Network Working Group
Request for Comments: 3470
BCP: 70
Category: Best Current Practice

S. Hollenbeck
VeriSign, Inc.
M. Rose
Dover Beach Consulting, Inc.
L. Masinter
Adobe Systems Incorporated

January 2003

Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols

Status of this Memo

Copyright Notice

Abstract

The Extensible Markup Language (XML) is a framework for structuring data. While it evolved from Standard Generalized Markup Language (SGML) -- a markup language primarily focused on structuring documents -- XML has evolved to be a widely-used mechanism for representing structured data.

There are a wide variety of Internet protocols being developed; many have need for a representation for structured data relevant to their application. There has been much interest in the use of XML as a representation method. This document describes basic XML concepts, analyzes various alternatives in the use of XML, and provides guidelines for the use of XML within IETF standards-track protocols.

Table of Contents

Conventions Used In This Document 2
1. Introduction and Overview 2
1.1 Intended Audience. 3
1.2 Scope 3
1.3 XML Evolution 3
1.4 XML Users, Support Groups, and Additional Information. 4
2. XML Selection Considerations 4
3. XML Alternatives 5

4. XML Use Considerations and Recommendations 7
4.1 XML Syntax and Well-Formedness 7
4.2 XML Information Set 7
4.3 Syntactic Restrictions 8
4.4 XML Declarations 9
4.5 XML Processing Instructions 9
4.6 XML Comments 10
4.7 Validity and Extensibility 10
4.8 Semantics as Well as Syntax. 12
4.9 Namespaces 12
4.9.1 Namespaces and Attributes. 13
4.10 Element and Attribute Design Considerations. 14
4.11 Binary Data and Text with Control Characters 16
4.12 Incremental Processing 16
4.13 Entity Declarations and Entity References 16
4.14 External References 17
4.15 URI Processing 17
4.16 White Space 18
4.17 Interaction with the IANA 19
5. Internationalization Considerations 19
5.1 Character Sets and Encodings 19
5.2 Language Declaration 20
5.3 Other Internationalization Considerations 20
6. IANA Considerations 21
7. Security Considerations 21
8. Acknowledgements 22
9. Normative References 22
10. Informative References 23
11. Authors' Addresses 27
12. Full Copyright Statement 28

Conventions Used In This Document

This document recommends, as policy, what specifications for Internet protocols -- and, in particular, IETF standards track protocol documents -- should include as normative language within them. The capitalized keywords "SHOULD", "MUST", "REQUIRED", etc. are used in the sense of how they would be used within other documents with the meanings as specified in BCP 14, RFC 2119 [1].

1. Introduction and Overview

The Extensible Markup Language (XML, [8]) is a framework for structuring data. While it evolved from the Standard Generalized Markup Language (SGML, [30]) -- a markup language primarily focused on structuring documents -- XML has evolved to be a widely-used mechanism for representing structured data in protocol exchanges. See "XML in 10 points" [47] for an introduction to XML.

1.1 Intended Audience

Many Internet protocol designers are considering using XML and XML fragments within the context of existing and new Internet protocols. This document is intended as a guide to XML usage and as IETF policy for standards track documents. Experienced XML practitioners will likely already be familiar with the background material here, but the guidelines are intended to be appropriate for those readers as well.

1.2 Scope

This document is intended to give guidelines for the use of XML content within a larger protocol. The goal is not to suggest that XML is the "best" or "preferred" way to represent data; rather, the goal is to lay out the context for the use of XML within a protocol once other factors point to XML as a possible data representation solution. The Common Name Resolution Protocol (CNRP, [24]) is an example of a protocol that would be addressed by these guidelines if it were being newly defined. This document does not address the use of protocols like SMTP or HTTP to send XML documents as ordinary email or web content.

There are a number of protocol frameworks already in use or under development which focus entirely on "XML protocol" -- the exclusive use of XML as the data representation in the protocol. For example, the World Wide Web Consortium (W3C) is developing an XML Protocol framework based on SOAP ([45] and [46]). The applicability of such protocols is not part of the scope of this document.

In addition, there are higher-level representation frameworks, based on XML, that have been designed as carriers of certain classes of information; for example, the Resource Description Framework (RDF, [38]) is an XML-based representation for logical assertions. This document does not provide guidelines for the use of such frameworks.

1.3 XML Evolution

XML 1.0 was originally published as a W3C recommendation in February 1998 [35], and was revised in a 2nd edition [8] in October 2000. Several additional facilities have also been defined that layer on the base specification. Although these additions are designed to be consistent with XML 1.0, they have varying levels of stability, consensus, and implementation. Accordingly, this document identifies the major evolutionary features of XML and makes suggestions as to the circumstances in which each feature should be used.

1.4 XML Users, Support Groups, and Additional Information

There are many XML support groups, with some devoted to the entire XML industry [51], some devoted to developers [52], some devoted to the business applications of XML [53], and many, many groups devoted to the use of XML in a particular context.

It is beyond the scope of this document to provide a comprehensive list of referrals. Interested readers are directed to the three references above as starting points, as well as their favorite Internet search engine.

