Frequently Asked Questions about the Extensible Markup Language

The XML FAQ

Draft Version 1.21 (3 February 1998)

Maintained on behalf of the World Wide Web Consortium's XML Special Interest Group by **Peter Flynn** *(University College Cork)* with the collaboration of **Terry Allen** *()* **Tom Borgman** *(Harlequin Ltd)* **Tim Bray** *(Textuality, Inc)* **Robin Cover** *(Summer Institute of Linguistics)* **Christopher Maden** *(O'Reilly & Associates)* **Eve Maler** *(Arbortext, Inc)* **Peter Murray-Rust** *(Nottingham University)* **Liam Quin** *()* **Michael Sperberg-McQueen** *(University of Illinois at Chicago)* **Joel Weber** *(MIT)* **Murata Makoto** *(Fuji Xerox Information Systems)* and many other members of the XML Special Interest Group of the W3C as well as FAQ readers around the world. Please use the form at the end for any corrections or additions.

Revision history

Rev. 0.1 (31 January 1997) **[PF]**
First draft. Sample questions devised by participants.

Rev. 0.2 (3 February 1997) **[PF]**
Revised draft. Additional questions and answers.

Rev. 0.3 (17 February 1997) **[PF]**
Extensive revision following comments from the group. Changes to markup and organization.

Rev. 0.4 (23 February 1997) **[PF]**
Minor editorial changes

Rev. 0.5 (1 April 1997) **[PF]**
Added Multidoc Pro as SGML browser; question on XML math; fixed ambiguity in explanation of NETs; added JUMBO; ERB changes of March 26; more details of linking and tools; adding element declaration minimization to the forbidden list.

Rev. 1.0 (1 May 1997) **[PF]**
Added reference to ToC and printed URLs; added disclaimer at A6; combined old A11 with A5 to explain SGML/XML/HTML; clarified explanation of XML not replacing HTML at C1; added new course and conference at (new) A11; clarified B1, C4, C8; added FPI server at C12; removed examples in C13.

Rev. 1.1 (1 October 1997) **[PF]**
No more minimization parameters in element declarations; Parsers must now pass all white-space to the application; Everything is now case-sensitive, including all markup; A new proposal for stylesheets: XSL, which combines DSSSL and CSS in an XML format; Java[Script] and and metadata and their use in XML; Updated list of software; First XML book is published; New public mailing list XML-L

Most recent changes:

Rev. 1.2 (3 February 1998) **[PF]**

Added a Mac icon (thanks to Martin Winter and others); Removed Draft from references to the spec; Changed revision colours; The RMD is gone: replaced references to it with standalone; Updated some broken URLs; [1.21] minor edits to URLs and updates on translation; Added XUA to details of MIME types;

Paragraphs which have been added since the last version are shown prefixed with a pilcrow (¶). Paragraphs which have been changed since the last version are shown prefixed with a section sign (§). Paragraphs marked for future deletion but retained at the moment for information are prefixed with a plus/minus sign (±).

Summary

This document contains the most frequently-asked questions (with answers) about XML, the Extensible Markup Language. It is intended as a first resource for users, developers, and the interested reader, and should not be regarded as a part of the XML Specification.

Organization

The FAQ is divided into four parts: A. General, B. User, C. Author, and D. Developer . The questions are numbered independently within each section. As the numbering may therefore change with each version, comments and suggestions should refer to the version number (see Revision History above) as well as the Part and Question Number.

There is a form at the end of this document which you can use to submit bug reports, suggestions for improvement, and other comments relating to *this FAQ only*. Comments about the XML Specification itself should be sent to the W3C.

Availability

The SGML file for use with any conforming SGML system is available at `http://www.ucc.ie/xml/faq.sgml` (this can also be used online with SGML browsers like *Panorama* or *Multidoc Pro*; you can also download the DTD and stylesheet installation self-extractor for faster local access with these browsers, or the DTD set as ASCII files).

The same text is available in an HTML version for use with an HTML browser (*eg Netscape Navigator*, *Microsoft Internet Explorer*, *Spry Mosaic*, *NCSA Mosaic*, *Lynx*, *Opera*, *GNUscape Navigator etc*) at `http://www.ucc.ie/xml/`.

An XML version will be produced once the specification has been agreed and when DTDs and browsers are available to handle it.

A plaintext (ASCII) version is available from the Web and (eventually) by anonymous FTP to one of several FAQ repositories. The versions above are also available by electronic mail to the *WebMail* server (for users with email-only access).

For printed copies there are PostScriptTM versions for A4 and Letter sizes of paper.

The document is also available in oil-based toner on flattened dead trees by sending $10 (or equivalent) to the editor (email first to check currency and postal address).

Thanks to Murata Makoto for making this document available in Japanese: see `http://www.fxis.co.jp/DMS/sgml/xml/xmlfaq.html`

You can download the XML logo and an icon for your files in ICO (Microsoft

Windows), Mac, or XBM (X Window system) format.

A. General questions

A.1. What is XML?

XML is the ‘Extensible Markup Language’ (extensible because it is not a fixed format like HTML). It is designed to enable the use of SGML on the World Wide Web.

