XML Media Types
draft-lilley-xml-mediatypes-00

Abstract

   This specification standardizes three media types -- application/xml,
   application/xml-external-parsed-entity, and application/xml-dtd --
   for use in exchanging network entities that are related to the
   Extensible Markup Language (XML) while defining text/xml and text/
   xml-external-parsed-entity as aliases for the respective application/
   types. This specification also standardizes a convention (using the
   suffix '+xml') for naming media types outside of these five types
   when those media types represent XML MIME entities. XML MIME
   entities are currently exchanged via the HyperText Transfer Protocol
   on the World Wide Web, are an integral part of the WebDAV protocol
   for remote web authoring, and are expected to have utility in many
   domains.

   Major differences from [RFC3023] are alignment of charset handling
   for text/xml and text/xml-external-parsed-entity with application/
   xml, the addition of XPointer and XML Base as fragment identifiers
   and base URIs, respectively, mention of the XPointer Registry, and
   updating of many references.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on April 02, 2013.

Table of Contents

1. Introduction

The World Wide Web Consortium has issued the Extensible Markup
Language (XML) 1.0 specification. [XML]. To enable the exchange of
XML network entities, this specification standardizes three media
types -- application/xml, application/xml-external-parsed-entity, and
application/xml-dtd and two aliases -- text/xml and text/xml-
external-parsed-entity, as well as a naming convention for
identifying XML-based MIME media types (using +xml).

XML entities are currently exchanged on the World Wide Web, and XML
is also used for property values and parameter marshalling by the
WebDAV [RFC4918] protocol for remote web authoring. Thus, there is
a need for a media type to properly label the exchange of XML network
entities.

Although XML is a subset of the Standard Generalized Markup Language
(SGML) ISO 8879 [SGML], which has been assigned the media types text
/sgml and application/sgml, there are several reasons why use of text
/sgml or application/sgml to label XML is inappropriate. First,
there exist many applications that can process XML, but that cannot
process SGML, due to SGML's larger feature set. Second, SGML
applications cannot always process XML entities, because XML uses
features of recent technical corrigenda to SGML. Third, the
definition of text/sgml and application/sgml in [RFC1874] includes
parameters for SGML bit combination transformation format (SGML-
bctf), and SGML boot attribute (SGML-boot). Since XML does not use
these parameters, it would be ambiguous if such parameters were given
for an XML MIME entity. For these reasons, the best approach for
labeling XML network entities has been to provide new media types for
XML.

Since XML is an integral part of the WebDAV Distributed Authoring
Protocol, and since World Wide Web Consortium Recommendations are
assigned standards tree media types, and since similar media types
(HTML, SGML) have been assigned standards tree media types, the XML
media types were also placed in the standards tree [RFC3023].

Similarly, XML has been used as a foundation for other media types,
including types in every branch of the IETF media types tree. To
facilitate the processing of such types, media types based on XML,
but that are not identified using application/xml (or text/xml),
SHOULD be named using a suffix of '+xml' as described in Section 8.
This will allow generic XML-based tools -- browsers, editors, search
engines, and other processors -- to work with all XML-based media
types.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
specification are to be interpreted as described in [RFC2119].

As defined in [RFC2781] (informative), the three charsets "utf-16",
"utf-16le", and "utf-16be" are used to label UTF-16 text. In this
specification, "the UTF-16 family" refers to those three charsets.
By contrast, the phrases "utf-16" or UTF-16 in this specification
refer specifically to the single charset "utf-16".

As sometimes happens between two communities, both MIME and XML have defined the term entity, with different meanings. Section 2.4 of [RFC2045] says:

"The term 'entity' refers specifically to the MIME-defined header
fields and contents of either a message or one of the parts in the
body of a multipart entity."

Section 4 of [XML] says:

"An XML document may consist of one or many storage units. These
are called entities; they all have content and are all (except for
the document entity and the external DTD subset) identified by
entity name".

In this specification, "XML MIME entity" is defined as the latter (an
XML entity) encapsulated in the former (a MIME entity).

3. XML Media Types

This specification standardizes three media types related to XML MIME
entities: application/xml (with text/xml as an alias), application/
xml-external-parsed-entity (with text/xml-external-parsed-entity as
an alias), and application/xml-dtd. Registration information for
these media types is described in the sections below.

Within the XML specification, XML MIME entities can be classified
into four types. In the XML terminology, they are called "document
entities", "external DTD subsets", "external parsed entities", and
"external parameter entities". The media types application/xml or
text/xml MAY be used for "document entities", while application/xml-
external-parsed-entity or text/xml-external-parsed-entity SHOULD be
used for "external parsed entities". Note that [RFC3023] (which this
specification obsoletes) recommended the use of text/xml and text/
xml-external-parsed-entity for document entities and external parsed
entities, respectively, but described charset handling which differed
from common implementation practice. These media types are still
commonly used, and this specification aligns the charset handling
with industry practice. The media type application/xml-dtd SHOULD be
used for "external DTD subsets" or "external parameter entities".
application/xml and text/xml MUST NOT be used for "external parameter
entities" or "external DTD subsets", and MUST NOT be used for
"external parsed entities" unless they are also well-formed "document
entities" and are referenced as such. Note that [RFC2376] (which is
obsolete) allowed such usage, although in practice it is likely to
have been rare.

Neither external DTD subsets nor external parameter entities parse as
XML documents, and while some XML document entities may be used as
external parsed entities and vice versa, there are many cases where
the two are not interchangeable. XML also has unparsed entities,
internal parsed entities, and internal parameter entities, but they

are not XML MIME entities.

Application/xml and application/xml-external-parsed-entity are
recommended. Compared to [RFC2376] or [RFC3023], this specification
alters the charset handling of text/xml and text/xml-external-parsed-
entity, treating them no differently from the respective application/
types. The reasons are as follows:

Conflicting specifications regarding the character encoding have
caused confusion. On the one hand, [RFC2046] specifies "The
default character set, which must be assumed in the absence of a
charset parameter, is US-ASCII.", [RFC2616] Section 3.7.1, defines
that "media subtypes of the 'text' type are defined to have a
default charset value of 'ISO-8859-1'", and [RFC2376] as well as
[RFC3023] specify the default charset is US-ASCII.

On the other hand, implementors and users of XML parsers,
following Appendix F of [XML], assume that the default is provided
by the XML encoding declaration or BOM. Note that this conflict
did not exist for application/xml or application/xml-external-
parsed-entity (see "Optional parameters" of application/xml
registration in Section 3.1).

The current situation, reflected in this specification, has been
simplified by [RFC6657] updating [RFC2046] to remove the US-ASCII
default. Furthermore, in accordance with [RFC6657]'s other
recommendations, [HTTPbis] changes [RFC2616] by removing the
ISO-8859-1 default and not defining any default at all.

The top-level media type "text" has some restrictions on MIME
entities and they are described in [RFC2045] and [RFC2046]. In
particular, for transports other than HTTP [RFC2616] or HTTPS
(which uses a MIME-like mechanism). the UTF-16 family, UCS-4, and
UTF-32 are not allowed However, section 4.3.3 of [XML] says:

"Each external parsed entity in an XML document may use a
different encoding for its characters. All XML processors MUST
be able to read entities in both the UTF-8 and UTF-16
encodings."

Thus, although all XML processors can read entities in at least
UTF-16, if an XML document or external parsed entity is encoded in
such character encoding schemes, it could not be labeled as text/
xml or text/xml-external-parsed-entity (except for HTTP).

It is not possible to deprecate text/xml because it is widely used
in practice, and implementations are largely interoperable,
following the rules of Appendix F of [XML] and ignoring the
requirements of [RFC3023].