2. XML Selection Considerations

XML is a tool that provides a means towards an end. Choosing the right tool for a given task is an essential part of ensuring that the task can be completed in a satisfactory manner. This section describes factors to be aware of when considering XML as a tool for use in IETF protocols:

1. XML is a meta-markup language that can be used to define markup languages for specific domains and problem spaces.

2. XML provides both logical structure and physical structure to describe data. Data framing is built-in.

3. XML instances can be validated against the formal definition of a protocol specification.

4. XML supports internationalization.

5. XML is extensible. Unlike some other markup languages (such as HTML), new tags (and thus new protocol elements) can be defined without requiring changes to XML itself.

6. XML is still evolving. The formal specifications are still being influenced and updated as use experience is gained and applied.

7. XML does not provide native mechanisms to support detailed data typing. Additional mechanisms (such as those described in Section 4.7) are required to specify abstract protocol data types.

8. XML is text-based, so XML fragments are easily created, edited, and managed using common utilities. Further, being text-based means it more readily supports incremental development,

debugging, and logging. A simple "canned" XML fragment can be embedded within a program as a string constant, rather than having to be constructed.

9. Binary data has to be encoded into a text-based form to be represented in XML.

10. XML is verbose when compared with many other structured data representation languages. A representation with element extensibility and human readability typically requires more bits when compared to one optimized for efficient machine processing.

11. XML implementations are still relatively new. As designers and implementers gain experience, it is not uncommon to find defects in early and current products.

12. XML support is available in a large number of software development utilities, available in both open source and proprietary products.

13. XML processing speed can be an issue in some environments. XML processing can be slower because XML data streams may be larger than other representations, and the use of general purpose XML parsers will add a software layer with its own performance costs (though these costs can be reduced through consistent use of an optimized parser). In some situations, processing XML requires examining every byte of the entire XML data stream, with higher overhead than with representations where uninteresting segments can be skipped.

3. XML Alternatives

This document focuses on guidelines for the use of XML. It is useful to consider why one might use XML as opposed to some other mechanism. This section considers some other commonly used representation mechanisms and compares XML to those alternatives.

For many fundamental protocols, the extensibility requirements are modest, and the performance requirements are high enough that fixed binary data blocks are the appropriate representation; mechanisms such as XML merely add bloat. RFC 3252 [23] describes a humorous example of XML as protocol bloat.

In addition, there are other representation and extensibility frameworks that have been used successfully within communication protocols. For example, Abstract Syntax Notation 1 (ASN.1) [28] along with the corresponding Basic Encoding Rules (BER, [29]) are part of the OSI communication protocol suite, and have been used in

many subsequent communications standards (e.g., the ANSI Information Retrieval protocol [27] and the Simple Network Management Protocol (SNMP, [13]). The External Data Representation (XDR, [14]) and variations of it have been used in many other distributed network applications (e.g., the Network File System (NFS) protocol [22]). With some ASN.1 encoding types, data types are explicit in the representation, while with XDR, the data types of components are described externally as part of an interface specification.

Many other protocols use data structures directly (without data encapsulation) by describing the data structure with Backus Normal Form (BNF, [25]); many IETF protocols use an Augmented Backus-Naur Form (ABNF, [16]). The Simple Mail Transfer Protocol (SMTP, [21]) is an example of a protocol specified using ABNF.

ASN.1, XDR, and BNF are described here as examples of alternatives to XML for use in IETF protocols. There are other alternatives, but a complete enumeration of all possible alternatives is beyond the scope of this document.

Other representation methods may differ from XML in several important ways:

Text Encoding and character sets: the character encoding used to represent a formal specification. XML defines a consistent character model based on the Universal Character Set (UCS, [31] and [33]), and requires that XML parsers accept at least UTF-8 [4] and UTF-16 [20], and allows for other encodings. While ASN.1 and XDR may carry strings in any encoding, there is no common mechanism for defining character encodings within them. Typically, ABNF definitions tend to be defined in terms of octets or characters in ASCII.

Data Encoding: XML is defined as a sequence of characters, rather than a sequence of bytes. XML Schema [42] includes mechanisms for representing some data types (integer, date, array, etc.) but many binary data types are encoded in Base64 [15] or hexadecimal. ASN.1 and XDR have rich mechanisms for encoding a wide variety of data types.

Extensibility: XML has a rich extensibility model such that XML specifications can frequently be versioned independently. Specifications can be extended by adding new element names and attributes (if done compatibly); other extensions can be added by defining new XML namespaces [9], though there is no standard mechanism in XML to indicating whether or not new extensions are mandatory to recognize. Similarly, there are several techniques available to extend ASN.1 specifications. XDR specifications tend to not be independently extensible by different parties because the

framing and data types are implicit and not self-describing. The extensibility of BNF-based protocol elements needs to be explicitly planned.

Legibility of protocol elements: As noted above, XML is text-based, and thus carries the advantages (and disadvantages) of text-based protocol elements. Typically this is shared with (A)BNF-defined protocol elements. ASN.1 and XDR use binary encodings which are not easily human readable.

4. XML Use Considerations and Recommendations

This section notes several aspects of XML and makes recommendations for use. Since the 1998 publication of XML version 1 [35], an editorial second edition [8] was published in 2000; this section refers to the second edition.

