XML-binary Optimized Packaging

W3C Recommendation 25 January 2005

This version:
http://www.w3.org/TR/2005/REC-xop10-20050125/
Latest version:
http://www.w3.org/TR/xop10/
Previous version:
http://www.w3.org/TR/2004/PR-xop10-20041116/

Editors:
Martin Gudgin, Microsoft
Noah Mendelsohn, IBM
Mark Nottingham, BEA
Hervé Ruellan, Canon

Please refer to the **errata** for this document, which may include normative corrections.

See also **translations**.

Abstract

This document defines the XML-binary Optimized Packaging (XOP) convention, a means of more efficiently serializing XML Infosets that have certain types of content.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document is a Recommendation of the W3C. It has been reviewed by W3C Members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document has been produced by the XML Protocol Working Group (WG) as part of the W3C Web Services Activity. The English version of this specification is the only normative version. However, for translations of this document, see http://www.w3.org/2003/03/Translations/byTechnology?technology=xop10.

Please report errors in this document to xmlp-comments@w3.org (archive). The errata list for this edition is available at http://www.w3.org/2005/01/xop10-errata

This document is based upon the XML-binary Optimized Packaging Proposed Recommendation of 16 November 2004. Feedback received during that review resulted in no changes. Evidence of interoperation between at least two implementations of this specification are documented in the Implementation Summary. Changes between these two versions are described in a diff document.

This document has been produced under the 24 January 2002 CPP as amended by the W3C Patent Policy Transition Procedure. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) with respect to this specification should disclose the information in accordance with section 6 of the W3C Patent Policy. Patent disclosures relevant to this specification may be found on the Working Group's patent disclosure page.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR/.

Table of Contents

Appendices

1 Introduction

This specification defines the XML-binary Optimized Packaging (XOP) convention, a means of more efficiently serializing XML Infosets (see [XMLInfoSet]) that have certain types of content.

A XOP package is created by placing a serialization of the XML Infoset inside of an extensible packaging format (such a MIME Multipart/Related, see [RFC 2387]). Then, selected portions of its content that are base64-encoded binary data are extracted and re-encoded (i.e., the data is decoded from base64) and placed into the package. The locations of those selected portions are marked in the XML with a special element that links to the packaged data using URIs.

In a number of important XOP applications, binary data need never be encoded in base64 form. If the data to be included is already available as a binary octet stream, then either an application or other software acting on its behalf can directly copy that data into a XOP package, at the same time preparing suitable linking elements for use in the root part; when parsing a XOP package, the binary data can be made available directly to applications, or, if appropriate, the base64 binary character representation can be computed from the binary data.

However, at the conceptual level, this binary data can be thought of as being base64-encoded in the XML Document. As this conceptual form might be needed during some processing of the XML Document (e.g., for signing the XML document), it is necessary to have a one to one correspondence between XML Infosets and XOP Packages. Therefore, the conceptual representation of such binary data is as if it were base64-encoded, using the canonical lexical form of XML Schema `base64Binary` datatype (see [XML Schema Part 2: Datatypes Second Edition] 3.2.16 base64Binary). In the reverse direction, XOP is capable of optimizing only base64-encoded Infoset data that is in the canonical lexical form.

Only element content can be optimized; attributes, non-base64-compatible character data, and data not in the canonical representation of the `base64Binary` datatype cannot be successfully optimized by XOP.

The remainder of this specification is organized in the following fashion:

- Section 2 describes the XOP Infoset, which preserves the non-optimized content and structure of the original XML Infoset.

- Section 3 specifies the XOP processing model.