It's actually slightly misnamed: XML itself is not a single markup language: it's a metalanguage to let you design your own markup language. A regular markup language defines a way to describe information in a certain class of documents (*eg* HTML). XML lets you define your own customized markup languages for many classes of document. It can do this because it's written in SGML, the international standard metalanguage for markup languages.

A.2. What is XML for?

XML is designed ‘to make it easy and straightforward to use SGML on the Web: easy to define document types, easy to author and manage SGML-defined documents, and easy to transmit and share them across the Web.’

It defines ‘an extremely simple dialect of SGML which is completely described in the XML Specification. The goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML.’

‘For this reason, XML has been designed for ease of implementation, and for interoperability with both SGML and HTML’ [quotes from the XML spec].

A.3. What is SGML?

SGML is the Standard Generalized Markup Language (ISO 8879), the international standard for defining descriptions of the structure and content of different types of electronic document. There is an SGML FAQ at `http://www.infosys.utas.edu.au/info/sgmlfaq.txt` and the SGML Web pages are at `http://www.sil.org/sgml/`.

A.4. What is HTML?

HTML is the HyperText Markup Language (RFC 1866), a specific application of SGML used in the World Wide Web.

A.5. Aren't XML, SGML, and HTML all the same thing?

Not quite. SGML is the ‘mother tongue’, used for describing thousands of different document types in many fields of human activity, from transcriptions of ancient Sumerian scrolls to the technical documentation for stealth bombers, and from patients' clinical records to musical notation.

HTML is just one of these document types, the one most frequently used in the Web. It defines a single, fixed type of document with markup that lets you describe a common class of simple office-style report, with headings, paragraphs, lists, illustrations, *etc*, and some provision for hypertext and multimedia.

XML is an abbreviated version of SGML, to make it easier for you to define your own document types, and to make it easier for programmers to write programs to handle them. It omits the more complex and less-used parts of SGML in return for the benefits of being easier to write applications, easier to understand, and more suited to delivery and interoperability over the Web. But it is still SGML, and XML files may still be parsed and validated the same as any other SGML file (see the question on XML software).

Programmers may find it useful to think of XML as being SGML-- rather than HTML++.

A.6. Who is responsible for XML?

XML is a project of the World Wide Web Consortium (W3C), and the development of the specification is being supervised by their XML Working Group. A Special Interest Group of co-opted contributors and experts from various fields contributes comments and reviews by email.

XML is a public format: it is not a proprietary development of any company.

A.7. Why is XML such an important development?

It removes two constraints which are holding back Web developments:

1. dependence on a single, inflexible document type (HTML);
2. the complexity of full SGML, whose syntax allows many powerful but hard-to-program options.

XML simplifies the levels of optionality in SGML, and allows the development of user-defined document types on the Web.

A.8. How does XML make SGML simpler and still let you define your own document types?

To make SGML simpler, XML redefines some of SGML's internal values and parameters, and removes a large number of the more complex and sometimes less-used features which made it harder to write processing programs (see Appendix A of the XML specification).

But it retains all of SGML's structural abilities which let you define your own document type. It also introduces a new class of document which does not require you to use a predefined document type. See the questions about ‘valid’ and ‘well-formed’ documents, and how to define your own document types in the Developers' Section.

A.9. Why not just carry on extending HTML?

HTML is already overburdened with dozens of interesting but often incompatible inventions from different manufacturers, because it provides only one way of describing your information.

XML will allow groups of people or organizations to create their own customized markup languages for exchanging information in their domain (music, chemistry, electronics, hill-walking, finance, surfing, linguistics, mathematics, knitting, history, engineering, rabbit-keeping *etc*).

HTML is at the limit of its usefulness as a way of describing information, and while it will continue to play an important role for the content it currently represents, many new applications require a more robust and flexible infrastructure.

A.10. Why do we need all this SGML stuff? Why not just use *Word* or *Notes*?

Information on a network which connects many different types of computer has to be usable on all of them. Public information cannot afford to be restricted to one make or model or manufacturer, or to cede control of its data format to private hands. It is also helpful for such information to be in a form that can be reused in many different ways, as this can minimize wasted time and effort.

SGML is the international standard which is used for defining this kind of application, but those who need an alternative based on different software are entirely free to implement similar services using such a system, especially if they are for private use.

A.11. Where do I find more information about XML?

Online, there's the XML Specification and ancillary documentation available from the W3C; an XML section with an extensive list of online reference material in Robin Cover's SGML pages; and a summary and condensed FAQ from Tim Bray.

The items listed below are the ones the maintainer has been able to discover: please mail me if you come across others. Old items are retained here for reference at the moment: they will eventually expire.

- § Technology Appraisals Ltd are holding a seminar in London, England, on *XML ready for prime time?* on 6-8 April 1998. Details from Susan Dennington at TAL.
- ¶ James Tauber & Associates is running a full-day tutorial on XML at WWW7 on 14 April 1998.
- Peter Murray-Rust is preparing an XML/Java Virtual Course entitled *Scientific Information Components using Java and XML* Details are at `http://www.vsms.nottingham.ac.uk/vsms/java/advert/advert.txt`. The XML will be very low-level (*ie* well-formed only, balanced tags, and quoted attributes; no DTDs, entities, marked sections, catalogs, links, etc.) It concentrates on building element trees (including those from legacy files).
- The annual SGML Conference run by the Graphic Communications Association was renamed the SGML/XML Conference. SGML/XML '97 was held in Washington DC, 8-11 December 1997 (further details on the GCA's Web site).