XML provides a general framework for defining sequences of structured
data. In some cases, it may be desirable to define new media types
that use XML but define a specific application of XML, perhaps due to
domain-specific display, editing, security considerations or runtime
information. Furthermore, such media types may allow UTF-8 or UTF-16
only and prohibit other charsets. This specification does not
prohibit such media types and in fact expects them to proliferate.
However, developers of such media types are STRONGLY RECOMMENDED to
use this specification as a basis for their registration. In
particular, the charset parameter, if used, MUST agree with the
encoding of the XML entity, as described in Section 8.1, in order to
enhance interoperability.

An XML document labeled as application/xml or text/xml, or with a
+xml media type, might contain namespace declarations, stylesheet-
linking processing instructions (PIs), schema information, or other
declarations that might be used to suggest how the document is to be
processed. For example, a document might have the XHTML namespace
and a reference to a CSS stylesheet. Such a document might be
handled by applications that would use this information to dispatch
the document for appropriate processing.

3.1. Application/xml Registration

MIME media type name: application

MIME subtype name: xml

Mandatory parameters: none

Optional parameters: charset

The charset parameter MUST only be used, when the charset is
reliably known and agrees with the encoding declaration. This
information can be used by non-XML processors to determine
authoritatively the charset of the XML MIME entity. The charset
parameter can also be used to provide protocol-specific
operations, such as charset-based content negotiation in HTTP.

"utf-8" [RFC3629] and "utf-16" [RFC2781] are the recommended
values, representing the UTF-8 and UTF-16 charsets, respectively.
These charsets are preferred since they are supported by all
conforming processors of [XML].

If an application/xml entity is received where the charset
parameter is omitted, no information is being provided about the
charset by the MIME Content-Type header. Conforming XML
processors MUST follow the requirements in section 4.3.3 of [XML]
that directly address this contingency. However, MIME processors
that are not XML processors SHOULD NOT assume a default charset if
the charset parameter is omitted from an application/xml entity.

There are several reasons that the charset parameter is optionally
allowed. First, recent web servers have been improved so that
users can specify the charset parameter. Second, [RFC2130]
(informative) specifies that the recommended specification scheme
is the "charset" parameter.

On the other hand, it has been argued that the charset parameter
should be omitted and the mechanism described in Appendix F of
[XML] (which is non-normative) should be solely relied on. This
approach would allow users to avoid configuration of the charset
parameter; an XML document stored in a file is likely to contain a
correct encoding declaration or BOM (if necessary), since the
operating system does not typically provide charset information
for files. If users would like to rely on the encoding
declaration or BOM and to hide charset information from protocols,
they SHOULD determine not to use the parameter.

Since a receiving application can, with very high reliability,
determine the encoding of an XML document by reading it, the XML
encoding declaration SHOULD be provided.

Encoding considerations: This media type MAY be encoded as
   appropriate for the charset and the capabilities of the underlying
   MIME transport. For 7-bit transports, data in either UTF-8 or
   UTF-16 MUST be encoded in quoted-printable or base64. For 8-bit
   clean transport (e.g., 8BITMIME [RFC1652] ESMTP or NNTP
   [RFC3977]), UTF-8 is not encoded, but the UTF-16 family MUST be
   encoded in base64. For binary clean transports (e.g., HTTP
   [RFC2616]), no content-transfer-encoding is necessary.

Security considerations: See Section 11.

Interoperability considerations: XML has proven to be interoperable
   across WebDAV clients and servers, and for import and export from
   multiple XML authoring tools. For maximum interoperability,
   validating processors are recommended. Although non-validating
   processors may be more efficient, they are not required to handle
   all features of XML. For further information, see sub-section 2.9
   "Standalone Document Declaration" and section 5 "Conformance" of
   [XML] .

Published specification: Extensible Markup Language (XML) 1.0 (Fifth
      Edition) [XML].

Applications which use this media type: XML is device-, platform-,
and vendor-neutral and is supported by a wide range of Web user
agents, WebDAV [RFC4918] clients and servers, as well as XML
authoring tools.

Additional information:

Magic number(s): None.

Although no byte sequences can be counted on to always be
present, XML MIME entities in ASCII-compatible charsets
(including UTF-8) often begin with hexadecimal 3C 3F 78 6D 6C
("<?xml"), and those in UTF-16 often begin with hexadecimal FE
FF 00 3C 00 3F 00 78 00 6D 00 6C or FF FE 3C 00 3F 00 78 00 6D
00 6C 00 (the Byte Order Mark (BOM) followed by "<?xml"). For
more information, see Appendix F of [XML].

File extension(s): .xml

Macintosh File Type Code(s): "TEXT"

Person and email address for further information:

MURATA Makoto (FAMILY Given) <eb2m-mrt@asahi-net.or.jp>

Alexey Melnikov <alexey.melnikov@isode.com>

Chris Lilley <chris@w3.org>

Henry S. Thompson <ht@inf.ed.ac.uk>

Intended usage: COMMON

Author/Change controller: The XML specification is a work product of
the World Wide Web Consortium's XML Working Group, and was edited
by:

Tim Bray <tbray@textuality.com>

Jean Paoli <jeanpa@microsoft.com>

C. M. Sperberg-McQueen <cmsmcq@uic.edu>

Eve Maler <eve.maler@east.sun.com>

Francois Yergeau <mailto:francois@yergeau.com>

3.2. Text/xml Registration

text/xml is an alias for application/xml, as defined in Section 3.1

above.

[3.3](). Application/xml-external-parsed-entity Registration

MIME media type name: application

MIME subtype name: xml-external-parsed-entity

Mandatory parameters: none

Optional parameters: charset

   The charset parameter of application/xml-external-parsed-entity is
   handled the same as that of application/xml as described in
   Section 3.1.

Encoding considerations: Same as application/xml as described in
   Section 3.1.

Security considerations: See Section 11.

Interoperability considerations: XML external parsed entities are as
   interoperable as XML documents, though they have a less tightly
   constrained structure and therefore need to be referenced by XML
   documents for proper handling by XML processors. Similarly, XML
   documents cannot be reliably used as external parsed entities
   because external parsed entities are prohibited from having
   standalone document declarations or DTDs. Identifying XML
   external parsed entities with their own content type should
   enhance interoperability of both XML documents and XML external
   parsed entities.

Published specification: Same as application/xml as described in
   Section 3.1.

Applications which use this media type: Same as application/xml as
   described in Section 3.1.

Additional information:

   Magic number(s): Same as application/xml as described in Section
      3.1.

   File extension(s): .xml or .ent

   Macintosh File Type Code(s): "TEXT"

Person and email address for further information: Same as application
   /xml as described in Section 3.1.

Intended usage: COMMON

Author/Change controller: Same as application/xml as described in

[Section 3.1](#).

3.4. Text/xml-external-parsed-entity Registration

text/xml-external-parsed-entity is an alias for application/xml-external-parsed-entity, as defined in Section 3.3 above.

3.5. Application/xml-dtd Registration

MIME media type name: application

MIME subtype name: xml-dtd

Mandatory parameters: none

Optional parameters: charset

   The charset parameter of application/xml-dtd is handled the same
   as that of application/xml as described in Section 3.1.

Encoding considerations: Same as Section 3.1.

Security considerations: See Section 11.

Interoperability considerations: XML DTDs have proven to be
   interoperable by DTD authoring tools and XML browsers, among
   others.

Published specification: Same as application/xml as described in
   Section 3.1.

Applications which use this media type: DTD authoring tools handle
   external DTD subsets as well as external parameter entities. XML
   browsers may also access external DTD subsets and external
   parameter entities.

Additional information:

   Magic number(s): Same as application/xml as described in Section
      3.1.

   File extension(s): .dtd or .mod

   Macintosh File Type Code(s): "TEXT"

Person and email address for further information: Same as application
   /xml as described in Section 3.1.