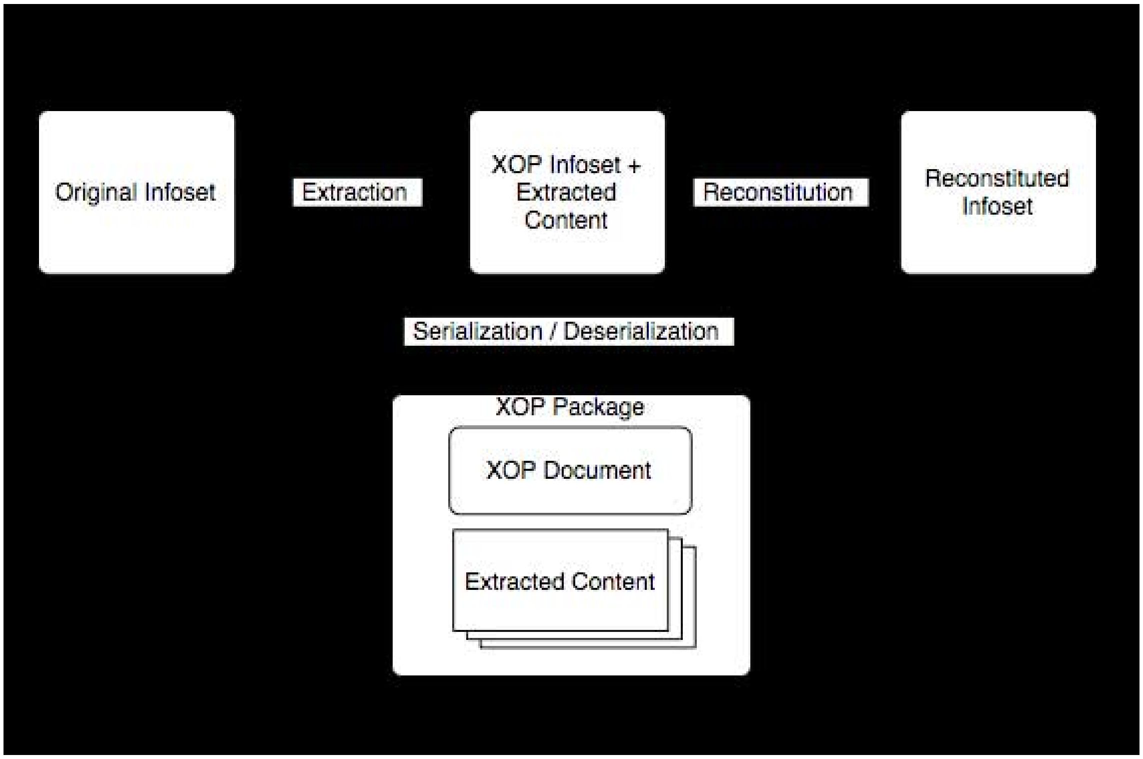
- Section 4 of this specification describes the form of the XOP Package.
- Section 5 describes how XOP Documents are identified.
- Section 6 explores the security considerations of using the XOP convention.

1.1 Terminology

This specification uses terminology from the XML Infoset (see [XMLInfoSet]) when discussing XML content and structure. This is only a convention for clear specification of XOP behavior.

The following terms are used in this specification:

- **Original XML Infoset** - An XML Infoset to be optimized.
- **Optimized Content** - Content which has been removed from the XML Infoset.
- **XOP Infoset** - The Original Infoset with any Optimized Content removed and replaced by `xop:Include` *element information items*.
- **XOP Document** - A serialization of the XOP Infoset using any W3C recommendation-level version of XML.
- **XOP Package** - A package containing the XOP Document and any Optimized Content. As a whole, the XOP Package is an alternate serialization of the Original Infoset.
- **Reconstituted XML Infoset** - An XML Infoset that has been constructed from the parts of a XOP Package.



1.2 Example

Example 1 shows an XML Infoset prior to XOP processing. Example 2 shows the same

Infoset, serialized using the XOP format in a MIME Multipart/Related package. The base64-encoded content of the `m:photo` and `m:sig` elements have been replaced by a `xop:Include` element, while the binary octets have been serialized in separate MIME parts. Note that those examples use [Assigning Media Types to Binary Data in XML] to identify the media type of the content of the `m:photo` and `m:sig` elements. Note also that the sample base64 data is smaller than would be typical and the binary octets are not shown; in practice, the optimized form is likely to be much smaller than the original.