4.1 XML Syntax and Well-Formedness

XML [8] is defined in terms of a concrete syntax: a sequence of characters, using the characters "<", "=", "&", etc. as delimiters. An instance is XML if and only if it is well-formed, i.e., all character and markup data conforms to the structural rules defined in section 2.1 of [8].

Character and markup data that is not well-formed is not XML; well-formedness is the basis for syntactic compatibility with XML. Without well-formedness, all of the advantages of using XML disappear. For this reason, it is recommended that protocol specifications explicitly require XML well-formedness ("MUST be well-formed").

The IETF has a long-standing tradition of "be liberal in what you accept" that might seem to be at odds with this recommendation. Given that XML requires well-formedness, conforming XML parsers are intolerant of well-formedness errors. When specifying the handing of erroneous XML protocol elements, a protocol design must never recommend attempting to partially interpret non-well-formed instances of an element which is required to be XML. Reasonable behaviors in such a scenario could include attempting retransmission or aborting an in-progress session.

4.2 XML Information Set

In addition to the concrete syntax of XML, there is an abstract model of XML content known as the "Information Set" (infoset) [37]. One might think of an XML parser as consuming the concrete syntax and producing an XML Information Set for further processing.

In typical use of XML, the definition of allowable XML documents is often defined in terms of the Information Set of the XML and not the concrete syntax. The notion is that any syntactic representation which yielded the same information set would be treated equivalently.

It some cases, protocols have been defined solely in terms of the XML Information Set, or by allowing other concrete syntax representations. However, since the context of XML embedded within other Internet protocols requires an unambiguous definition of the concrete syntax, defining an XML protocol element in terms of its XML Information Set alone and allowing other concrete syntax representations is out of scope for this document.

4.3 Syntactic Restrictions

In some circumstances a protocol designer may be tempted to define an XML-based protocol element as "XML", but at the same time imposing additional restrictions beyond those imposed by the XML recommendation itself -- for example, restricting the document character encoding, or avoiding CDATA sections, character entity references, imposing additional restrictions on use of white space, etc. The general category of restrictions addressed by this section are ones that would allow some but not other of the set of syntactic representations which have the same canonical representation according to canonical XML described in RFC 3076 [6].

Making these kinds of restrictions in a protocol definition may have the disadvantage that an implementer of the protocol may not be able to use an otherwise conforming XML processor to parse the XML-based protocol elements. In some cases, the motivation for subsetting XML is to allow implementers to build special-purpose processors that are lighter weight than a full-scale conforming XML processor. There are a number of good, conforming XML parsers that are small, fast, and free, while special-purpose processors have frequently been known to fail to handle some cases of legal XML syntax.

In general, such syntactic restrictions should be avoided. In circumstances where restrictions on the variability of the syntactic representation of XML is necessary for one reason or another, designers should consider using "Canonical XML" [6] as the definition of the protocol element, since all such variability has been removed. Some specific issues are discussed in Section 4.4, Section 4.13, and Section 5.1 below.

4.4 XML Declarations

An XML declaration (defined in section 2.8 of [8]) is a small header at the beginning of an XML data stream that indicates the XML version and the character encoding used. For example,

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

specifies the use of XML version 1 and UTF-8 character encoding.

In some uses of XML as an embedded protocol element, the XML used is a small fragment in a larger context, where the XML version is fixed at "1.0" and the character encoding is known to be "UTF-8". In those cases, an XML declaration might add extra overhead. In cases where the XML is a larger component which may find its way alone as an external entity body (transported as a MIME message, for example), the XML declaration is an important marker and is useful for reliability and extensibility. The XML declaration is also an important marker for character set/encoding (see Section 5.1), if any encoding other than UTF-8 or UTF-16 is used. Note that in the case of UTF-16, XML requires that the entity starts with a Byte Order Mark (BOM), which is not part of the character data. Note that the XML Declaration itself is not part of the XML document's Information Set.

Protocol specifications must be clear about use of XML declarations. XML [8] notes that "XML documents should begin with an XML declaration which specifies the version of XML being used." In general, an XML declaration should be encouraged ("SHOULD be present") and must always be allowed ("MAY be sent"). An XML declaration should be required in cases where, if allowed, the character encoding is anything other than UTF-8 or UTF-16.

4.5 XML Processing Instructions

An XML processing instruction (defined in section 2.6 of [8]) is a component of an XML document that signals extra "out of band" information to the receiver; a common use of XML processing instructions are for document applications. For example, the XML2RFC application used to generate this document and described in RFC 2629 [19] supports a "table of contents" processing instruction:

```
<?rfc toc="yes"?>
```

As described in section 2.6 of [8], processing instructions are not part of the document's character data, but must be passed through to the application. As a consequence, it is recommended that processing instructions be ignored when encountered in normal protocol processing. It is thus also recommended that processing instructions

not be used to define normative protocol data structures or extensions for the following reasons:

- Processing instructions are not namespace aware; there is no way to qualify a processing instruction target with a namespace.
- Processing instruction use can not be constrained by most schema languages,
- Character references are not recognized within a processing instruction.
- Processing instructions don't have any XML-defined structure beyond the division between the target and everything else. This means that applications typically have to parse the content of the processing instruction in a system-dependent way; if the content was provided within an element instead, the structure could be expressed in the XML and the parsing could be done by the XML parser.

