - XML Forms Architecture
- XML Certification Program
- XRI, XDI
- YAML
- Category:XML-based standards
- W3C XML standards:
  - XForms
  - XHTML
  - XInclude
  - XLink
  - XML Base
  - XML Encryption
  - XML-infoset
  - DOM (the XML processing *reference model*).
  - XQuery
  - XML Schema
  - XML Signature
  - XPath
  - XPointer
  - XML Protocol: XMLP and SOAP.
  - WSDL, Web service
  - XSL and XSLT
  - LGML Linguistics Markup Language

- StAX,
- E4X
- VTD-XML

- Sedna

Notes and references

1. ^ It is often said to be a markup language itself. This is incorrect.[*citation needed*]
2. ^ Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau (September 2006). "Extensible Markup Language (XML) 1.0 (Fourth Edition) - Origin and Goals". World Wide Web Consortium. Retrieved on October 29, 2006.
3. ^ JSON and YAML are among other alternative text-based formats commonly described as lighter-weight and less verbose in comparison to XML. See Critique of XML in this article.
4. ^ XHTML is an attempt to simplify and improve the consistency of HTML, which was based on SGML.
5. ^ The XML specification does not (by itself) indicate rules for formatting and layout. This is in contrast to the HTML specification (for example) which does indicate such rules. Nevertheless, some XML documents may indicate graphical layout, display or formatting based on some other specification. An example is SVG, which is a modularized language for describing two-dimensional vector and mixed vector/raster graphics in XML.
6. ^ xml-dev - Fw: An I-D for text/xml, application/xml, etc
7. ^ Java Platform SE 6
8. ^ Push, Pull, Next! by Bob DuCharme, at XML.com
9. ^ Simplify XML processing with VTD-XML - Java World
10. ^ http://www.ximpleware.com/wp_SUN.pdf
11. ^ Cut, paste, split, and assemble XML documents with VTD-XML - Java World
12. ^ VTD+XML format spec
13. ^ Index XML documents with VTD-XML
14. ^ http://java.sun.com/xml/jaxb/
15. ^ http://www.artima.com/cppsource/xml_data_binding.html
16. ^ http://www.codesynthesis.com/products/xsd/
17. ^ Bray, Tim (February 2005). "A conversation with Tim Bray: Searching for ways to tame the world's vast stores of information". Association for Computing Machinery's "Queue site". Retrieved on April 16, 2006.
18. ^ *(1988) "Publishers, multimedia, and interactivity", Interactive multimedia. Cobb Group. ISBN 1-55615-124-1.*
19. ^ The working group was originally called the "Editorial Review Board." The original members and seven who were added before the first edition was complete, are listed

at the end of the first edition of the XML Recommendation, at http://www.w3.org/TR/1998/REC-xml-19980210.
20. ^ Reports From the W3C SGML ERB to the SGML WG And from the W3C XML ERB to the XML SIG
21. ^ Extensible Markup Language (XML)
22. ^ "Extensible Markup Language (XML) 1.1 (Second Edition) - Rationale and list of changes for XML 1.1". W3C. Retrieved on 2006-12-21.
23. ^ *Harold, Elliotte Rusty (2004). Effective XML. Addison-Wesley, 10-19. ISBN 0321150406.*
24. ^ Small company makes big claims on XML patents - CNET News.com
25. ^ XML co-inventor Bray responds to patent assault | Between the Lines | ZDNet.com
26. ^ (See e.g., XML-QL Proposal discussing XML benefits, When to use XML, "XML Sucks" on c2.com, Daring to Do Less with XML)
27. ^ *Harold, Elliotte Rusty (2002). Processing XML with Java(tm): a guide to SAX, DOM, JDOM, JAXP, and TrAX. Addison-Wesley. 0201771861.* XML documents are too verbose compared with binary equivalents.
28. ^ *Harold, Elliotte Rusty (2002). XML in a Nutshell: A Desktop Quick Reference. O'Reilly. 0596002920.* XML documents are very verbose and searching is inefficient for high-performance largescale database applications.
29. ^ However, the Binary XML effort strives to alleviate these problems by using a binary representation for the XML document. For example, the Java reference implementation of the Fast Infoset standard parsing speed is better by a factor 10 compared to Java Xerces, and by a factor 4 compared to the Piccolo driver, one of the fastest Java-based XML parser [1].
30. ^ *Bierman, Gavin (2005). Database Programming Languages: 10th international symposium, DBPL 2005 Trondheim, Norway. Springer. 3540309519.* XML syntax is too verbose for human readers in for certain applications. Proposes a dual syntax for human readability.
31. ^ Although many purportedly "less verbose" text formats actually cite XML as both inspiration and prior art. See e.g., http://yaml.org/spec/current.html, http://innig.net/software/sweetxml/index.html, http://www.json.org/xml.html.
32. ^ A hierarchical model only gives a fixed, monolithic view of the tree structure. For example, either actors under movies, or movies under actors, but not both.
33. ^ *Lim, Ee-Peng (2002). Digital Libraries: People, Knowledge, and Technology. Springer. 3540002618.* Discusses some of the limitation with fixed hierarchy. Proceedings of the 5th International Conference on Asian Digital Libraries, ICADL 2002, held in Singapore in December 2002.
34. ^ *Searle, Leroy F. (2004). Voice, text, hypertext: emerging practices in textual studies. University of Washington Press. 0295983051.* Proposes an alternative system for encoding overlapping elements.