There is a list of books and articles on XML in Robin Cover's SGML pages.

A.12. Where can I discuss implementation and development of XML?

There is a mailing list called `xml-dev` for those committed to developing components for XML. You can subscribe by sending a 1-line mail message to majordomo@ic.ac.uk saying:

```
subscribe xml-dev yourname@yoursite
```

The list is hypermailed for online reference at http://www.lists.ic.ac.uk/hypermail/xml-dev/.

Note that this list is for those people actively involved in developing resources for XML. It is *not* for general information about XML (see this FAQ and other sources) or for general discussion about SGML implementation and resources (see comp.text.sgml).

There is a general-purpose mailing list `XML-L` for public discussions: to subscribe, send a 1-line mail message to LISTSERV@listserv.hea.ie saying

```
subscribe XML-L forename surname
```

(substituting your own forename and surname). To unsubscribe, send a 1-line message to the same address saying

```
unsubscribe XML-L
```

Please Read The Fine Documentation which you will be sent when you join either mailing list, as it contains important information, particularly about what to do when your email address changes.

B. Users of SGML (including browsers of HTML)

B.1. Do I have to do anything to use XML?

Not yet. XML is still being developed, but there are already some pilot browsers, so you can experiment with them. When the specification is complete, more software should start to appear, and you may be able to download browsers and use them to browse the Web much as you do with current applications.

You can use the pilot browsers to look at some of the emerging XML material, such as Jon Bosak's Shakespeare plays and the molecular experiments of the Chemical Markup Language (CML). There are some more example sources listed at `http://www.sil.org/sgml/xml.html#examples`.

If you want to start preparations for writing your own XML, see the questions in the Authors' Section.

B.2. Why should I use XML instead of HTML?

- Authors and providers can design their own document types using XML, instead of being stuck with HTML. Document types can be explicitly tailored to an audience, so the cumbersome fudging that has to take place with HTML to achieve special effects should become a thing of the past: authors and designers will be free to invent their own markup elements;
- Information content can be richer and easier to use, because the hypertext linking abilities of XML are much greater than those of HTML.
- XML can provide more and better facilities for browser presentation and performance;
- It removes many of the underlying complexities of SGML in favor of a more flexible model, so writing programs to handle XML will be much easier than doing the same for full SGML.
- Information will be more accessible and reusable, because the more flexible markup of XML can be used by any XML software instead of being restricted to specific manufacturers as has become the case with HTML.
- Valid XML files are kosher SGML, so they can be used outside the Web as well, in an SGML environment (once the spec is stable and SGML software adopts it).

B.3. Where can I get an XML browser?

There are already some browsers emerging, but the XML specification is still new. As with HTML, there won't be just one browser, but many. However, because the potential number of different XML applications is not limited, no single browser should be expected to handle 100% of everything.

The generic parts of XML (*eg* parsing, tree management, searching, formatting, *etc*) are being combined into general-purpose browser libraries or toolkits to make it easier for developers to take a consistent line when writing XML applications. Such applications could then be customized by adding semantics for specific markets, or using languages like *Java* to develop plugins for generic browsers and have the specialist modules delivered transparently over the Web.

Netscape and Microsoft are both now developing XML facilities: some development work at Microsoft can be seen at `http://www.microsoft.com/msdn/sdk/inetsdk/help/`. The status at Netscape is unknown to me.

See also the notes on software for authors and developers, and the more detailed list on the XML pages in the SGML Web site at `http://www.sil.org/sgml/xml.html`.

B.4. Do I have to switch from SGML or HTML to XML?

No, existing SGML and HTML applications software will continue to work with existing files. But as with any enhanced facility, if you want to view or download and use XML files, you will need to add XML-aware software when it becomes available.

C. Authors of SGML (including writers of HTML)

Authors should also read the Developers' Section, which contains further information about the internals of XML files.

C.1. Does XML replace HTML?

No, XML itself does not replace HTML: instead, it provides an alternative by allowing you to define your own set of markup elements. HTML is expected to remain in common use for some time to come, and DTDs will be available in XML versions as well as the original SGML versions. XML is designed to make the writing of DTDs much simpler than with full SGML.

Work is going on to produce XML versions of HTML and other popular DTDs, but this may not take off until the specification for XML 1.0 is complete (targeted November 1997). Watch `comp.text.sgml` and `XML-L` for announcements.

C.2. What does an XML document look like inside?