Intended usage: COMMON

Author/Change controller: Same as application/xml as described in
   Section 3.1.

3.6. Summary

o If the charset parameter is omitted, conforming XML processors
   MUST follow the requirements in section 4.3.3 of [XML] or [XML1.1]
   as appropriate.

o If provided, the charset parameter MUST agree with the xml
   encoding declaration.

4. The Byte Order Mark (BOM) and Conversions to/from the UTF-16 Charset

Section 4.3.3 of [XML] specifies that XML MIME entities in the
charset "utf-16" MUST begin with a byte order mark (BOM), which is a
hexadecimal octet sequence 0xFE 0xFF (or 0xFF 0xFE, depending on
endian). The XML Recommendation further states that the BOM is an
encoding signature, and is not part of either the markup or the
character data of the XML document.

Due to the presence of the BOM, applications that convert XML from
"utf-16" to a non-Unicode encoding MUST strip the BOM before
conversion. Similarly, when converting from another encoding into
"utf-16", the BOM MUST be added after conversion is complete.

In addition to the charset "utf-16", [RFC2781] introduces "utf-16le"
(little endian) and "utf-16be" (big endian) as well. The BOM is
prohibited for these charsets. When an XML MIME entity is encoded in
"utf-16le" or "utf-16be", it MUST NOT begin with the BOM but SHOULD
contain an encoding declaration. Conversion from "utf-16" to "utf-
16be" or "utf-16le" and conversion in the other direction MUST strip
or add the BOM, respectively.

5. Fragment Identifiers

Uniform Resource Identifiers (URIs) may contain fragment identifiers
(see Section 3.5 of [RFC3986]). Likewise, Internationalized Resource
Identifiers (IRIs) [RFC3987] may contain fragment identifiers.

The syntax and semantics of fragment identifiers for the XML media
types defined in this specification are based on the
[XPointerFramework] W3C Recommendation. It allows simple names, and
more complex constructions based on named schemes. When the syntax
of a fragment identifier part of any URI or IRI with a retrieved
media type governed by this specification conforms to the syntax
specified in [XPointerFramework], conformant applications MUST
attempt to interpret such fragment identifiers as designating that
part of the retrieved representation specified by
[XPointerFramework] and whatever other specifications define any
XPointer schemes used. Conformant applications MUST support the
'element' scheme as defined in [XPointerElement], but need not
support other schemes.

If an XPointer error is reported in the attempt to process the part,
this specification does not define an interpretation for the part.

A registry of XPointer schemes [XPtrReg] is maintained at the W3C.
Unregistered schemes SHOULD NOT be used.

See Section 8.1 for additional rquirements which apply when an XML-
based MIME media type follows the naming convention '+xml'.

If [XPointerFramework] and [XPointerElement] are inappropriate for
some XML-based media type, it SHOULD NOT follow the naming convention
'+xml'.

When a URI has a fragment identifier, it is encoded by a limited
subset of the repertoire of US-ASCII [ASCII] characters, as defined
in [RFC3986]. When an IRI contains a fragment identifier, it is
encoded by a much wider repertoire of characters. The conversion
between IRI fragment identifiers and URI fragment identifiers is
presented in Section 7 of [RFC3987].

6. The Base URI

Section 5.1 of [RFC3986] specifies that the semantics of a relative
URI reference embedded in a MIME entity is dependent on the base URI.
The base URI is either (1) the base URI embedded in context, (2) the
base URI from the encapsulating entity, (3) the base URI from the
Retrieval URI, or (4) the default base URI, where (1) has the highest
precedence. [RFC3986] further specifies that the mechanism for
embedding the base URI is dependent on the media type.

The media type dependent mechanism for embedding the base URI in a
MIME entity of type application/xml, text/xml, application/xml-
external-parsed-entity or text/xml-external-parsed-entity is to use
the xml:base attribute described in detail in [XBase].

Note that the base URI may be embedded in a different MIME entity,
since the default value for the xml:base attribute may be specified
in an external DTD subset or external parameter entity.

7. XML Versions

application/xml, application/xml-external-parsed-entity, and
application/xml-dtd, text/xml and text/xml-external-parsed-entity are
to be used with [XML]   In all examples herein where version="1.0" is
shown, it is understood that version="1.1" may also be used,
providing the content does indeed conform to [XML1.1].

The normative requirement of this specification upon XML is to follow
the requirements of [XML], section 4.3.3. Except for minor
clarifications, that section is substantially identical from the
first edition to the current (5th) edition of XML 1.0, and for XML
1.1. Therefore, this specification may be used with any version or

edition of XML 1.0 or 1.1.

Specifications and recommendations based on or referring to this RFC
SHOULD indicate any limitations on the particular versions of XML to
be used. For example, a particular specification might indicate:
"content MUST be represented using media-type application/xml, and
the document must either (a) carry an xml declaration specifying
version="1.0" or (b) omit the XML declaration, in which case per the
XML recommendation the version defaults to 1.0"

8. **A Naming Convention for XML-Based Media Types**

This specification recommends the use of a naming convention (a
suffix of '+xml') for identifying XML-based MIME media types,
whatever their particular content may represent. This allows the use
of generic XML processors and technologies on a wide variety of
different XML document types at a minimum cost, using existing
frameworks for media type registration.

Although the use of a suffix was not considered as part of the
original MIME architecture, this choice is considered to provide the
most functionality with the least potential for interoperability
problems or lack of future extensibility. The alternatives to the
'+xml' suffix and the reason for its selection are described in
Appendix A.

As XML development continues, new XML document types are appearing
rapidly. Many of these XML document types would benefit from the
identification possibilities of a more specific MIME media type than
text/xml or application/xml can provide, and it is likely that many
new media types for XML-based document types will be registered in
the near and ongoing future.

While the benefits of specific MIME types for particular types of XML
documents are significant, all XML documents share common structures
and syntax that make possible common processing.

Some areas where 'generic' processing is useful include:

o Browsing - An XML browser can display any XML document with a
   provided [CSS] or [XSLT] style sheet, whatever the vocabulary of
   that document.

o Editing - Any XML editor can read, modify, and save any XML
   document.

o Fragment identification - XPointers (see Section 5) can work with
   any XML document, whatever vocabulary it uses.

o Hypertext linking - [XLink] hypertext linking is designed to
   connect any XML documents, regardless of vocabulary.

o Searching - XML-oriented search engines, web crawlers, agents, and
   query tools should be able to read XML documents and extract the
   names and content of elements and attributes even if the tools are
   ignorant of the particular vocabulary used for elements and
   attributes.

o Storage - XML-oriented storage systems, which keep XML documents
   internally in a parsed form, should similarly be able to process,
   store, and recreate any XML document.

o Well-formedness and validity checking - An XML processor can
   confirm that any XML document is well-formed and that it is valid
   (i.e., conforms to its declared DTD or Schema).

When a new media type is introduced for an XML-based format, the name
of the media type SHOULD end with '+xml'. This convention will allow
applications that can process XML generically to detect that the MIME
entity is supposed to be an XML document, verify this assumption by
invoking some XML processor, and then process the XML document
accordingly. Applications may match for types that represent XML
MIME entities by comparing the subtype to the pattern '*/*+xml'. (Of
course, 4 of the 5 media types defined in this specification -- text/
xml, application/xml, text/xml-external-parsed-entity, and
application/xml-external-parsed-entity -- also represent XML MIME
entities while not conforming to the '*/*+xml' pattern.)

    NOTE: Section 14.1 of HTTP [RFC2616] does not support Accept
    headers of the form "Accept: */*+xml" and so this header MUST NOT
    be used in this way. Instead, content negotiation [RFC2703]
    could potentially be used if an XML-based MIME type were needed.