Example: XML Infoset prior to XOP processing (Example 1, SOAP)

```
<soap:Envelope
    xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
    xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
  <soap:Body>
    <m:data xmlns:m='http://example.org/stuff'>
      <m:photo
  xmlmime:contentType='image/png'>/aWKKapGGyQ=</m:photo>
      <m:sig
  xmlmime:contentType='application/pkcs7-signature'>Faa7vROi2VQ=</m:sig>
    </m:data>
  </soap:Body>
</soap:Envelope>
```

Example: XML Infoset serialized as a XOP package (Example 2, SOAP)

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<mymessage.xml@example.org>";
    startinfo="application/soap+xml; action=\"ProcessData\""
Content-Description: A SOAP message with my pic and sig in it

--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<soap:Envelope
    xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
    xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
  <soap:Body>
    <m:data xmlns:m='http://example.org/stuff'>
      <m:photo
  xmlmime:contentType='image/png'><xop:Include
    xmlns:xop='http://www.w3.org/2004/08/xop/include'
    href='cid:http://example.org/me.png'/></m:photo>
      <m:sig
  xmlmime:contentType='application/pkcs7-signature'><xop:Include
    xmlns:xop='http://www.w3.org/2004/08/xop/include'
    href='cid:http://example.org/my.hsh'/></m:sig>
    </m:data>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/me.png>

// binary octets for png
```

```
--MIME_boundary
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/my.hsh>

// binary octets for signature

--MIME_boundary--
```

Example 3 shows an XML Infoset prior to XOP processing. Example 4 shows the same Infoset, serialized using the XOP format in a MIME Multipart/Related package. The base64-encoded content of the `m:photo` and `m:sig` elements have been replaced by a `xop:Include` element, while the binary octets have been serialized in separate MIME parts. Note also that the sample base64 data is smaller than would be typical and the binary octets are not shown; in practice, the optimized form is likely to be much smaller than the original.

Example: XML Infoset prior to XOP processing (Example 3, non-SOAP)

```
<m:data xmlns:m='http://example.org/stuff'>
  <m:photo>/aWKKapGGyQ=</m:photo>
  <m:sig>Faa7vROi2VQ=</m:sig>
</m:data>
```

Example: XML Infoset serialized as a XOP package (Example 4, non-SOAP)

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<mymessage.xml@example.org>";
    start-info="text/xml"
Content-Description: An XML document with my pic and sig in it

--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="text/xml"
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<m:data xmlns:m='http://example.org/stuff'>
  <m:photo><xop:Include
  xmlns:xop='http://www.w3.org/2004/08/xop/include'
  href='cid:http://example.org/me.png'/></m:photo>
  <m:sig><xop:Include
  xmlns:xop='http://www.w3.org/2004/08/xop/include'
  href='cid:http://example.org/my.hsh'/></m:sig>
</m:data>

--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/me.png>

// binary octets for png

--MIME_boundary
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/my.hsh>

// binary octets for signature
```

```
--MIME_boundary--
```

1.3 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

This specification uses a number of namespace prefixes throughout; they are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Prefixes and Namespaces used in this specification.

Prefix	Namespace
	Notes
xop	"http://www.w3.org/2004/08/xop/include"
	A non-normative XML Schema [XML Schema Part 1: Structures Second Edition], [XML Schema Part 2: Datatypes Second Edition] document for the "http://www.w3.org/2004/08/xop/include" namespace can be found at http://www.w3.org/2004/08/xop/include. Note that XML Schema > currently provides only for validation of XML 1.0 Infosets; accordingly, > the schema may not be usable > with XOP Infosets corresponding to later versions of XML.
xmlmime	"http://www.w3.org/2004/11/xmlmime"
	The namespace for the content type attribute.
soap	"http://www.w3.org/2003/05/soap-envelope"
	The SOAP 1.2 namespace[SOAP12].
xs	"http://www.w3.org/2001/XMLSchema"
	The namespace of XML Schema data types [XML Schema Part 2: Datatypes Second Edition].

Editorial note: HR	
Note that the "http://www.w3.org/2004/11/xmlmime" URI is not final and will be changed to track the evolution of the "Assigning Media Types to Binary Data in XML" document.	