35. ^ (See e.g., http://www-128.ibm.com/developerworks/library/x-abolns.html)
36. ^ "The Myth of Self-Describing XML". Retrieved on 2007-05-12.
37. ^ (See e.g., Use–mention distinction, Naming collision, Polysemy)
38. ^ "Does XML Suck?". Retrieved on 2007-12-15.(See "8. Complexity: Attributes and Content")
39. ^ http://www.sfu.ca/~ksc13/xml2.html

External links

See Picture License Information Here

Wikibooks has more on the topic of
XML

Specifications

- W3C XML homepage
- The XML 1.0 specification
- The XML 1.1 specification

Parsers

- AltovaXML free parser from Altova, also included in XMLSpy, MapForce, and StyleVision
- RomXML Embedded XML commercial toolkit written in ANSI-C.
- XDOM open-source XML parser (and DOM and XPath implementation) in Delphi/Kylix.
- XML resources at the Open Directory Project
- TinyXml Simple and small C++ XML parser.
- FoX fully validating XML parser library, written in Fortran.
- Intel_XSS XML parsing, validation, XPath, XSLT.
- sw8t.xml Lightweight, high-performance, intuitive JavaScript XML Parser. Includes API docs and developer's guide.

Sources

- Introduction to Generalized Markup by Charles Goldfarb
- Making Mistakes with XML by Sean Kelly
- Annex A of ISO 8879:1986 (SGML)
- The Multilingual WWW by Gavin Nicol
- Retrospective on Extended Reference Concrete Syntax by Rick Jelliffe
- XML Based languages
- Essential XML Quick Reference
- XML, Java and the Future of the Web by Jon Bosak
- XML tutorials in w3schools
- XML.gov

Retrospectives

- Thinking XML: The XML decade by Uche Ogbuji
- XML: Ten year anniversary by Elliot Kimber
- Closing Keynote, XML 2006 by Jon Bosak
- Five years later, XML... by Simon St. Laurent
- 23 XML fallacies to watch out for by Sean McGrath
- W3C XML is Ten!, XML 10 years press release

Papers

- *Lawrence A. Cunningham (2005). "Language, Deals and Standards: The Future of XML Contracts". Washington University Law Review. SSRN 900616.*

v • d • e

Standards of the World Wide Web Consortium

Recommendations	Canonical XML · CDF · CSS · DOM · HTML · MathML · OWL · PLS · RDF · RDF Schema · SISR · SMIL · SOAP · SRGS · SSML · SVG · SPARQL · Timed Text · VoiceXML · WSDL · XForms · XHTML · · XML Base · XML Events · XML Informa-

	tion Set · XML Schema (W3C) · XML Signature · XPath · XPointer · XQuery · XSL Transformations · XSL-FO · XSL · XLink
Notes	XHTML+SMIL · XAdES
Working Drafts	CCXML · CURIE · InkML · XFrames · XFDL · WICD · XHTML+MathML+SVG · XBL · XProc · HTML 5

EDIFACT

XML/EDIFACT is an Electronic Data Interchange format used in Business-to-business transactions. It allows EDIFACT message types to be used by XML systems.

EDIFACT is a formal language for machine readable description of electronic business documents. It uses a syntax close to delimiter separated files. This syntax was invented in the 1980s to keep files as small as possible. Because of the Internet boom around 2000, XML started to become the most widely supported file syntax. But for example, an invoice is still an invoice, containing information about buyer, seller, product, due amount. EDIFACT works perfectly from the content viewpoint, but many software systems struggle to handle its syntax. So combining EDIFACT vocabulary and grammar with XML syntax makes XML/EDIFACT.

The rules for XML/EDIFACT are defined by ISO TS 20625.