Media types following the naming convention '+xml' SHOULD introduce
the charset parameter for consistency, since XML-generic processing
applies the same program for any such media type. However, there are
some cases that the charset parameter need not be introduced. For
example:

    When an XML-based media type is restricted to UTF-8, it is not
    necessary to introduce the charset parameter. "UTF-8 only" is a
    generic principle and UTF-8 is the default of XML.

    When an XML-based media type is restricted to UTF-8 and UTF-16, it
    might not be unreasonable to omit the charset parameter. Neither
    UTF-8 nor UTF-16 require encoding declarations of XML.

    Note: Some argue that XML-based media types should not introduce
    the charset parameter, although others disagree.

XML generic processing is not always appropriate for XML-based media types. For example, authors of some such media types may wish that the types remain entirely opaque except to applications that are specifically designed to deal with that media type. By NOT following the naming convention '+xml', such media types can avoid XML-generic processing. Since generic processing will be useful in many cases, however -- including in some situations that are difficult to predict ahead of time -- those registering media types SHOULD use the '+xml' convention unless they have a particularly compelling reason not to.

*HST: This paragraph needs updating once some pending RFCs are out there *The registration process for these media types is described in [RFC4288] and [RFC4289] . The registrar for the IETF tree will encourage new XML-based media type registrations in the IETF tree to follow this guideline. Registrars for other trees SHOULD follow this convention in order to ensure maximum interoperability of their XML-based documents. Similarly, media subtypes that do not represent XML MIME entities MUST NOT be allowed to register with a '+xml' suffix.

8.1. Referencing

Registrations for new XML-based media types under top-level types SHOULD, in specifying the charset parameter and encoding considerations, define them as: "Same as [charset parameter / encoding considerations] of application/xml as specified in RFC XXXX."

The use of the charset parameter is STRONGLY RECOMMENDED, since this information can be used by XML processors to determine authoritatively the charset of the XML MIME entity.   If there are some reasons not to follow this advice, they SHOULD be included as part of the registration. As shown above, two such reasons are "UTF-8 only" or "UTF-8 or UTF-16 only".

These registrations SHOULD specify that the XML-based media type being registered has all of the security considerations described in RFC XXXX plus any additional considerations specific to that media type.

These registrations SHOULD also make reference to RFC XXXX in specifying magic numbers, base URIs, and use of the BOM.

When these registrations use the '+xml' convention, they MUST also make reference to RFC XXXX in specifying fragment identifier syntax and semantics, and they MAY restrict the syntax to a specified subset of schemes, except that they MUST NOT disallow barenames or 'element' scheme pointers. They MAY further require support for other registered schemes. They also MAY add additional syntax (which MUST

NOT overlap with [XPointerFramework] syntax) together with associated
semantics, and MAY add additional semantics for barename XPointers
which, as provided for in Section 5, will only apply when this
specification does not define an interpretation.

These registrations MAY reference the application/xml registration in
RFC XXXX in specifying interoperability considerations, if these
considerations are not overridden by issues specific to that media
type.

9. Examples

The examples below give the value of the MIME Content-type header and
the XML declaration (which includes the encoding declaration) inside
the XML MIME entity. For UTF-16 examples, the Byte Order Mark
character is denoted as "{BOM}", and the XML declaration is assumed
to come at the beginning of the XML MIME entity, immediately
following the BOM. Note that other MIME headers may be present, and
the XML MIME entity may contain other data in addition to the XML
declaration; the examples focus on the Content-type header and the
encoding declaration for clarity.

9.1. application/xml or text/xml with Omitted Charset and 8-bit MIME entity

Content-type: application/xml or text/xml

<?xml version="1.0" encoding="iso-8859-1"?>

Since the charset parameter is not provided in the Content-Type
header, XML processors MUST treat the "iso-8859-1" encoding as
authoritative. XML-unaware MIME processors SHOULD make no
assumptions about the charset of the XML MIME entity.

9.2. application/xml or text/xml with Omitted Charset and 16-bit MIME entity

Content-type: application/xml or text/xml

{BOM}<?xml version="1.0" encoding="utf-16"?>

or

{BOM}<?xml version="1.0"?>

This example shows a 16-bit MIME entity with no charset parameter.
Since the charset parameter is not provided in the Content-Type
header, in this case XML processors MUST treat the "utf-16" encoding
and/or the BOM as authoritative. XML-unaware MIME processors SHOULD
make no assumptions about the charset of the XML MIME entity.

Omitting the charset parameter is NOT RECOMMENDED for application/xml
when used with transports other than HTTP or HTTPS---text/xml SHOULD
NOT be used for 16-bit MIME with transports other than HTTP or HTTPS

(see. Section 9.5).

9.3. **application/xml or text/xml with UTF-8 Charset**

   Content-type: application/xml or text/xml; charset="utf-8"

   <?xml version="1.0" encoding="utf-8"?>

   This is the recommended encoding for use with all the media types
   defined in this specification. Since the charset parameter is
   provided, both MIME and XML processors MUST treat the enclosed entity
   as UTF-8 encoded.

   If sent using a 7-bit transport (e.g. SMTP [RFC5321]), the XML MIME
   entity MUST use a content-transfer-encoding of either quoted-
   printable or base64. For an 8-bit clean transport (e.g., 8BITMIME
   ESMTP or NNTP), or a binary clean transport (e.g., HTTP), no content-
   transfer-encoding is necessary.

9.4. **application/xml with UTF-16 Charset**

   Content-type: application/xml; charset="utf-16"

   {BOM}<?xml version="1.0" encoding="utf-16"?>

   or

   {BOM}<?xml version="1.0"?>

   If sent using a 7-bit transport (e.g., SMTP) or an 8-bit clean
   transport (e.g., 8BITMIME ESMTP or NNTP), the XML MIME entity MUST be
   encoded in quoted-printable or base64. For a binary clean transport
   (e.g., HTTP), no content-transfer-encoding is necessary.

9.5. **text/xml with UTF-16 Charset**

   Content-type: text/xml; charset="utf-16"

   {BOM}<?xml version='1.0' encoding='utf-16'?>

   or

   {BOM}<?xml version='1.0'?>

   This is possible only when the XML MIME entity is transmitted via
   HTTP or HTTPS, which use a MIME-like mechanism and are binary-clean
   protocols, hence do not perform CR and LF transformations and allow
   NUL octets. As described in [RFC2781], the UTF-16 family MUST NOT be
   used with media types under the top-level type "text" except over
   HTTP or HTTPS (see section 19.4.1 of [RFC2616] for details).

   Since HTTP is binary clean, no content-transfer-encoding is

necessary.

9.6. **application/xml with UTF-16BE Charset**

    Content-type: application/xml; charset="utf-16be"

    <?xml version='1.0' encoding='utf-16be'?>

    Observe that the BOM does not exist. Since the charset parameter is
    provided, MIME and XML processors MUST treat the enclosed entity as
    UTF-16BE encoded.

9.7. text/xml with UTF-16BE Charset

    Content-type: text/xml; charset="utf-16be"

    <?xml version='1.0' encoding='utf-16be'?>

    Observe that the BOM does not exist. As for UTF-16, this is possible
    only when the XML MIME entity is transmitted via HTTP.

9.8. application/xml or text/xml with ISO-2022-KR Charset

    Content-type: application/xml; charset="iso-2022-kr"

    <?xml version="1.0" encoding="iso-2022-kr"?>

    This example shows the use of a Korean charset (e.g., Hangul) encoded
    following the specification in [RFC1557]. Since the charset
    parameter is provided, MIME processors MUST treat the enclosed entity
    as encoded per RFC 1557. Since the XML MIME entity has an internal
    encoding declaration (this example does show such a declaration,
    which agrees with the charset parameter) XML processors MUST also
    treat the enclosed entity as encoded per RFC 1557. Thus,
    interoperability is assured.