The basic structure is very similar to most other applications of SGML, including HTML. XML documents can be very simple, with no document type declaration, and straightforward nested markup of your own design:

```
<?XML version="1.0" standalone="yes"?>
<conversation>
  <greeting>Hello, world!</greeting>
  <response>Stop the planet, I want to get off!</response>
</conversation>
```

Or they can be more complicated, with a DTD specified, and maybe an internal subset, and a more complex structure:

```
<?XML version="1.0" standalone="no" encoding="UTF-8"?>
<!DOCTYPE titlepage SYSTEM "http://www.frisket.org/dtds/typo.dtd"
[<!ENTITY % active.links "INCLUDE">]>
<titlepage>
  <white-space type="vertical" amount="36"/>
  <title font="Baskerville" size="24/30"
         alignment="centered">Hello, world!</title>
  <white-space type="vertical" amount="12"/>
  <!-- In some copies the following decoration is
        hand-colored, presumably by the author -->
  <image location="http://www.foo.bar/fleuron.eps" type="URL" alignment="centered"/>
  <white-space type="vertical" amount="24"/>
  <author font="Baskerville" size="18/22" style="italic">Munde Salutem</author>
</titlepage>
```

Or they can be anywhere between: a lot will depend on how you want to define your document type (or whose you use) and what it will be used for. See the question on valid and well-formed files.

C.3. How does XML handle white-space in my documents?

The SGML rules regarding white-space have been changed for XML, so *all* white-space, including linebreaks, TAB characters, and regular spaces, is passed by the parser *unchanged* to the application (browser, formatter, viewer, *etc*). This means:

- § 'insignificant' white-space between structural elements (those which can contain only other elements, not text data, sometimes called 'element content') *will* get passed to the application (under 'full' SGML this white-space is suppressed);
- § 'significant' white-space within elements which can contain text and markup mixed together ('mixed content' or PCDATA [parsed character data]) will still get passed to the application as before.

```
<chapter>
  <section>
    <title>
      My title for Section
1.
    </title>
    <para>
      ...
    </para>
  </section>
</chapter>
```

The parser must, however, still inform the application what white-space occurred in element content, if known. (Users of 'full' SGML may recognize that this information was not in the ESIS, but it *is* in the grove.) In the above example, the application will receive all the pretty-printing linebreaks, TABs, and spaces between the elements as well as those embedded in the section title. It is the function of the application (browser, formatter, viewer, *etc*) to decide which type of white-space to discard and which to retain.

C.4. Which parts of an XML document are case-sensitive?

All of an XML file is case-sensitive, both markup *and* text. This is significantly different from HTML and many other SGML document types. It was introduced to allow markup in non-Latin-alphabet scripts and to obviate problems with case-folding in scripts which are caseless.

- Element names (used in start-tags and end-tags) are case-sensitive: you must stick with whatever combination of upper- or lower-case you use to define them (either by usage or in a DTD);
- For well-formed files with no DTD, the *first occurrence* of an element name defines the casing. So you can't say `<BODY>...</body>`: upper- and lower-case must match; thus `<IMG/>` and `<img/>` are *two different elements*;
- Attribute names are also case-sensitive, on a per-element basis: for example `<PIC width="7in"/>` and `<PIC WIDTH="6in"/>` in the same file exhibit two *separate* attributes, because the different casings of `width` and `WIDTH` distinguish them;
- Attribute values are also case-sensitive. Character data values (*eg* `HRef="MyFile.SGML"`) are exactly as before, but ID and IDREF attributes are case-sensitive and no longer get folded to uppercase for comparisons;
- All entity names (`Á`), and your data content (your text), are

case-sensitive, exactly as before.

C.5. How can I make my existing HTML files work in XML?

Make them well-formed (see below). A DTD is optional in XML, but HTML files currently have to be DTDless anyway, because there is no XML version of the HTML DTD yet (on its way). It is necessary to convert existing HTML files to be well-formed because XML does not allow end-tag minimization as allowed in most HTML DTDs. Many HTML authoring tools already produce almost (but not quite) *well-formed* XML.

All XML documents must be well-formed (see below), but a DTD is optional. HTML files can be converted to a DTD-less form in XML. However, there cannot be XML versions of current SGML HTML DTDs and . Many HTML authoring tools already produce almost (but not quite) well-formed DTD-less XML.

If you have created your HTML files conforming to one of the several HTML Document Type Definitions (DTDs), and they validate OK, then they can be converted as follows:

- § replace the `DOCTYPE` declaration and any internal subset (basically everything within the first set of angled brackets `<!DOCTYPE HTML...>`) with the XML Declaration `<?XML version="1.0" standalone="yes"?>`
- change any `EMPTY` elements (*eg* `<ISINDEX>`, `<BASE>`, `<META>`, `<LINK>`, `<NEXTID>` and `<RANGE>` in the header, and `<IMG>`, `<BR>`, `<HR>`, `<FRAME>`, `<WBR>`, `<BASEFONT>`, `<SPACER>`, `<AUDIOSCOPE>`, `<AREA>`, `<PARAM>`, `<KEYGEN>`, `<COL>`, `<LIMITTEXT>`, `<SPOT>`, `<TAB>`, `<OVER>`, `<RIGHT>`, `<LEFT>`, `<CHOOSE>`, `<ATOP>`, and `<OF>` in the body) so that they end with '`/>`', for example `<IMG SRC="mypic.gif" alt="Picture"/>`
- ensure there are correctly-matched explicit end-tags for all non-empty elements; *eg* every `<P>` must have a `</P>`, *etc*: this can be automated by a normalizer program like *sgmlnorm* (part of *SP*) or a function in an editor like *Emacs/psgml*'s `sgml-normalize`;
- escape all markup characters (`<` and `&`) as `<` and `&`
- ensure all attribute values are in quotes;
- ensure all occurrences of all element names in start-tags *and* end-tags match with respect to upper- and lower-case and that they are consistent throughout the file;
- ensure all attribute names are similarly in a consistent case throughout the file.