Use-cases

XML/EDIFACT is used in B2B scenarios as listed below.

1) Newer EAI or B2B systems, e.g. SAP XI, often cannot handle EDI (Electronic Data Interchange) syntax directly. Simple syntax converters do a 1:1 conversion before. Their input is an EDIFACT transaction file, their output an XML/EDIFACT instance file.

2) XML/EDIFACT keeps XML B2B transactions relatively small. XML element names derived from EDIFACT tags are much shorter and more formal than those derived from natural language. Such formal tags, taken from the "EDIFACT modelling language", are readable by B2B experts worldwide.

3) A company does not want to invest into new vocabularies from scratch. XML/EDIFACT reuses business content defined in UN/EDIFACT. Since 1987, the UN/EDIFACT library was enriched by global business needs for all sectors of industry, transport and public services. For XML, there is no such comprehensive vocabulary available.

4) Large companies can order goods from small companies via XML/EDIFACT. The small companies use XSL stylesheets to browse the message content in human readable forms.

Example 1: EDIFACT source code

A name and address (NAD) segment, containing customer ID and customer address, expressed in EDIFACT syntax:

NAD+BY+CST9955::91++Candy Inc+Sirup street 15+Sugar Town++55555'

Example 2: XML/EDIFACT source code

The same information content in an XML/EDIFACT instance file:

```
<S_NAD>
  <D_3035>BY</D_3035>
  <C_C082><D_3039>CST9955</D_3039><D_3055>91</D_3055></C_C082>
  <C_C080><D_3036>Candy Inc</D_3036></C_C080>
  <C_C059><D_3042>Sirup street 15</D_3042></C_C059>
  <D_3164>Sugar Town</D_3164><D_3251>55555</D_3251>
</S_NAD>
```

Example 3: XML/EDIFACT in Internet Explorer

The same XML/EDIFACT instance presented with help of an XSL stylesheet:

External links

- UN/EDIFACT Main Page
- Altova MapForce EDIFACT<->XML converter (also supports ANSI X12 and other file formats)
- Open Source EDIFACT<->XML converter
- Another Open Source EDIFACT<->XML converter

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

List of Contributors

Peterl
Dreftymac
Bunnyhop11
TimR
SmackBot
Hervegirod
Dkrms
Xompanthy
Versageek
TimBray
SnakeBot
Rr2bwreain
Rick Block
Peashy
Michael Hardy
Mah159
Lambiam
Idioma-bot
Gjlubbertsen
Dolcecars
Cspan64
Caomhin
Zorrobot
Ygramul
Vojta
Tony1
Terjen
Talktovalentine
Stwalkerster
Sderose
SashatoBot
Rror
Rjwilmsi
Red660
Osquar F
Nwbeeson
Netsnipe
Nigelj
Cybercobra
Sydius
VoABot II
Hogman500
DumZiBoT
Cbdorsett
VolkovBot
Treekids
Think777
SieBot
Rklawton
Philip Trueman
Montco
Mhkay
MER-C
Katalaveno
GrEp
Ghettoblaster
Dickpenn
Cipherynx
Ahoerstemeier
Yonkie
Whale plane
Toussaint
Tobias Bergemann
Teddyb
Svetovid
Shinkolobwe
Scottielad
SandiCastle
Robomaeyhem
RickBeton
PaulXemdli
Ohnoitsjamie
Nowa
NawlinWiki

Mthibault
Mola8sses
Mion
Mbell
Mayfare
Matthäus Wander
Kubigula
KickAssClown
Kamalakannanprogrammer
Jzhang2007
Jerazol
Jauerback
Jacobko
JAnDbot
Imars
Hicketyhicketyhack
Golwengaud
Frap
Fnielsen
Ewsers
Erikdw
Donmay12
Dino72
DeadEyeArrow
DarkFalls
Cydebot
ClueBot
Chininazu12
CanadianLinuxUser
Businessman332211
Bobo192
Bhadani
Allkeyword
Actam
AHMartin
Mrjmcneil
Mjb
Melab-1
Mbbradford
Maximus06
Lazynitwit
Kl4m
Keithgabryelski
Kai.Klesatschke
Jilplo Haggins
Jeenuv
Jacobolus
JForget
Iridescent
Highwayman65251
Hairy Dude
Glass of water
Francl
FatalError
Evaluist
Egandrews
Dlohcierekim
Dingbats
Davis685
Da monster under your bed
CptAnonymous
Clayoquot
Carewolf
C.M.Sperberg-McQueen
Bungopolis
Bobianite
Amire80
Aitias
Aadaam