2 XOP Infoset Constructs

XOP operates by extracting the Optimized Content from the Original Infoset to create the XOP Infoset. In particular, the *character information item* children of *element information items* to be optimized are removed and replaced with an *element information item* named `xop:Include`. The `xop:Include` *element information item* contains an *attribute information item* with a link to the part of the XOP Package that carries a binary representation of the data removed from the original *element information item*. Details of the construction and processing of XOP serializations are provided in **3 XOP Processing Model**.

The Infoset used as input to XOP processing MUST NOT contain any *element information item* with a [namespace name] property of "http://www.w3.org/2004/08/xop/include" and a [local name] property of `Include`. Infosets containing such *element information items* cannot be serialized using XOP. This is because during infoset reconstruction a processor is unable to differentiate between `xop:Include` *element information items* inserted during XOP package construction and those that were part of the original infoset.

The following subsections provide formal definitions for allowable content in the *element information item* and *attribute information items* used to construct a XOP serialization; content not explicitly specified is disallowed. A non-normative XML Schema for [Extensible Markup Language (XML) 1.0 (Third Edition)] serializations of those *element information item* and *attribute information items* can be found at http://www.w3.org/2004/08/xop/include.

2.1 `xop:Include` element information item

The `xop:Include` *element information item* has:

- A [local name] of `Include`.
- A [namespace name] of "http://www.w3.org/2004/08/xop/include".
- One or more *attribute information items* amongst its [attributes] property as follows:
    - A mandatory `href` *attribute information item* (see **2.2 href attribute information item**).
    - Zero or more additional namespace qualified *attribute information items*. Any such *attribute information items* MUST NOT have a [namespace name] of "http://www.w3.org/2004/08/xop/include", MUST NOT change the semantics of processing the `xop:Include` *element information item* and MUST be ignored if not recognized.
- Zero or more namespace qualified *element information items* in its [children] property. Any such *element information items* MUST NOT have a [namespace name] of "http://www.w3.org/2004/08/xop/include", MUST NOT change the semantics of processing the `xop:Include` *element information item* and MUST be ignored if not recognized.

2.2 `href` attribute information item

The `href` *attribute information item* has:

- A [local name] of `href`.
- An empty [namespace name].
- A [normalized value] which is a representation of a URI (see [RFC 2396] as amended by [RFC 2732]) referencing the part of the package containing the data logically included by the [owner element] (i.e., the `xop:Include` *element information item*). The [normalized value] MUST be a valid URI per the cid: URI scheme (see [RFC 2392]). In addition, the [normalized value] MUST be a valid lexical form of the XML Schema `xs:anyURI` datatype (see [XML Schema Part 2: Datatypes Second Edition]3.2.17 anyURI).
- An [owner element] which is the `xop:Include` *element information item* containing the *attribute information item*.

3 XOP Processing Model

This section describes the processing model for creating XOP Packages and interpreting XOP Packages. Unless otherwise stated, the result of such processing MUST be semantically equivalent to performing the specified steps separately, and in the order given.

3.1 Creating XOP Packages

To create a XOP Package from an Original XML Infoset:

1. Ensure that the Original XML Infoset contains no *element information item* with a [namespace name] of "http://www.w3.org/2004/08/xop/include" and a [local name] of `Include`. As discussed in **2 XOP Infoset Constructs**, XML Infosets with such *element information items* cannot be represented using XOP.
2. Create an empty package.
3. Identify within the Original XML Infoset the *element information items* to be optimized. To be optimized, the characters comprising the [children] of the *element information item* MUST be in the canonical form of `xs:base64Binary` (see [XML Schema Part 2: Datatypes Second Edition]3.2.16 base64Binary) and MUST NOT contain any whitespace characters, preceding, inline with or following the non-whitespace content.
4. Create a XOP Infoset which is a copy of the Original XML Infoset, but with the [children] of each *element information item* identified in the previous step replaced by a `xop:Include` *element information item* (see **2.1 xop:Include element information item**) constructed as follows:
    a. Transform the replaced characters into binary data by processing them as base64-encoded data.
    b. Serialize the binary data into a new part of the package, with appropriate metadata corresponding to the [normalized value] of the `href` *attribute information item* of the `xop:Include` *element information item* (see **2.2 href attribute information item**).
    c. If the *element information item* being optimized (i.e., the [parent] of the newly inserted `xop:Include` *element information item*) has a `xmlmime:contentType` *attribute information item*, its value SHOULD be reflected appropriately in the metadata for the part.
5. Serialize the resulting XOP Infoset into the package using any W3C recommendation-level version of XML (e.g., [Extensible Markup Language (XML) 1.0 (Third Edition)], [Extensible Markup Language (XML) 1.1]) and identify it as the root part according to the packaging mechanism's convention, labeling it with the application/xop+xml media type, as described in **5 Identifying XOP Documents**.