4.6 XML Comments

An XML comment (defined in section 2.5 of [8]) is a component of an XML document that provides descriptive information that is not part of the document's character data. XML comments, like comments used in programming languages, are often used to provide explanatory information in human-understandable terms. An example:

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

XML comments can be ignored by conformant processors. As a consequence, it is strongly recommended that comments not be used to define normative protocol data structures or extensions. It is thus also strongly recommended that comments be ignored if encountered in normal protocol processing.

4.7 Validity and Extensibility

One important value of XML is that there are formal mechanisms for defining structural and data content constraints; these constrain the identity of elements or attributes or the values contained within them. There is more than one such formalism:

- A "Document Type Definition" (DTD) is defined in section 2.8 of [8]; the concept came from a similar mechanism for SGML. There is significant experience with using DTDs, including in IETF protocols.

- XML Schema (defined in [41] and [42]) provides additional features to allow a tighter and more precise specification of allowable protocol syntax and data type specifications.

- There are also a number of other mechanisms for describing XML instance validity; these include, for example, Schematron [49] and RELAX NG [48]. Part 2 of the ISO/IEC Document Schema Definition Language (DSDL, [32]) standard is based on RELAX NG.

There is ongoing discussion (and controversy) within the XML community on the use and applicability of various validity constraint mechanisms. The choice of tool depends on the needs for extensibility or for a formal language and mechanism for constraining permissible values and validating adherence to the constraints.

There are cases where protocols have defined validity using one or another validity mechanism, but the protocol definitions have not insisted that all corresponding protocol elements be "valid". The decision depends in part on the design for protocol extensibility. Each formalism has different ways of allowing for future extensions; in addition, a protocol design may have its own versioning mechanism, way of updating the schema, or pointing to a new one. For example, the use of XML namespaces (Section 4.9) with XML Schema allows other kinds of extensibility without compromising schema validity.

No matter what formalism is chosen, there are usually additional syntactic constraints, and inevitably additional semantic constraints, on the validity of XML elements that cannot be expressed in the formalism.

This document makes the following recommendations for the definition of protocols using XML:

- Protocols should use an appropriate formalism for defining validity of XML protocol elements.

- Protocols may or may not insist that all corresponding protocol elements be valid, according to the validity mechanism chosen; in either case, the extensibility design should be clear. What happens if the data is not valid?

- As described in Section 3 there is no standard mechanism in XML for indicating whether or not new extensions are mandatory to recognize. XML-based protocol specifications should thus explicitly describe extension mechanisms and requirements to recognize or ignore extensions.

An idealized model for XML processing might first check for well-formedness; if OK, apply the primary formalism and, if the instances "passes", apply the other constraints so that the entire set (or as much as is machine processable) can be checked at the same time.

However, it is reasonable to allow conforming implementations to avoid doing validation at run-time and rely instead on ad-hoc code to avoid the higher expense, for example, of schema validation, especially given that there will likely be additional hand-crafted semantic validation.

4.8 Semantics as Well as Syntax

While the definition of an XML protocol element using a validity formalism is useful, it is not sufficient. XML by itself does not supply semantics. Any document defining a protocol element with XML MUST also have sufficient prose in the document describing the semantics of whatever XML the document has elected to define.

4.9 Namespaces

XML namespaces, defined in [9], provide a means of assigning markup to a specific vocabulary. If two elements or attributes from different vocabularies have the same name, they can be distinguished unambiguously if they belong to different namespaces. Additionally, namespaces provide significant support for protocol extensibility as they can be defined, reused, and processed dynamically.

Markup vocabulary collisions are very possible when namespaces are not used to separate and uniquely identify vocabularies. Protocol definitions should use existing XML namespaces where appropriate. When a new namespace is needed, the "namespace name" is a URI that is used to identify the namespace; it’s also useful for that URI to point to a description of the namespace. Typically (and recommended practice in W3C) is to assign namespace names using persistent http URIs.

In the case of namespaces in IETF standards-track documents, it would be useful if there were some permanent part of the IETF’s own web space that could be used for this purpose. In lieu of such, other permanent URIs can be used, e.g., URNs in the IETF URN namespace (see [11] and [12]). Although there are instances of IETF specifications creating new URI schemes to define XML namespaces, this practice is strongly discouraged.

4.9.1 Namespaces and Attributes

There is a frequently misunderstood aspect of the relationship between unprefixed attributes and the default XML namespace - the natural assumption is that an unprefixed attribute is qualified by the default namespace, but this is not true. Rather, the unprefixed attribute belongs to no namespace at all. Thus, in the following example:

```
<ns1:fox a="xxx" ns1:b="qqq"
 xmlns="http://example.org"/>
<fox a="xxx" ns1:b="qqq"
 xmlns="http://example.org" xmlns:ns1="http://example.org"/>
```

the attribute "a" is in no namespace, while "ns1:b" is the same namespace as the containing element. A specific description of the relationship between default namespaces and attributes can be found in section 5.2 of [9]. The practical implication of the relationship between namespaces and attributes is that care must be taken to ensure that no element contains multiple attributes that have identical names or have qualified names with the same local part and with prefixes which have been bound to namespace names that are identical.