Be aware that many HTML browsers may not accept XML-style `EMPTY` elements with the trailing slash, so the above changes are not backwards-compatible. An alternative is to add a dummy end-tag to all `EMPTY` elements, so `<IMG>` becomes `<IMG></IMG>`.

If you have a lot of valid HTML files, you could write a script in an SGML conversion system to do this (such as *Omnimark*, *Balise*, *SGMLC*, or a system using one of the SGML *Perl* libraries), or you could even use edit macros if you know what you're doing.

If your HTML files are invalid then they will almost certainly have to be converted manually, although if the deformities are regular and carefully constructed, the files may actually be almost well-formed, and you could write a program or script to do as described above. To test for invalidity and non-conformance, check the following:

- § do the files contain markup syntax errors? For example, are there any backslashes instead of forward slashes on end-tags; or elements which nest incorrectly (*eg* `<SAMP>an element which starts <EM>inside one element</SAMP> but ends outside it</EM>`)?
- do the files contain markup which conflicts with the HTML DTDs, such as headings inside list items, or list items outside list environments?
- § do the files use elements which are not in any DTD? Although this is easy to transform to a DTDless well-formed file (because you don't have to define elements in advance) most proprietary [browser-specific] extensions have never been formally defined, so it is often impossible to work out where they can meaningfully be used.

Markup which is valid but which is meaningless or void may need to be edited out before conversion (such as repeated empty paragraphs or linebreaks, empty tables, invisible 'spacing' GIFs *etc*: XML uses stylesheets, so you won't need any of these)

See the rules for 'well-formed' XML files for details of what you need to check in XML when converting.

Note there are XML versions of the HTML DTD in preparation:

- Ben Trafford is developing an XML version of HTML 4.2
- [details of others sought: please contact the editor]

C.6. If XML is just a subset of SGML, can I use XML files directly with SGML tools?

Yes, provided: A. the document has a valid Document Type Definition (DTD), *ie* the files are valid, not just well-formed; and B. you use software which knows about the features needed to support XML, such as the special form for `EMPTY` elements; some aspects of the SGML Declaration such as `NAMECASE GENERAL NO`; multiple attribute declarations.

At the moment there are few tools which handle XML files unchanged because of the format of these `EMPTY` elements, but this is changing. The *nsgmls* parser has an experimental XML conformance switch, and the first XML-specific editors and parsers are appearing (see the question on software).

The rules of ISO 8879 are up for minor amendments, some of which are to facilitate changes needed for Web-enablement.

C.7. I'm used to authoring and serving HTML. Can I learn XML easily?

Yes, very easily, but at the moment there is still a need for tutorials, simple tools, and more examples of XML documents. Well-formed XML documents may look similar to HTML except for some small but very important points of syntax.

As every user community can have their own document type defined, it should be much easier to learn, because element names can be picked for relevance.

C.8. Will XML be able to use non-Latin characters?

Yes, the XML Specification explicitly says XML uses ISO 10646, the international standard 31-bit character repertoire which covers most human (and some non-human) written languages. This is currently congruent with Unicode.

The spec says (2.2): 'All XML processors must accept the UTF-8 and UTF-16 encodings of ISO 10646 . . . '. UTF-8 is an encoding of Unicode into 8-bit characters: the first 128 are the same as ASCII, the rest are used to encode the rest of Unicode into sequences of between 2 and 6 bytes. UTF-8 in its single-octet form is therefore the same as ISO 646 IRV (ASCII), so you can continue to use ASCII for English or other unaccented languages using the Latin alphabet. Note that UTF-8 is incompatible with ISO 8859-1 (ISO Latin-1) after code point 126 decimal (the end of ASCII). UTF-16 is like UTF-8 but with a scheme to represent the next 16 planes of 64k characters as two 16-bit characters.

' . . . the mechanisms for signalling which of the two are in use, and for bringing other encodings into play, are [. . .] in the discussion of character encodings.' The XML Specification explains how to specify in your XML file which coded character set you are using.

Use of UCS-4 can only legally be specified in SGML or XML when the pending 'WebSGML Adaptations' to ISO 8879 come into force to enable numbers longer than eight digits to be used in the SGML Declaration.

'Regardless of the specific encoding used, any character in the ISO 10646 character set may be referred to by the decimal or hexadecimal equivalent of its bit string': so no matter which character set you personally use, you can still refer to specific individual characters from elsewhere in the encoded repertoire by using `&#dddd;` (decimal character code) or `&#UHHHH;` (hexadecimal character code, in uppercase).

The terminology can get confusing, as can the numbers: see the ISO 10646 Concept Dictionary.