    Since ISO-2022-KR has been defined to use only 7 bits of data, no
    content-transfer-encoding is necessary with any transport.

9.9. application/xml or text/xml with Omitted Charset, no Internal
      Encoding Declaration and UTF-8 Entity

    Content-type: application/xml or text/xml

    <?xml version='1.0'?>

    In this example, the charset parameter has been omitted, the is no
    internal encoding declaration, and there is no BOM. Since there is
    no BOM, the XML processor follows the requirements in section 4.3.3,
    and optionally applies the mechanism described in Appendix F (which
    is non-normative) of [XML] to determine the charset encoding of

UTF-8. Although the XML MIME entity does not contain an encoding declaration, the encoding actually -is- UTF-8, so this is still a conforming XML MIME entity.

An XML-unaware MIME processor SHOULD make no assumptions about the charset of the XML MIME entity.

9.10. **application/xml or text/xml with Omitted Charset and Internal Encoding Declaration**

Content-type: application/xml or text/xml

<?xml version='1.0' encoding="iso-10646-ucs-4"?>

In this example, the charset parameter has been omitted, and there is no BOM. However, the XML MIME entity does have an encoding declaration inside the XML MIME entity that specifies the entity's charset. Following the requirements in section 4.3.3, and optionally applying the mechanism described in Appendix F (non-normative) of [XML], the XML processor determines the charset encoding of the XML MIME entity (in this example, UCS-4).

An XML-unaware MIME processor SHOULD make no assumptions about the charset of the XML MIME entity.

9.11. **application/xml-external-parsed-entity or text/xml-external-parsed-entity with UTF-8 Charset**

Content-type: text/xml-external-parsed-entity or application/xml-external-parsed-entity; charset="utf-8"

<?xml encoding="utf-8"?>

Since the charset parameter is provided, MIME and XML processors MUST treat the enclosed entity as UTF-8 encoded.

If sent using a 7-bit transport (e.g. SMTP), the XML MIME entity MUST use a content-transfer-encoding of either quoted-printable or base64. For an 8-bit clean transport (e.g., 8BITMIME ESMTP or NNTP), or a binary clean transport (e.g., HTTP) no content-transfer-encoding is necessary.

9.12. **application/xml-external-parsed-entity with UTF-16 Charset**

Content-type: application/xml-external-parsed-entity; charset="utf-16"

{BOM}<?xml encoding="utf-16"?>

or

{BOM}<?xml?>

Since the charset parameter is provided, MIME and XML processors MUST treat the enclosed entity as UTF-16 encoded.

If sent using a 7-bit transport (e.g., SMTP) or an 8-bit clean transport (e.g., 8BITMIME ESMTP or NNTP), the XML MIME entity MUST be encoded in quoted-printable or base64. For a binary clean transport (e.g., HTTP), no content-transfer-encoding is necessary.

9.13. application/xml-external-parsed-entity with UTF-16BE Charset

Content-type: application/xml-external-parsed-entity; charset="utf-16be"

<?xml encoding="utf-16be"?>

Since the charset parameter is provided, MIME and XML processors MUST treat the enclosed entity as UTF-16BE encoded.

9.14. application/xml-dtd

Content-type: application/xml-dtd; charset="utf-8"

<?xml encoding="utf-8"?>

Charset "utf-8" is a recommended charset value for use with application/xml-dtd. Since the charset parameter is provided, MIME and XML processors MUST treat the enclosed entity as UTF-8 encoded.

9.15. application/mathml+xml

Content-type: application/mathml+xml

<?xml version="1.0" ?>

MathML documents are XML documents whose content describes mathematical information, as defined by [MathML]. As a format based on XML, MathML documents SHOULD follow the '+xml' suffix convention and use 'mathml+xml' in their MIME content-type identifier.This media type has been registered at IANA and is fully defined in [MathML].

9.16. **application/xslt+xml**

   Content-type: application/xslt+xml

   <?xml version="1.0" ?>

   Extensible Stylesheet Language (XSLT) documents are XML documents
   whose content describes stylesheets for other XML documents, as
   defined by [XSLT]. As a format based on XML, XSLT documents SHOULD
   follow the '+xml' suffix convention and use 'xslt+xml' in their MIME
   content-type identifier.This media type has been registered at IANA
   and is fully defined in [XSLT].

9.17. **application/rdf+xml**

   Content-type: application/rdf+xml

   <?xml version="1.0" ?>

   Resources identified using the application/rdf+xml media type are XML
   documents whose content describe RDF metadata. This media type has
   been registered at IANA and is fully defined in [RFC3870].

9.18. **image/svg+xml**

   Content-type: image/svg+xml

   <?xml version="1.0" ?>

   Scalable Vector Graphics (SVG) documents are XML documents whose
   content describes graphical information, as defined by [SVG]. As a
   format based on XML, SVG documents SHOULD follow the '+xml' suffix
   convention and use 'svg+xml' in their MIME content-type
   identifier.The image/svg+xml media type has been registered at IANA
   and is fully defined in [SVG]. .

9.19. **model/x3d+xml**

   Content-type: model/x3d+xml

   <?xml version="1.0" ?>

   X3D is derived from VRML and is used for 3D models. Besides the XML
   representation, it may also be serialised in classic VRML syntax and
   using a fast infoset. Separate, but clearly related media types are
   used for these serialisations (model/x3d+vrml and model/
   x3d+fastinfoset respectively).

9.20. **INCONSISTENT EXAMPLE: text/xml with UTF-8 Charset**

      Content-type: text/xml; charset="utf-8"

      <?xml version="1.0" encoding="iso-8859-1"?>

      Since the charset parameter is provided in the Content-Type header
      and differs from the XML encoding declaration , MIME and XML
      processors will not interoperate. MIME processors will treat the
      enclosed entity as UTF-8 encoded. That is, the "iso-8859-1" encoding
      will be be ignored. XML processors on the other hand will ignore the
      charset parameter and treat the XML entity as encoded in iso-8859-1.

      Processors generating XML MIME entities MUST NOT label conflicting
      charset information between the MIME Content-Type and the XML
      declaration. In particular, the addition of an explicit, site-wide
      charset without inspecting the XML entity has frequently lead to
      interoperability problems.

9.21. application/soap+xml

      Content-type: application/soap+xml

      <?xml version="1.0" ?>

      Resources identified using the application/soap+xml media type are
      SOAP 1.2 message envelopes that have been serialized with XML 1.0.
      This media type has been registered at IANA and is fully defined in
      [RFC3902].

10. IANA Considerations

      As described in Section 8, this specification updates the [RFC4288]
      and [RFC4289] registration process for XML-based MIME types.

11. Security Considerations

      XML, as a subset of SGML, has all of the same security considerations
      as specified in [RFC1874], and likely more, due to its ubiquitous
      deployment.

      To paraphrase section 3 of RFC 1874, XML MIME entities contain
      information to be parsed and processed by the recipient's XML system.
      These entities may contain and such systems may permit explicit
      system level commands to be executed while processing the data. To
      the extent that an XML system will execute arbitrary command strings,

recipients of XML MIME entities may be a risk. In general, it may be
possible to specify commands that perform unauthorized file
operations or make changes to the display processor's environment
that affect subsequent operations.

In general, any information stored outside of the direct control of
the user -- including CSS style sheets, XSL transformations, entity
declarations, and DTDs -- can be a source of insecurity, by either
obvious or subtle means. For example, a tiny "whiteout attack"
modification made to a "master" style sheet could make words in
critical locations disappear in user documents, without directly
modifying the user document or the stylesheet it references. Thus,
the security of any XML document is vitally dependent on all of the
documents recursively referenced by that document.

The entity lists and DTDs for XHTML 1.0 [XHTML], for instance, are
likely to be a commonly used set of information. Many developers
will use and trust them, few of whom will know much about the level
of security on the W3C's servers, or on any similarly trusted
repository.