In XML applications, the choice between prefixed and non-prefixed attributes frequently is based on whether they always appear inside elements of the same namespace (in which case non-prefixed and thereby non-namespaced names are used) or whether it’s required that they can be applied to elements in other arbitrary namespaces (in which case a prefixed name is used). Both situations occur in the XSLT [43] language: while attributes are unprefixed when they occur inside elements in the XSLT namespace, such as:

```
<xsl:value-of select="."/>
```

they are prefixed when they appear in non-XSLT elements, such as the "xsl:version" attribute when using "literal result element stylesheets":

```
<html xsl:version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns="http://www.w3.org/TR/xhtml1/strict">
  <head>
    <title>Expense Report Summary</title>
  </head>
  <body>
    <p>Total: <xsl:value-of select="exp-rep/total"/></p>
  </body>
</html>
```

4.10 Element and Attribute Design Considerations

XML provides much flexibility in allowing a designer to use either elements, attributes, or element content to carry data. This section gives a flavor of the design considerations; there is much written about this in the XML literature. Consistent use of elements, attributes, and values is an important characteristic of a sound design.

Attributes are generally intended to contain meta-data that describes the element, and as such they are subject to the following restrictions:

- Attributes are unordered,
- There can be no more than one instance of a given attribute within a given element, though an attribute may contain several values separated by white space ([8], section 2.3 and 3.3.1),
- Attribute values can have no internal XML markup for providing internal structure, and
- Attribute values are normalized ([8], section 3.3) before processing

Consider the following example that describes an IP address using an attribute to describe the address value:

```
<address addrType="ipv4">10.1.2.3</address>
```

One might encode the same information using an <addrType> element instead of an "addrType" attribute:

```
<address>
  <addrType>ipv4</addrType>
  <value>10.1.2.3</value>
</address>
```

Another way of encoding the same information would be to use markup for the "addrType":

```
<address>
  <addrType><ipv4/></addrType>
  <value>10.1.2.3</value>
</address>
```

Choosing between these designs involves tradeoffs concerning, among other considerations, the likely extensibility patterns and the ability of the formalism to constrain the values appropriately. In the first example, the attribute can be thought of as meta-data to the element which it modifies, and provides for a kind of "element extensibility". The third example allows for a different kind of extensibility: the "ipv4" space can be extended using other namespaces, and the <ipv4> element can include additional markup.

Many protocols include parameters that are selected from an enumerated set of values. Such enumerated values can be encoded as elements, attributes, or strings within element values. Any protocol design should consider how the set of enumerated values is to be extended: by revising the protocol, by including different values in different XML namespaces, or by establishing an IANA registry (as per RFC 2434 [18]). In addition, a common practice in XML is to use a URI as an XML attribute value or content.

Languages that describe syntactic validity (including XML Schema and DTDs) often provide a mechanism for specifying "default" values for an attribute. If an element does not specify a value for the attribute, then the "default" value is used. The use of default values for attributes is discouraged by this document. Although the use of this feature can reduce both the size and clutter of XML documents, it has a negative impact on software which doesn’t know the document’s validity constraints (e.g., for packet tracing or digital signature).

4.11 Binary Data and Text with Control Characters

XML is defined as a character stream rather than a stream of octets. There is no way to embed raw binary data directly within an XML data stream; all binary data must be encoded as characters. There are a number of possible encodings; for example, XML Schema [42] defines encodings using decimal digits for integers, Base64 [15], or hexadecimal digits. In addition, binary data might be transmitted using some other communication channel, and referenced within the XML data itself using a URI.

Protocols that need a container that can hold both structural data and large quantities of binary data should consider carefully whether XML is appropriate, since the Base64 and hex encodings are inefficient. Otherwise, protocols should use the mechanisms of XML Schema to represent binary data; the Base64 encoding is best for larger quantities of data.

XML does not allow "control" characters (0x00-0x1F) except for TAB (0x09), CR (0x0A), and LF (0x0D). They can not be specified even using character entity references. There is currently no common way of encoding them within what is otherwise ordinary text. This means that strings that might be considered "text" within an ABNF-defined protocol element may need to be treated as binary data within an XML representation, or some other encoding mechanism might need to be invented.

4.12 Incremental Processing

In some situations, it is possible to incrementally process an XML document as each tag is received; this is analogous to the process by which browsers incrementally render HTML pages as they are received. Note that incremental processing is difficult to implement if interspersed across multiple interactions. In other words, if a protocol requires incremental processing across both directions of a bidirectional stream, then it may place an unusual burden on protocol implementers.

4.13 Entity Declarations and Entity References

In addition to its role as a validity mechanism, an XML DTD provides a facility for "entity declarations" ([8], section 4.2). An entity declaration defines, in the DTD, a kind of macro capability where an "entity reference" may be used to call up and include the content of the entity declaration.

This feature adds complexity to XML processing, and seems more appropriate for use of XML in document processing than in data representation. As such, this document recommends avoiding entity declarations in protocol specifications.

On the other hand, there are five standard entity references built into XML: "&", "<", ">", "'", and """. XML also has the ability to write character data using numeric entity references (using the Unicode [33] value for the character). Entity references are normally expanded before the XML Information Set is computed. Restricting the use of these entity references would introduce an additional syntactic restriction (see Section 4.3) unnecessarily; these entity references should be allowed.