C.9. What's a Document Type Definition (DTD) and where do I get one?

A DTD is usually a file (or several files to be used together) which contains a formal definition of a particular type of document. This sets out what names can be used for elements, where they may occur, and how they all fit together. For example, if you want a document type to describe `<LIST>`s which contain `<ITEM>`s, part of your DTD would contain something like

```
<!ELEMENT item (#pcdata)>
<!ELEMENT list (item)+>
```

This defines items containing text, and lists containing items. It's a formal language which lets processors automatically parse a document and identify where every element comes and how they relate to each other, so that stylesheets, navigators, browsers, search engines, databases, printing routines, and other applications can be used.

[Note that in XML, there are no minimization parameters (‘`-`’ and ‘`O`’ characters in element definitions between element name and content model), because all elements except empty ones must have both start-tag and end-tag present at all times.]

There are thousands of SGML DTDs already in existence in all kinds of areas (see the SGML Web pages for examples). Many of them can be downloaded and used freely; or you can write your own. As with any language, you need to learn it to do this: but XML is much simpler than full SGML: see the list of restrictions which shows what has been cut out. Existing SGML DTDs need to be converted to XML for use with XML systems: expect to see announcements soon of popular DTDs becoming available in XML format.

C.10. How will XML affect my document links?

The linking abilities of XML systems are much more powerful than those of HTML, so you'll be able to do much more with them. Existing `HREF`-style links will remain usable, but new linking technology is based on the lessons learned in the development of other standards involving hypertext, such as TEI and HyTime, which let you manage bidirectional and multi-way links, as well as links to a span of text (within your own or other documents) rather than to a single point. This is already implemented for SGML in browsers like *Panorama* and *Multidoc Pro*.

The XML Linking Specification (XLL) document contains a detailed specification. An XML link can be either a URL or a TEI-style Extended Pointer (‘Xptr’), or both. A URL on its own is assumed to be a resource (as with HTML); if an Xptr follows it, it is assumed to be a sub-resource of that URL; an Xptr on its own is assumed to apply to the current document.

An Xptr is always preceded by one of `#`, `?`, or `|`. The `#` and `?` mean the same as in HTML applications; the `|` means the sub-resource can be found by applying the Xptr to the resource, but the method of doing this is left to the application.

The TEI Extended Pointer Notation (EPN) is much more powerful than the ‘fragment address’ on the end of some URLs. For example, the word ‘Xptr’ two paragraphs back could be referred to as `http://www.ucc.ie/xml/faq.sgml#ID(faq-hypertext)CHILD(2,*)(4,*)`, meaning the fourth child object within the second child object after the element whose ID is `faq-hypertext`. Count the objects from the start of this question in the SGML version (which has the ID ‘`faq-hypertext`’):

1. the title of the question;

   ```
   <SECT2 ID="faq-hypertext">
   <TITLE>How will XML affect my document links?</TITLE>
   ```

2. the second paragraph:
   1. the character data from the start of the paragraph to the first item of markup:

      ```
      <PARA>The
      ```

   2. the markup item:

      ```
      <ULINK URL="http://www.w3.org/TR/WD-xml-link">XML Linking
      ```

```
Specification (XLL)</ULINK>
```

3. the next stretch of character data:

```
document contains a detailed specification. An XML link can
be either a URL or a TEI-style Extended Pointer (
```

4. the next markup item:

```
<LINK LINKEND="tei-link">Xptr</LINK>
```

If you view this file with *Panorama* or *MultiDoc Pro* you can click on the highlighted cross-reference button at the start of the example sentence, and it will display the locations in Extended Pointer Notation of all the links to it, including the word 'Xptr' mentioned. (Doing this in an HTML browser is not meaningful, as they do not support bidirectional linking or EPN.)

C.11. Can I do mathematics using XML?

Yes, if the document type you use provides for math. The mathematics-using community is developing software, and there is a MathML proposal at the W3C, which is a native XML application. It would also be possible to make XML fragments from the long-expired HTML3, HTML Pro, or ISO 12083 Math, or OpenMath, or one of your own making. Browsers which display simple math embedded in SGML already exist (*eg Panorama*, *Multidoc Pro*).

The sophistication could vary from math expressions like x_i through simple inline equations such as $E = mc^2$ to display equations like

$$\frac{\sum_{i=1}^{n} (x_i - \pi)^2}{n}$$

(If you are using an HTML browser to read this, the above equations may not be rendered correctly unless you have a math plugin for *Netscape* like IBM's *TechExplorer* which reads the embedded TeX equivalent.

C.12. How does XML handle metadata?

Because XML lets you define your own markup language, you can make full use of the extended hypertext features (see the question on Links) of XML to store or link to metadata in any format (*eg* Dublin Core, Warwick Framework, Resource Description Framework (RDF), and Platform for Internet Content Selection (PICS)).

There are no predefined elements in XML, because it is an architecture, not an application, so it is not part of XML's job to specify how or if authors should or should not implement metadata. You are therefore free to use any suitable method from simple attributes to the embedding of entire Dublin Core/Warwick Framework metadata records. Browser makers may also have their own architectural recommendations or methods to propose.