Additional parts MAY be added to the package to satisfy application specific requirements. Other content-specific metadata MAY be reflected in the packaging metadata as appropriate.

If content cannot be successfully encoded into the XOP package, implementations SHOULD behave as if that portion of the Original XML Infoset was not nominated for optimization.

3.2 Interpreting XOP Packages

This section specifies the means by which the Original XML Infoset can be reconstructed from a XOP Package that has been prepared according to the rules of **3.1 Creating XOP Packages**.

Note: conventions or error reporting mechanisms to be used in processing packages that incorrectly purport to be XOP Packages are beyond the scope of this specification.

To create a Reconstituted XML Infoset from a XOP Package:

1. Construct an XML Infoset by parsing the root part of the package as an XML document. The document MUST be parsed according to the level of the XML Recommendation identified by the XML declaration of that document. If no XML declaration is present, then the document MUST be parsed per [Extensible Markup Language (XML) 1.0 (Third Edition)].
2. Using that XML Infoset, for each *element information item*, E, which has, as the sole member of its [children] property, a `xop:Include` *element information item* (as defined in **2.1 xop:Include element information item**):
   a. Locate the part of the package corresponding to the URI in the `href` *attribute information item* of the `xop:Include` *element information item* (i.e., corresponding to the URI encoded in the *attribute information item*'s [normalized value]).
   b. Replace the `xop:Include` *element information item* that appears in the [children] property of E with *character information items* representing the canonical base64 encoding of the entity body of the identified package part (i.e., effectively replace the `xop:Include` *element information item* with the data reconstructed from the package part).

4 XOP Packages

XOP is capable of using a variety of underlying packaging mechanisms. Such packaging mechanisms MUST be able to represent, with full fidelity all the parts created according to **3 XOP Processing Model** (see **3.1 Creating XOP Packages**), and MUST be used in a manner that provides a means of designating a distinguished root (main, primary etc.) part.

The subsection below specifies normatively how a particular packaging mechanism, MIME Multipart/Related, is used, but does not preclude the use of other packaging mechanisms with the XOP convention.

4.1 MIME Multipart/Related XOP Packages

This section describes how MIME Multipart/Related packaging (as specified in [RFC 2387]) is used with XOP.

The root MIME part is the root part of the XOP package, MUST be a serialization of the XOP Infoset using any W3C recommendation-level version of XML (e.g., [Extensible Markup Language (XML) 1.0 (Third Edition)], [Extensible Markup Language (XML) 1.1]), and MUST be identified with a media type of "application/xop+xml" (as defined below).

The "start-info" parameter of the package's media type MUST contain the content type associated with the content's XML serialization. (i.e. it will contain the same value as the "type" parameter of the root part).

Except for purposes of determining the root MIME part, as specified by [RFC 2387], ordering of MIME parts MUST NOT be considered significant to XOP processing or to the construction of the XOP Infoset.

Part metadata is reflected in MIME header fields. Specifically, the URI used in the value of an `href` *attribute information item* on a `xop:Include` *element information item* contains a URI that uses the 'cid:' scheme (see [RFC 2392]), so the corresponding MIME part MUST have a Content-ID header field (see [RFC 2387] with a corresponding field-value.

Furthermore, if a `xmlmime:contentType` *attribute information item* is found (as described in **3 XOP Processing Model**), it SHOULD be reflected in the field value of the MIME Content-Type header.