4.14 External References

When using XML in the context of a stateless protocol, be it the protocol itself (e.g., SOAP), or simply as content transferred by an existing protocol (e.g., XML/HTTP), care must be taken to not make the meaning of a message depend on information outside the message itself. XML provides external entities (see Section 4.13), which are an easy way to make the meaning of a message depend on something external. Using schema languages that can change the Infoset, like XML Schema, is another way.

4.15 URI Processing

The XML Base specification [36] defines an attribute "xml:base" in the XML namespace that is intended to affect the "base" to be used for relative URI processing described in RFC 2396 [17]. The facilities of xml:base for controlling URI processing may be useful to protocol designers, but if xml:base is allowed the interaction with any other protocol facilities for establishing URI context must be specified clearly. Note that use of relative URIs in namespace declarations has been deprecated by the W3C; some specific issues with relative URIs in namespace declarations and canonical XML can be found in section 1.3 of RFC 3076 [6].

Note also that, in many cases, the term "URI" and the syntactic use of URIs within XML allows non-ASCII characters within URIs. For example, the XML Schema "anyURI" datatype ([42] section 3.2.17) allows for direct encoding of characters outside of the US-ASCII range. Most current IETF protocols and specifications do not allow this syntax. Protocol specifications should be clear about the range of characters specified, e.g., by adding a restriction to the range of characters allowed in the anyURI schema datatype, or by specifying that characters outside the US-ASCII range should be escaped when passed to older protocols or APIs.

4.16 White Space

XML's prescribed white space handling behavior can be a source of confusion between protocol designers and implementers. In XML instances all white space is considered significant and is by default visible to processing applications. Consider this example from Section 4.10:

```
<address>
  <addrType><ipv4/></addrType>
  <value>10.1.2.3</value>
</address>
```

This fragment contains an <address> element and two child elements. It also contains white space for pretty-printing purposes:

- at least three line separators, which will be converted by the XML processor to newline (U+000A) characters (see section 2.11 of [8]), and
- one or more white space characters prefixing the <addrType> and <value> elements, which an XML processor will make visible to software reading the instance.

Implementers might safely assume that they can ignore the white space in the example above, but white space used for pretty-printing can be a source of confusion in other situations. Consider a minor change to the <value> element:

```
<value>
  10.1.2.3
</value>
```

where white space is found on both sides of the IP address. XML processors treat the white space surrounding "10.1.2.3" as an integral part of the <value> element. A failure to recognize this behavior can lead to confusion and errors in both design and implementation.

All white space is considered significant in XML instances. As a consequence, it is recommended that protocol designers provide specific guidelines to address white space handling within protocols that use XML.

4.17 Interaction with the IANA

When XML is used in an IETF protocol there are multiple factors that might require IANA action, including:

- XML media types. A piece of XML in a protocol element is sometimes intrinsically bound to the protocol context in which it appears, and in particular might be directly derived from and/or input to protocol state-machine implementations. In cases where the XML content has no relevant meaning outside it's original protocol context, there is no reason to register a MIME type. When it is possible that XML content can be interpreted outside of its original context (such as when that XML content is being stored in a file system or tunneled over another protocol), then a MIME type can be registered to specify the specific format for the data and to provide a hint as to how it might be processed.

  If MIME labeling is needed, then the advice of RFC 3023 [5] applies. In particular, if the XML represents a new language or document type, a new MIME media type should be registered for the reasons described in RFC 3023 sections 7 and A.1. In situations where XML is used to encode generic structured data (e.g., a document-oriented application that involves combining XML with a stylesheet), "application/xml" might be appropriate ("MAY be used"). The "text/xml" media type is not recommended ("SHOULD NOT be used") because of issues involving display behavior and default charsets.

- URI registration. There is an ongoing effort ([11], [12]) to create a URN namespace explicitly for defining URIs for namespace names and other URI-designated protocol elements for use within IETF standards track documents; it might also establish IETF policy for such use.

5. Internationalization Considerations

This section describes internationalization considerations for the use of XML to represent data in IETF protocols. In addition to the recommendations here, IETF policy on the use of character sets and languages described in RFC 2277 [3] also applies.

5.1 Character Sets and Encodings

IETF protocols frequently speak of the "character set" or "charset" of a string, which is used to denote both the character repertoire and the encoding used to represent sequences of characters as sequences of bytes.

XML performs all character processing in terms of the Universal Character Set (UCS, [31] and [33]). XML requires all XML processors to support both the UTF-8 [4] and UTF-16 [20] encodings of UCS, although other encodings (charsets) compatible with UCS may be allowed. Documents and external parsed entities encoded in UTF-16 are required to begin with a Byte Order Mark ([8] section 4.3.3).

IETF policy [3] requires that the UTF-8 charset be allowed for all text.

This document requires that IETF protocols using XML allow for the UTF-8 encoding of XML data. Since conforming XML processors are mandated to also accept UTF-16 encoding, also allowing for UTF-16 encoding (with the mandated Byte Order Mark) is recommended. Some XML applications are using a Byte Order Mark with UTF-8 encoding, but this use should not be encouraged and isn't appropriate for XML embedded in other protocols.