C.13. Can I use Java, ActiveX, *etc* in XML?

This depends on what facilities the browser makers implement. XML is about describing information; scripting languages and languages for embedded functionality are the software which enables the information to be manipulated at the user's end.

XML itself provides a way to define the markup needed to implement scripting languages: as a neutral standard it neither encourages not discourages their use, and does not favour one language over another, so the field is wide open. Developments are ongoing: see John Tigue's suggestions for standardising the API for Java in respect of XML.

Scripting languages *are* provided for in a proposal for an Extensible Style Language, XSL (see question on Stylesheets).

C.14. How do I control appearance?

The use of a stylesheet is implicit in XML. Some browsers may possibly provide simple default styles for popular elements like `<PARA>`, or `<LIST>` containing `<ITEM>`, but in general a stylesheet gives the author much better control of the layout. But as with any system where files can be viewed at random by arbitrary users, the author cannot know what resources (such as fonts) are on the user's system, so care is needed.

- The international standard for stylesheets for SGML documents is DSSSL, the Document Style and Semantics Specification Language (ISO 10179). This provides Scheme-like languages for stylesheets and document conversion, and is extensively implemented in the *Jade* formatter.
- The Cascading Stylesheet Specification (CSS) provides a simple syntax for assigning styles to elements, and has been implemented in HTML browsers.
- The Synex stylesheet DTD as already used in *Panorama* and *MultiDoc Pro*;
- A new Extensible Style Language (XSL) is being proposed for use specifically with XML. This uses XML syntax (a stylesheet is actually an XML file) and combines formatting features from both DSSSL and CSS (HTML) and has already attracted support from several major vendors.

It remains to be seen which ones browsers will implement.

D. Developers and Implementors (including WebMasters and server operators)

D.1. Where's the spec?

Right here (`http://www.w3.org/TR/WD-xml`). Includes the EBNF. There's also a version in Japanese at `http://www.fxis.co.jp/DMS/sgml/xml/wd-xml-lang.html` and `http://www.fxis.co.jp/DMS/sgml/xml/wd-xml-link.html`.

D.2. What are these terms 'DTDless', 'valid', and 'well-formed'?

Full SGML uses a Document Type Definition (DTD) to describe the markup (elements) available in any specific type of document. However, the design and construction of a DTD can be a complex and non-trivial task, so XML has been designed so it can be used either with or without a DTD. DTDless operation means you can invent markup without having to define it formally.

To make this work, a DTDless file in effect 'defines' its own markup, informally, by the existence and location of elements where you create them. But when an XML application such as a browser encounters a DTDless file, it needs to be able to understand the document structure as it reads it, because it has no DTD to tell it what to expect, so some changes have been made to the rules.

For example, HTML's `<IMG>` element is defined as '`EMPTY`': it doesn't have an end-tag. Without a DTD, an XML application would have no way to know whether or not to expect an end-tag for an element, so the concept of 'well-formed' has been introduced. This makes the start and end of every element, and the occurrence of `EMPTY` elements completely unambiguous.

'Well-formed' documents

All XML documents must be well-formed:

- § if there is no DTD in use, the document must start with a Standalone Document Declaration (SDD) saying so:

```
<?XML version="1.0" standalone="yes"?>
<foo>
  <bar>...<blort/>...</bar>
</foo>
```

- all tags must be balanced: that is, all elements which may contain character data must have both start- and end-tags present (omission is not allowed except for empty elements, see below);
- all attribute values must be in quotes (the single-quote character [the apostrophe] may be used if the value contains a double-quote character, and *vice versa*): if you need both, use `'` and `"`
- any `EMPTY` element tags (*eg* those with no end-tag like HTML's `<IMG>`, `<HR>`, and `<BR>` and others) must either end with '`/>`' or you have to make them non-`EMPTY`

by adding a real end-tag;

Example: `<BR>` would become either `<BR/>` or `<BR></BR>`.

- § there must not be any isolated markup characters (`<` or `&`) in your text data (*ie* they must be given as *`<`* and *`&`*), and the sequence `]]>` must be given as `]]>` if it does not occur as the end of a `CDATA` marked section;
- elements must nest inside each other properly (no overlapping markup, same rule as for all SGML);
- Well-formed files with no DTD may use attributes on any element, but the attributes must all be of type CDATA by default.

Well-formed XML files with no DTD are considered to have *`<`*, *`>`*, *`'`*, *`"`*, and *`&`* predefined and thus available for use even without a DTD. Valid XML files must declare them explicitly if they use them.

Valid XML

Valid XML files are those which have a Document Type Definition (DTD) like all other SGML applications, and which adhere to it. *They must also be well-formed*.