5 Identifying XOP Documents

XOP Documents, when used in MIME-like systems, are identified with the "application/xop+xml" media type, with the required "type" parameter conveying the original XML serialisation's associated content type. Note that when the type parameter contains reserved characters, it needs to be appropriately quoted and escaped.

For example, a XOP package using MIME multipart/related packaging to serialize a SOAP 1.2 message [SOAP Version 1.2 Part 1: Messaging Framework] with an action parameter of "http://www.example.net/foo" would label the package itself with the "multipart/related" media type, and the root part with the "application/xop+xml" media type along with a type parameter containing "application/soap+xml;action=\"http://www.example.net/foo\"".

5.1 Registration

MIME media type name:

application

MIME subtype name:

xop+xml

Required parameters:

type

This parameter conveys the content type associated with the XML serialization of the XOP infoset, including parameters as appropriate.

Optional parameters:

charset

This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in RFC 3023 [RFC3023].

Encoding considerations:

Identical to those of "application/xml" as described in RFC 3023 [RFC3023], section 3.2.

Security considerations:

In addition to application-specific considerations, XOP has the same security considerations described in RFC3023 [RFC3023], section 10.

Interoperability considerations:

There are no known interoperability issues.

Published specification:

This document

Applications which use this media type:

No known applications currently use this media type.

Additional information:

File extension:

XOP

Fragment identifiers:

Identical to that of "application/xml" as described in RFC 3023 [RFC3023], section 5.

Base URI:

As specified in RFC 3023 [RFC3023], section 6.

Macintosh File Type code:

TEXT

Person and email address to contact for further information:

Mark Nottingham <mnot@pobox.com>

Intended usage:

COMMON

Author/Change controller:

The XOP specification is a work product of the World Wide Web Consortium's XML Protocol Working Group. The W3C has change control over this specification.

6 Security Considerations

6.1 XOP Package Integrity

The integrity of Infosets optimized using XOP may need to be ensured. As XOP packages can be transformed to recover such Infosets (see **3.2 Interpreting XOP Packages**), existing XML Digital Signature techniques can be used to protect them. Note, however, that a signature over the Infoset does not necessarily protect against modifications of other aspects of the XOP packaging; for example, an Infoset signature check might not protect against re-ordering of non-root parts.

In the future a transform algorithm for use with XML Signature could provide a more efficient processing model where the raw octets are digested directly.

6.2 XOP Package Confidentiality

The confidentiality of XOP Packages may need to be ensured. As such packages can be transformed to an XML Information Set, existing XML Encryption (see [XML Encryption Syntax and Processing]) techniques can be used to protect such packages. Any part of a package can be encrypted, whether it includes base64 characters or not. The resulting `CipherData` *element information item* can then be optimized because the content of such an *element information item* is base64 characters.

In the future a transform algorithm for use with XML Encryption could provide a more efficient processing model where the raw octets are encrypted directly.

A Relationship to other specifications

This appendix summarizes the XOP dependencies upon underlying specifications, the nature of appropriate payloads for XOP and the means of extending XOP.

A.1 Dependencies

The XOP convention builds upon a number of underlying specifications. They are:

- XML (e.g., [Extensible Markup Language (XML) 1.0 (Third Edition)], [Extensible Markup Language (XML) 1.1]) - The XOP Document is encoded using any W3C recommendation-level version of XML (see **3.1 Creating XOP Packages**). Formats that use XOP MUST identify which versions of XML are permissible for encoding the XOP Infoset. XOP does not constrain the use of any mechanisms defined by XML, including those explicitly allowing extensions, nor does it constrain the use of underlying specifications.

- Namespaces in XML (e.g., [Namespaces in XML], [Namespaces in XML 1.1]) - The XOP Document uses any W3C recommendation-level version of Namespaces in XML compatible with the version(s) of XML used. Formats that use XOP MUST identify which versions of Namespaces in XML are permissible for encoding the XOP Infoset. XOP does not constrain the use of any mechanisms defined by Namespaces in XML, including those explicitly allowing extensions, nor does it constrain the use of underlying specifications.