Restricting XML data to only be expressed in UTF-8 is an additional syntactic restriction (see Section 4.3) which, depending on circumstances, might add additional implementation complexity. When encodings other than UTF-8 or UTF-16 are used, the encoding must be specified using an "encoding" attribute in the XML declaration (see Section 4.4), even if there might be other protocol mechanisms for designating the encoding.

5.2 Language Declaration

Text encapsulated in XML can be represented in many different human languages, and it is often useful to explicitly identify the language used to present the text. XML defines a special attribute in the "xml" namespace, xml:lang, that can be used to specify the language used to represent data in an XML document. The xml:lang attribute (which has to be explicitly declared for use within a DTD or XML Schema) and the values it can assume are defined in section 2.12 of [8].

It is strongly recommended that protocols representing data in a human language mandate use of an xml:lang attribute if the XML instance might be interpreted in language-dependent contexts.

5.3 Other Internationalization Considerations

There are standard mechanisms in the typography of some human languages that can be difficult to represent using merely XML character string data types. For example, pronunciation clues can be provided using Ruby annotation [39], and embedding controls (such as those described in section 3.4 of [34]) or an XHTML [40] "dir"

attribute can be used to note the proper display direction for bidirectional text.

There are a number of tricky issues that can arise when using extended character sets with XML document formats. For example:

- There are different ways of representing characters consisting of combining characters, and
- There has been some debate about whether URIs should be represented using a restricted US-ASCII subset or arbitrary Unicode (e.g., "URI character sequence" vs "original character sequence" in RFC 2396 [17]).

Some of these issues are discussed, with recommendations, in the W3C's "Character Model for the World Wide Web" document [44].

It is strongly recommended that protocols representing data in a human language reuse existing mechanisms as needed to ensure proper display of human-legible text.

6. IANA Considerations

This memo, per se, has no impact on the IANA. Section 4.17 notes some factors that might require IANA action when protocols using XML are defined.

7. Security Considerations

Network protocols face many different kinds of threats, including unintended disclosure, modification, and replay. Passive attacks, such as packet sniffing, allow an attacker to capture and view information intended for someone else. Captured data can be modified and replayed to the original intended recipient, with the recipient having no way to know that the information has been compromised, detect modifications, be assured of the sender's identity, or to confirm which protocol instance is legitimate.

Several security service options for XML are available to help mitigate these risks. Though XML does not include any built-in security services, other protocols and protocol layers provide services that can be used to protect XML protocols. XML encryption [10] provides privacy services to prevent unintended disclosure. Canonical XML [6] and XML digital signatures [7] provide integrity services to detect modification and authentication services to confirm the identity of the data source. Other IETF security protocols (e.g., the Transport Layer Security (TLS) protocol [2]) are also available to protect data and service endpoints as appropriate.

Given the lack of security services in XML, it is imperative that protocol specifications mandate additional security services to counter common threats and attacks; the specific required services will depend on the protocol's threat model.

Experience has shown that code that parses network traffic is often a "soft target" for blackhats. Accordingly, implementers MUST take great care to ensure that their XML handling code is robust with respect to malformed XML, buffer overruns, misuse of entity declarations, and so on.

XML mechanisms that follow external references (Section 4.14) may also expose an implementation to various threats by causing the implementation to access external resources automatically. It is important to disallow arbitrary access to such external references within XML data from untrusted sources. Many XML grammars define constructs using URIs for external references; in such cases, the same precautions must be taken.

8. Acknowledgements

The authors would like to thank the following people who have provided significant contributions to the development of this document:

Mark Baker, Tim Berners-Lee, Tim Bray, James Clark, Josh Cohen, John Cowan, Alan Crouch, Martin Duerst, Jun Fujisawa, Christian Geuer-Pollmann, Yaron Goland, Graham Klyne, Dan Kohn, Rick Jeliffe, Chris Lilley, Murata Makoto, Michael Mealling, Jean-Jacques Moreau, Andrew Newton, Julian Reschke, Jonathan Rosenberg, Miles Sabin, Rich Salz, Peter Saint-Andre, Simon St Laurent, Margaret Wasserman, and Daniel Veillard.

9. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[3] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[4] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[5] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[6] Boyer, J., "Canonical XML Version 1.0", RFC 3076, March 2001.

[7] Eastlake, D., Reagle, J. and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

[8] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (2nd ed)", W3C REC-xml, October 2000, <http://www.w3.org/TR/REC-xml>.

[9] Bray, T., Hollander, D. and A. Layman, "Namespaces in XML", W3C REC-xml-names, January 1999, <http://www.w3.org/TR/REC-xml-names>.

[10] Imamura, T., Dillaway, B., Schaad, J. and E. Simon, "XML Encryption Syntax and Processing", W3C REC-xmlenc-core, October 2001, <http://www.w3.org/TR/xmlenc-core/>.

10. Informative References

[11] Masinter, L., Mealling, M., Klyne, G. and T. Hardie, "An IETF URN Sub-namespace for Registered Protocol Parameters", Work in Progress.

[12] Mealling, M., "The IETF XML Registry", Work in Progress.

[13] Case, J., Fedor, M., Schoffstall, M. and C. Davin, "Simple Network Management Protocol (SNMP)", STD 15, RFC 1157, May 1990.