A valid file begins like any other SGML file with a Document Type Declaration, but may have an optional XML Declaration prepended:

```
<?XML version="1.0"?>
<!DOCTYPE advert SYSTEM "http://www.foo.org/ad.dtd">
<advert>
  <headline>...<pic/>...</headline>
  <text>...</text>
</advert>
```

The XML Specification defines an SGML Declaration for XML which is fixed for all instances. An XML version of the specified DTD must be accessible to the XML processor, either by being available locally (*ie* the user already has a copy on disk), or by being retrievable via the network. You can enable this by supplying the URL for the DTD in a System Identifier (as in the example above). It is possible (some people would say preferable) to supply a Formal Public Identifier, but if used, this must be *precede* the System Identifier, which must still be given:

```
<!DOCTYPE advert PUBLIC "-//Foo, Inc//DTD Advertisements//EN" "http://www.foo.org/ad.dtd"
```

The defaults for the other attributes of the XML Declaration are `VERSION="1.0"` and `ENCODING="UTF-8"`.

D.3. What else has changed between SGML and XML?

The principal changes are in what you can do in writing a Document Type Definition (DTD). To simplify the syntax and make it easier to write processing software, a large number of markup declaration options have been suppressed (see Appendix A of the XML Specification).

A new delimiter is permitted in Names (the colon) for use in experiments with namespaces (enabling DTDs to distinguish element source, ownership, or application). A colon may only appear in mid-name, though, not at the start or the end, and the syntax may change in a future version.

D.4. What XML software can I use today?

Details have been removed as they are now changing too rapidly to be duplicated in this FAQ: see the XML pages at `http://www.sil.org/sgml/xml.html`.

For browsers see the question on XML Browsers and the details of the `xml-dev` mailing list for software developers. Bert Bos keeps a list of some XML developments in bison, flex, perl and Python.

D.5. Do I have to change any of my server software to work with XML?

Only to serve up `.xml` files as the correct MIME type. MIME types of `text/xml` and `text/xsl` are usable, so for serving XML documents all that is needed is to edit the `mime-types` file (or its equivalent) and add the lines

```
text/xml                xml  XML
text/xsl       xsl  XSL
```

Since XML is designed to support stylesheets and sophisticated hyperlinking, XML documents will be accompanied by ancillary files such as DTDs, entity files, catalogs, stylesheets, etc, which may need their own MIME entry, and which require placing in the appropriate directories. XUA (XML User Agent), which is one of the planned deliverables of the XML WG, might provide a mechanism for packaging XML documents and XSL styles into a single message.

If you run scripts generating HTML, which you wish to work with XML, they will need to be modified to produce the relevant document type.

D.6. Can I still use server-side `INCLUDE`s?

Yes, so long as what they generate ends up as part of an XML-conformant file (*ie* either valid or just well-formed).

D.7. Can I (and my authors) still use client-side `INCLUDE`s?

The same rule applies as for server-side `INCLUDE`s, so you need to ensure that any embedded code which gets passed to a third-party engine (*eg SDQL* enquiries, *Java* `write`s, *LiveWire* requests, streamed content, *etc*) does not contain any characters which might be misinterpreted as XML markup (*ie* no angle brackets or ampersands): either use a `CDATA` marked section to avoid your XML application parsing the embedded code, or use the standard `<`, `>`, and `&` character entity references instead.

D.8. I'm trying to understand the XML Spec: why does SGML (and XML) have such difficult terminology?

For implementation to succeed, the terminology needs to be precise.

Example: ‘element’ and ‘tag’ are not synonymous: an element is a whole unit of information with its markup, and may consist of a start-tag alone (as in HTML's `<BR>`) or a start-tag and an end-tag *and the content which goes between them*; tags alone are simply the markers at the start and end of elements.

Sloppy terminology in specifications causes misunderstandings, so formal standards have to be phrased in formal terminology. This is not a formal document, and the astute reader may already have noticed it refers to ‘element names’ where ‘element type names’ is more correct; but the former is more widely understood.

Those new to SGML may want to read something like the *Gentle Introduction to SGML* chapter of the TEI.

D.9. Is there a Developer's API kit for XML?

Several are reported to be under development. The ones I have found so far are:

- The Language Technology Group has produced the *LT XML* toolkit (`http://www.ltg.ed.ac.uk/software/xml/`) and the DSSSL Syntax Checker (DSC: `http://www.ltg.ed.ac.uk/~ht/dsc-blurb.html`).
- [anyone with details of others please let me know]

The big SGML conversion and application development engines like *Balise*, *Omnimark*, and *SGMLC* are all working on XML versions. Details of SGML software of all kinds are on the SGML Web pages.

Response and query form

Illustration from Dale Dougherty's article in Web Review (courtesy of the publishers).

```
NOTATION SYSTEM "GIF"
ENTITY PUBLIC "+//SILMARIL//NONSGML THE XML FILES//EN"
```

Section and question:
`Widget: html.input`

1. New material

`Widget: html.input` New question, answer not known

`Widget: html.input` New question, with sample answer

2. Corrections to existing wording

`Widget: html.input` Correction to an existing question only

`Widget: html.input` Correction to an existing answer only

`Widget: html.input` Correction to both question and answer

3. Additional material

`Widget: html.input` Addition to an existing question only

`Widget: html.input` Addition to an existing answer only

`Widget: html.input` Addition to both question and answer

1. Question and Answer

`Widget: html.textarea`

`Widget: html.textarea`

2. Details

Your name:
`Widget: html.input`

Affiliation:
`Widget: html.input`

Email address:
`Widget: html.input`

`Widget: html.input`
`Widget: html.input`