The simplest attack involves adding declarations that break
validation. Adding extraneous declarations to a list of character
entities can effectively "break the contract" used by documents. A
tiny change that produces a fatal error in a DTD could halt XML
processing on a large scale. Extraneous declarations are fairly
obvious, but more sophisticated tricks, like changing attributes from
being optional to required, can be difficult to track down. Perhaps
the most dangerous option available to crackers is redefining default
values for attributes: e.g., if developers have relied on defaulted
attributes for security, a relatively small change might expose
enormous quantities of information.

Apart from the structural possibilities, another option, "entity
spoofing," can be used to insert text into documents, vandalizing and
perhaps conveying an unintended message. Because XML 1.0 permits
multiple entity declarations, and the first declaration takes
precedence, it's possible to insert malicious content where an entity
is used, such as by inserting the full text of Winnie the Pooh in
every occurrence of —.

Use of the digital signatures work currently underway by the xmldsig
working group may eventually ameliorate the dangers of referencing
external documents not under one's own control.

Use of XML is expected to be varied, and widespread. XML is under
scrutiny by a wide range of communities for use as a common syntax
for community-specific metadata. For example, the Dublin Core
[RFC5013] group is using XML for document metadata, and a new effort

has begun that is considering use of XML for medical information.
Other groups view XML as a mechanism for marshalling parameters for

remote procedure calls. More uses of XML will undoubtedly arise.

Security considerations will vary by domain of use. For example, XML
medical records will have much more stringent privacy and security
considerations than XML library metadata. Similarly, use of XML as a
parameter marshalling syntax necessitates a case by case security
review.

XML may also have some of the same security concerns as plain text.
Like plain text, XML can contain escape sequences that, when
displayed, have the potential to change the display processor
environment in ways that adversely affect subsequent operations.
Possible effects include, but are not limited to, locking the
keyboard, changing display parameters so subsequent displayed text is
unreadable, or even changing display parameters to deliberately
obscure or distort subsequent displayed material so that its meaning
is lost or altered. Display processors SHOULD either filter such
material from displayed text or else make sure to reset all important
settings after a given display operation is complete.

Some terminal devices have keys whose output, when pressed, can be
changed by sending the display processor a character sequence. If
this is possible the display of a text object containing such
character sequences could reprogram keys to perform some illicit or
dangerous action when the key is subsequently pressed by the user.
In some cases not only can keys be programmed, they can be triggered
remotely, making it possible for a text display operation to directly
perform some unwanted action. As such, the ability to program keys
SHOULD be blocked either by filtering or by disabling the ability to
program keys entirely.

Note that it is also possible to construct XML documents that make
use of what XML terms "entity references" (using the XML meaning of
the term "entity" as described in Section 2), to construct repeated
expansions of text. Recursive expansions are prohibited by [XML] and
XML processors are required to detect them. However, even non-
recursive expansions may cause problems with the finite computing
resources of computers, if they are performed many times. (Entity A
consists of 100 copies of entity B, which in turn consists of 100
copies of entity C, and so on)

12. References

12.1. Normative References

[ASCII]   "US-ASCII. Coded Character Set -- 7-Bit American Standard
            Code for Information Interchange", ANSI X3.4-1986, 1986.

[CSS]   Bos, B., Lie, H.W., Lilley, C., and I. Jacobs, "Cascading

Style Sheets, level 2 (CSS2) Specification", World Wide
Web Consortium Recommendation REC-CSS2, May 1998, <http://
www.w3.org/TR/REC-CSS2/>.

[HTTPbis] Fielding, R., "Hypertext Transfer Protocol -- HTTP/1.2?",
           RFC ???, January 2013.

[ISO8859] "ISO-8859. International Standard -- Information
           Processing -- 8-bit Single-Byte Coded Graphic Character
           Sets -- Part 1: Latin alphabet No. 1, ISO-8859-1:1987",
           1987.

[PNG]   Boutell, T., "PNG (Portable Network Graphics)
           Specification", World Wide Web Consortium Recommendation
           REC-png, October 1996, <http://www.w3.org/TR/REC-png>.

[RFC1652] Klensin, J., Freed, N., Rose, M., Stefferud, E., and D.
           Crocker, "SMTP Service Extension for 8bit-MIMEtransport",
           RFC 1652, July 1994.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
           Extensions (MIME) Part One: Format of Internet Message
           Bodies", RFC 2045, November 1996.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
           Extensions (MIME) Part Two: Media Types", RFC 2046,
           November 1996.

[RFC2077] Nelson, S.D., Parks, C., and Mitra, "The Model Primary
           Content Type for Multipurpose Internet Mail Extensions",
           RFC 2077, January 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2445] Dawson, F. and D. Stenerson, "Internet Calendaring and
           Scheduling Core Object Specification (iCalendar)", RFC
           2445, November 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Nielsen, H.,
           Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
           Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3023] Murata, M., St.Laurent, S., and D. Kohn, "XML Media
           Types", January 2001.

[RFC3501] Crispin, M., "Internet Message Access Protocol - Version
           4rev1", RFC 3501, March 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO

                10646", RFC 3629, November 2003.

   [RFC3977]   Feather, B., "Network News Transfer Protocol", RFC 3977,
                October 2006.

   [RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
                Resource Identifiers (URI): Generic Syntax.", RFC 3986,
                January 2005.

   [RFC3987]   DUeerst, M. and M. Suignard, "Internationalized Resource
                Identifiers (IRIs)", RFC 3987, July 2005.

   [RFC4288]   Freed, N. and J. Klensin, "Media Type Specifications and
                Registration Procedures", RFC 4288, December 2005.

   [RFC4289]   Freed, N. and J. Klensin, "Multipurpose Internet Mail
                Extensions (MIME) Part Four: Registration Procedures", RFC
                4289, December 2005.

   [RFC4918]   Dusseault, L., "HTTP Extensions for Distributed Authoring
                -- WEBDAV", RFC 4918, June 2007.

   [RFC5321]   Klensin, J., "Simple Mail Transfer Protocol", RFC 5321,
                October 2008.

   [RFC6657]   Melnikov, A. and J. Reschke, "Update to MIME regarding
                "charset" Parameter Handling in Textual Media Types", RFC
                6657, July 2012, <http://www.rfc-editor.org/rfc/
                rfc6657.txt>.

   [SGML]   International Standard Organization, "Information
                Processing -- Text and Office Systems -- Standard
                Generalized Markup Language (SGML)", ISO 8879, October
                1986.

   [TAGMIME]   Bray, T., Ed., "Internet Media Type registration,
                consistency of use", April 2004, <http://www.w3.org/2001/
                tag/2004/0430-mime>.

   [UML]   Object Management Group, "OMG Unified Modeling Language
                Specification, Version 1.3", OMG Specification ad/
                99-06-08, June 1999, <http://www.omg.org/uml/>.

   [XBase]   Marsh, J. and R. Tobin, "XML Base", World Wide Web
                Consortium Recommendation xmlbase, January 2009, <http://
                www.w3.org/TR/xmlbase>.

   [XHTML]   Pemberton, S. and et al, "XHTML 1.0: The Extensible
                HyperText Markup Language", World Wide Web Consortium
                Recommendation xhtml1, December 1999, <http://www.w3.org/

TR/xhtml1>.

   [XLink]   DeRose, S., Maler, E., Orchard, D., and N. Walsh, "XML
              Linking Language (XLink) Version 1.1", World Wide Web
              Consortium Recommendation xlink11, May 2010, <http://
              www.w3.org/TR/xlink/>.

   [XML1.1]   Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E.,
              Yergeau, F., and J. Cowan, "Extensible Markup Language
              (XML) 1.1", World Wide Web Consortium Recommendation REC-
              xml, September 2006, <http://www.w3.org/TR/xml11>.