[14] Srinivasan, R., "XDR: External Data Representation Standard", RFC 1832, August 1995.

[15] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[16] Crocker, D. (Ed.) and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[17] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[18] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[19] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[20] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

[21] Klensin, J. (Ed.), "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[22] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M. and D. Noveck, "NFS version 4 Protocol", RFC 3010, December 2000.

[23] Kennedy, H., "Binary Lexical Octet Ad-hoc Transport", RFC 3252, April 2002.

[24] Popp, N., Mealling, M. and M. Moseley, "Common Name Resolution Protocol (CNRP)", RFC 3367, August 2002.

[25] Backus, J., "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference", June 1959.

[26] American National Standards Institute, "Code Extension Techniques for Use with the 7-bit Coded Character Set of American National Standard Code (ASCII) for Information Interchange", ANSI X3.41, FIPS PUB 35, 1974.

[27] American National Standards Institute, "Information Retrieval: Application Service Definition and Protocol Specification", ANSI Z39.50, ISO Standard 23950, 1995.

[28] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)", ISO Standard 8824, December 1990.

[29] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)", ISO Standard 8825, December 1990.

[30] International Organization for Standardization, "Information processing - Text and office systems - Standard Generalized Markup Language (SGML)", ISO Standard 8879, 1988.

[31] International Organization for Standardization, "Information Technology - Universal Multiple-octet coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", ISO Standard 10646-1, May 1993.

[32] International Organization for Standardization, "DSDL Part 0 - Overview", December 2001, <http://www.jtc1.org/FTP/Public/SC34/DOCREG/0275.htm>.

[33] Unicode Consortium, "The Unicode Standard, as it may from time to time be revised or amended", March 2002, <http://www.unicode.org/unicode/standard/standard.html>.

[34] Duerst, M. and A. Freytag, "Unicode in XML and other Markup Languages", February 2002, <http://www.w3.org/TR/unicode-xml/>.

[35] Bray, T., Paoli, J. and C. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", W3C REC-xml-1998, February 1998, <http://www.w3.org/TR/1998/REC-xml-19980210/>.

[36] Marsh, J., "XML Base", W3C REC-xmlbase, June 2001, <http://www.w3.org/TR/xmlbase/>.

[37] Cowan, J. and R. Tobin, "XML Information Set", W3C REC-infoset, October 2001, <http://www.w3.org/TR/xml-infoset/>.

[38] Lassila, O. and R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification", W3C REC-rdf-syntax, February 1999, <http://www.w3.org/TR/REC-rdf-syntax>.

[39] Suignard, M., Ishikawa, M., Duerst, M. and T. Texin, "Ruby Annotation", W3C REC-RUBY, May 2001, <http://www.w3.org/TR/ruby/>.

[40] Pemberton, S., "XHTML 1.0: The Extensible HyperText Markup Language", W3C REC-XHTML, January 2000, <http://www.w3.org/TR/xhtml1/>.

[41] Thompson, H., Beech, D., Maloney, M. and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC-xmlschema-1, May 2001, <http://www.w3.org/TR/xmlschema-1/>.

[42] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes", W3C REC-xmlschema-2, May 2001, <http://www.w3.org/TR/xmlschema-2/>.

[43] Clark, J., "XSL Transformations (XSLT) Version 1.0", W3C REC-xslt, November 1999, <http://www.w3.org/TR/xslt>.

[44] Duerst, M., Yergeau, F., Ishida, R., Wolf, M., Freytag, A. and T. Texin, "Character Model for the World Wide Web 1.0", April 2002, <http://www.w3.org/TR/charmod/>.

[45] Gudgin, M., Hadley, M., Moreau, JJ. and H. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework", June 2002, <http://www.w3.org/TR/soap12-part1/>.

[46] Gudgin, M., Hadley, M., Moreau, JJ. and H. Nielsen, "SOAP Version 1.2 Part 2: Adjuncts", June 2002, <http://www.w3.org/TR/soap12-part2/>.

[47] W3C Communications Team, "XML in 10 points", November 2001, <http://www.w3.org/XML/1999/XML-in-10-points>.

[48] OASIS Technical Committee: RELAX NG, "RELAX NG Specification", December 2001, <http://www.oasis-open.org/committees/relax-ng/spec-20011203.html>.

[49] Jelliffe, R., "The Schematron", November 2001, <http://www.ascc.net/xml/schematron/>.

URIs

[50] <http://www.imc.org/ietf-xml-use/>

[51] <http://xml.org/>

[52] <http://xmlhack.com/>

[53] <http://oasis-open.org/>

11. Authors' Addresses

Scott Hollenbeck
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
US

Phone: +1 703 948 3257
EMail: shollenbeck@verisign.com

Marshall T. Rose
Dover Beach Consulting, Inc.
POB 255268
Sacramento, CA 95865-5268
US

Phone: +1 916 483 8878
EMail: mrose@dbc.mtview.ca.us

Larry Masinter
Adobe Systems Incorporated
Mail Stop W14
345 Park Ave.
San Jose, CA 95110
US

Phone: +1 408 536 3024
EMail: LMM@acm.org
URI: http://larry.masinter.net

12. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.