   [XML]   Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E.,
              and F. Yergeau, "Extensible Markup Language (XML) 1.0
              (Fifth Edition)", World Wide Web Consortium Recommendation
              REC-xml, November 2008, <http://www.w3.org/TR/REC-xml>.

   [XPointerElement]
              Grosso, P., Maler, E., Marsh, J., and N. Walsh, "XPointer
              element() Scheme", World Wide Web Consortium
              Recommendation REC-XPointer-Element, March 2003, <http://
              www.w3.org/TR/xptr-element/>.

   [XPointerFramework]
              Grosso, P., Maler, E., Marsh, J., and N. Walsh, "XPointer
              Framework", World Wide Web Consortium Recommendation REC-
              XPointer-Framework, March 2003, <http://www.w3.org/TR/
              xptr-framework/>.

   [XPtrReg] Hazael-Massieux, D., "XPointer Registry", 2005, <http://
              www.w3.org/2005/04/xpointer-schemes/>.

12.2. Informative References

   [MathML]   Carlisle, D., Ion, P., and R. Miner, "Mathematical Markup
              Language (MathML) Version 3.0", World Wide Web Consortium
              Recommendation MathML, October 2010, <http://www.w3.org/TR
              /MathML/>.

   [RFC1557] Choi, U., Chon, K., and H. Park, "Korean Character
              Encoding for Internet Messages", RFC 1557, December 1993.

   [RFC1874] Levinson, E., "SGML Media Types", RFC 1874, December 1995.

   [RFC2130] Weider, C., Cecilia Preston, C., Simonsen, K., Alvestrand,
              H., Atkinson, R., Crispin, M., and P. Svanberg, "The
              Report of the IAB Character Set Workshop held 29 February
              - 1 March, 1996", RFC 2130, April 1997.

   [RFC2376] Whitehead, E. and M. Murata, "XML Media Types", RFC 2376,
              July 1998.

   [RFC2703] Klyne, G., "Protocol-independent Content Negotiation
              Framework", RFC 2703, September 1999.

[RFC2781]   Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO
            10646", RFC 2781, Februrary 2000.

[RFC2801]   Burdett, D., "Internet Open Trading Protocol - IOTP
            Version 1.0", RFC 2801, April 2000.

[RFC3870]   3870, A., "application/rdf+xml Media Type Registration",
            RFC 3870, September 2004.

[RFC3902]   Baker, M. and M. Nottingham, "The "application/soap+xml"
            media type", RFC 3902, September 2004.

[RFC5013]   Kunze, J. and T. Baker, "Dublin Core Metadata for Resource
            Discovery", RFC 5013, August 2007.

[SVG]   Dahlstroem, E. and others. , "Scalable Vector Graphics
            (SVG) 1.1 Specification (Second edition)", World Wide Web
            Consortium Recommendation SVG, August 2011, <http://
            www.w3.org/TR/SVG/>.

[XSLT]   Kay, M., "XSL Transformations (XSLT) Version 2.0", World
            Wide Web Consortium Recommendation xslt20, January 2007,
            <http://www.w3.org/TR/xslt20/>.

Appendix A. Why Use the '+xml' Suffix for XML-Based MIME Types?

Although the use of a suffix was not considered as part of the
original MIME architecture, this choice is considered to provide the
most functionality with the least potential for interoperability
problems or lack of future extensibility. The alternatives to the
'+xml' suffix and the reason for its selection are described below.

A.1. Why not just use text/xml or application/xml and let the XML processor dispatch to the correct application based on the referenced DTD?

text/xml and application/xml remain useful in many situations,
especially for document-oriented applications that involve combining
XML with a stylesheet in order to present the data. However, XML is
also used to define entirely new data types, and an XML-based format
such as image/svg+xml fits the definition of a MIME media type
exactly as well as image/png [PNG] does. (Note that image/svg+xml
is not yet registered.) Although extra functionality is available for
MIME processors that are also XML processors, XML-based media types
-- even when treated as opaque, non-XML media types -- are just as
useful as any other media type and should be treated as such.

Since MIME dispatchers work off of the MIME type, use of text/xml or
application/xml to label discrete media types will hinder correct

dispatching and general interoperability. Finally, many XML
documents use neither DTDs nor namespaces, yet are perfectly legal
XML.

A.2. **Why not create a new subtree (e.g., image/xml.svg) to represent** XML MIME types?

   The subtree under which a media type is registered -- IETF, vendor (* /vnd.*), or personal (*/prs.*); see [RFC4288] and [RFC4289] for details -- is completely orthogonal from whether the media type uses XML syntax or not. The suffix approach allows XML document types to be identified within any subtree. The vendor subtree, for example, is likely to include a large number of XML-based document types. By using a suffix, rather than setting up a separate subtree, those types may remain in the same location in the tree of MIME types that they would have occupied had they not been based on XML.

A.3. **Why not create a new top-level MIME type for XML-based media** types?

   The top-level MIME type (e.g., model/* [RFC2077]) determines what kind of content the type is, not what syntax it uses. For example, agents using image/* to signal acceptance of any image format should certainly be given access to media type image/svg+xml, which is in all respects a standard image subtype. It just happens to use XML to describe its syntax. The two aspects of the media type are completely orthogonal.

   XML-based data types will most likely be registered in ALL top-level categories. Potential, though currently unregistered, examples could include application/mathml+xml [MathML], model/uml+xml [UML], and image/svg+xml [SVG].

A.4. **Why not just have the MIME processor 'sniff' the content to** determine whether it is XML?

   Rather than explicitly labeling XML-based media types, the processor could look inside each type and see whether or not it is XML. The processor could also cache a list of XML-based media types.

   Although this method might work acceptably for some mail applications, it would fail completely in many other uses of MIME. For instance, an XML-based web crawler would have no way of determining whether a file is XML except to fetch it and check. The same issue applies in some IMAP4 [RFC3501] mail applications, where the client first fetches the MIME type as part of the message structure and then decides whether to fetch the MIME entity. Requiring these fetches just to determine whether the MIME type is XML could have significant bandwidth and latency disadvantages in many situations.

Sniffing XML also isn't as simple as it might seem. DOCTYPE
declarations aren't required, and they can appear fairly deep into a
document under certain unpreventable circumstances. (E.g., the XML
declaration, comments, and processing instructions can occupy space
before the DOCTYPE declaration.) Even sniffing the DOCTYPE isn't
completely reliable, thanks to a variety of issues involving default
values for namespaces within external DTDs and overrides inside the
internal DTD. Finally, the variety in potential character encodings
(something XML provides tools to deal with), also makes reliable
sniffing less likely.

A.5. **Why not use a MIME parameter to specify that a media type uses XML**
     syntax?

For example, one could use "Content-Type: application/iotp;
alternate-type=text/xml" or "Content-Type: application/iotp;
syntax=xml".

Section 5 of [RFC2045] says that "Parameters are modifiers of the
media subtype, and as such do not fundamentally affect the nature of
the content". However, all XML-based media types are by their nature
always XML. Parameters, as they have been defined in the MIME
architecture, are never invariant across all instantiations of a
media type.

More practically, very few if any MIME dispatchers and other MIME
agents support dispatching off of a parameter. While MIME agents on
the receiving side will need to be updated in either case to support
(or fall back to) generic XML processing, it has been suggested that
it is easier to implement this functionality when acting off of the
media type rather than a parameter. More important, sending agents
require no update to properly tag an image as "image/svg+xml", but
few if any sending agents currently support always tagging certain
content types with a parameter.

A.6. **How about labeling with parameters in the other direction (e.g.,**
     application/xml; Content-Feature=iotp)?

This proposal fails under the simplest case, of a user with neither
knowledge of XML nor an XML-capable MIME dispatcher. In that case,
the user's MIME dispatcher is likely to dispatch the content to an
XML processing application when the correct default behavior should
be to dispatch the content to the application responsible for the
content type (e.g., an ecommerce engine for application/iotp+xml
[RFC2801], once this media type is registered).

Note that even if the user had already installed the appropriate
application (e.g., the ecommerce engine), and that installation had
updated the MIME registry, many operating system level MIME
registries such as .mailcap in Unix and HKEY_CLASSES_ROOT in Windows
do not currently support dispatching off a parameter, and cannot
easily be upgraded to do so. And, even if the operating system were
upgraded to support this, each MIME dispatcher would also separately
need to be upgraded.

A.7. **How about a new superclass MIME parameter that is defined to apply**
      to all MIME types (e.g., Content-Type: application/iotp;
      $superclass=xml)?

This combines the problems of Appendix A.5 and Appendix A.6.

If the sender attaches an image/svg+xml file to a message and
includes the instructions "Please copy the French text on the road
sign", someone with an XML-aware MIME client and an XML browser but
no support for SVG can still probably open the file and copy the
text. By contrast, with superclasses, the sender must add superclass
support to her existing mailer AND the receiver must add superclass
support to his before this transaction can work correctly.

If the receiver comes to rely on the superclass tag being present and
applications are deployed relying on that tag (as always seems to
happen), then only upgraded senders will be able to interoperate with
those receiving applications.

A.8. **What about adding a new parameter to the Content-Disposition**
      header or creating a new Content-Structure header to indicate XML
      syntax?

This has nearly identical problems to Appendix A.7, in that it
requires both senders and receivers to be upgraded, and few if any
operating systems and MIME dispatchers support working off of
anything other than the MIME type.

A.9. **How about a new Alternative-Content-Type header?**

This is better than Appendix A.8, in that no extra functionality
needs to be added to a MIME registry to support dispatching of
information other than standard content types. However, it still
requires both sender and receiver to be upgraded, and it will also
fail in many cases (e.g., web hosting to an outsourced server), where
the user can set MIME types (often through implicit mapping to file
extensions), but has no way of adding arbitrary HTTP headers.

A.10. **How about using a conneg tag instead (e.g., accept-features:**
       (syntax=xml))?

When the conneg protocol is fully defined, this may potentially be a
reasonable thing to do. But given the limited current state of
conneg [RFC2703] development, it is not a credible replacement for a
MIME-based solution.

Also, note that adding a content-type parameter doesn't work with
conneg either, since conneg only deals with media types, not their
parameters. This is another illustration of the limits of parameters
for MIME dispatchers.

A.11. How about a third-level content-type, such as text/xml/rdf?

MIME explicitly defines two levels of content type, the top-level for
the kind of content and the second-level for the specific media type.
[RFC4288] and [RFC4289] extends this in an interoperable way by using
prefixes to specify separate trees for IETF, vendor, and personal
registrations. This specification also extends the two-level type by
using the '+xml' suffix. In both cases, processors that are unaware
of these later specifications treat them as opaque and continue to
interoperate. By contrast, adding a third-level type would break the
current MIME architecture and cause numerous interoperability
failures.

A.12. Why use the plus ('+') character for the suffix '+xml'?

As specified in Section 5.1 of [RFC2045], a tspecial can't be used:

```
tspecials :=
"(" / ")" / "<" / ">" / "@" /
"," / ";" / ":" / "\" / <">
"/" / "[" / "]" / "?"  / "="
```

It was thought that "." would not be a good choice since it is
already used as an additional hierarchy delimiter. Also, "*" has a
common wildcard meaning, and "-" and "_" are common word separators
and easily confused. The characters %'`#& are frequently used for
quoting or comments and so are not ideal.

That leaves: ~!$^+{}|

Note that "-" is used heavily in the current registry. "$" and "_"
are used once each. The others are currently unused.

It was thought that '+' expressed the semantics that a MIME type can
be treated (for example) as both scalable vector graphics AND ALSO as
XML; it is both simultaneously.

A.13. What is the semantic difference between application/foo and
      application/foo+xml?

MIME processors that are unaware of XML will treat the '+xml' suffix
as completely opaque, so it is essential that no extra semantics be
assigned to its presence. Therefore, application/foo and application
/foo+xml SHOULD be treated as completely independent media types.
Although, for example, text/calendar+xml could be an XML version of
text/calendar [RFC2445], it is possible that this (hypothetical) new
media type would include new semantics as well as new syntax, and in
any case, there would be many applications that support text/calendar
but had not yet been upgraded to support text/calendar+xml.

A.14. **What happens when an even better markup language (e.g., EBML) is**
       defined, or a new category of data?

In the ten years that MIME has existed, XML is the first generic data
format that has seemed to justify special treatment, so it is hoped
that no further suffixes will be necessary. However, if some are
later defined, and these documents were also XML, they would need to
specify that the '+xml' suffix is always the outermost suffix (e.g.,
application/foo+ebml+xml not application/foo+xml+ebml). If they were
not XML, then they would use a regular suffix (e.g., application/
foo+ebml).

A.15. **Why must I use the '+xml' suffix for my new XML-based media type?**

You don't have to, but unless you have a good reason to explicitly
disallow generic XML processing, you should use the suffix so as not
to curtail the options of future users and developers.

Whether the inventors of a media type, today, design it for dispatch
to generic XML processing machinery (and most won't) is not the
critical issue. The core notion is that the knowledge that some
media type happens to use XML syntax opens the door to unanticipated
kinds of processing beyond those envisioned by its inventors, and on
this basis identifying such encoding is a good and useful thing.

Developers of new media types are often tightly focused on a
particular type of processing that meets current needs. But there is
no need to rule out generic processing as well, which could make your
media type more valuable over time. It is believed that registering
with the '+xml' suffix will cause no interoperability problems
whatsoever, while it may enable significant new functionality and
interoperability now and in the future. So, the conservative
approach is to include the '+xml' suffix.

Appendix B. Changes from RFC 3023

There are numerous and significant differences between this
specification and [RFC3023], which it obsoletes. This appendix
summarizes the major differences only.

First, XPointer ([XPointerFramework] and [XPointerElement] has been
added as fragment identifier syntax for "application/xml", and the
XPointer Registry ([XPtrReg]) mentioned. Second, [XBase] has been
added as a mechanism for specifying base URIs. Third, the language
regarding charsets was updated to correspond to the W3C TAG finding
Internet Media Type registration, consistency of use [TAGMIME].
Fourth, many references are updated.

Appendix C. **Acknowledgements**

This specification reflects the input of numerous participants to the
ietf-xml-mime@imc.org mailing list, though any errors are the
responsibility of the authors. Special thanks to:

Mark Baker, James Clark, Dan Connolly, Martin Duerst, Ned Freed,
Yaron Goland, Rick Jelliffe, Larry Masinter, David Megginson, Keith
Moore, Chris Newman, Gavin Nicol, Marshall Rose, Jim Whitehead and
participants of the XML activity and the TAG at the W3C.

Jim Whitehead and Simon St.Laurent are editors of [RFC2376] and
[RFC3023], respectively.

Authors' Addresses

Chris Lilley
World Wide Web Consortium
2004, Route des Lucioles - B.P. 93 06902
Sophia Antipolis Cedex
France

Email: chris@w3.org
URI: http://www.w3.org/People/chris/

MURATA Makoto (FAMILY Given)
International University of Japan

Email: eb2m-mrt@asahi-net.or.jp

Alexey Melnikov
Isode Ltd.

Email: alexey.melnikov@isode.com
URI: http://www.melnikov.ca/

Henry S. Thompson
University of Edinburgh

      Email: ht@inf.ed.ac.uk
      URI: http://www.ltg.ed.ac.uk/~ht/