- Uniform Resource Identifiers (see [RFC 2396]) - The XOP Document uses URIs to locate parts in the XOP Package (see **2.2 href attribute information item**. XOP does not constrain the use of any mechanisms defined by URIs, including those explicitly allowing extensions, nor does it constrain the use of underlying specifications.
- Packaging Mechanism - XOP requires the use of a packaging mechanism that satisfies the requirements in **4 XOP Packages**. One such mechanism MUST be in use, but XOP does not require a specific mechanism. Formats using XOP MUST identify at least one such mechanism permissible for creating the XOP Package, and MUST specify how each allowed mechanism is to be used for building the XOP Package.

  The relationship of one such mechanism to XOP, The MIME Multipart/Related Content-type, is specified in **4.1 MIME Multipart/Related XOP Packages**.

A.2 Payload

The payload of a XOP Package is an XML Infoset. XOP constrains the range of admissible characters in the payload to those contained in the "Char" production of a W3C recommendation-level version of XML. Additionally, the Original XML Infoset cannot contain an *element information item* with a [local name] of of `Include` and a [namespace name] of "http://www.w3.org/2004/08/xop/include". Finally, portions of the payload which are nominated for optimization in XOP MUST be base64-encoded data in the canonical lexical form of XML Schema `base64Binary` datatype (see [XML Schema Part 2: Datatypes Second Edition] 3.2.16 base64Binary).

A.3 Extension

XOP Documents allow extensions to the `xop:Include` element when they do not change its semantics. Changes to the semantics MUST be identified by a new namespace URI (i.e., they MUST define a new `Include` *element information item* in another namespace).

The extensibility of the specifications underlying XOP is not constrained by their use in XOP.

A.4 Requirements

This document along with [SOAP Message Transmission Optimization Mechanism] and [SOAP Representation Header] has been produced in conjunction with the development of requirements embodied in the [SOAP Optimized Serialization Use Cases and Requirements] document.

B References

B.1 Normative References

Extensible Markup Language (XML) 1.0 (Third Edition)

Extensible Markup Language (XML) 1.0 (Third Edition), Jean Paoli, Eve Maler, Tim Bray, *et. al.*, Editors. World Wide Web Consortium, 04 February 2004. This version is http://www.w3.org/TR/2004/REC-xml-20040204. The latest version is available at http://www.w3.org/TR/REC-xml.

Extensible Markup Language (XML) 1.1

Extensible Markup Language (XML) 1.1, John Cowan, C. M. Sperberg-McQueen, François Yergeau, *et. al.*, Editors. World Wide Web Consortium, 04 February 2004. This version is http://www.w3.org/TR/2004/REC-xml11-20040204/. The latest version is available at http://www.w3.org/TR/xml11/.

SOAP Message Transmission Optimization Mechanism

SOAP Message Transmission Optimization Mechanism, Hervé Ruellan, Noah Mendelsohn, Martin Gudgin, and Mark Nottingham, Editors. World Wide Web Consortium, 25 January 2005. This version is http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/. The latest version is available at http://www.w3.org/TR/soap12-mtom/.

Resource Representation SOAP Header Block

Resource Representation SOAP Header Block, Martin Gudgin, Yves Lafon, and Anish Karmarkar, Editors. World Wide Web Consortium, 25 January 2005. This version is http://www.w3.org/TR/2005/REC-soap12-rep-20050125/. The latest version is available at http://www.w3.org/TR/soap12-rep/.

SOAP Optimized Serialization Use Cases and Requirements

SOAP Optimized Serialization Use Cases and Requirements, Tony Graham, Mark Jones, and Anish Karmarkar, Editors. World Wide Web Consortium, 08 June 2004. This version is http://www.w3.org/TR/2004/WD-soap12-os-ucr-20040608/. The latest version is available at http://www.w3.org/TR/soap12-os-ucr/.

Namespaces in XML

Namespaces in XML, Andrew Layman, Dave Hollander, and Tim Bray, Editors. World Wide Web Consortium, 14 January 1999. This version is http://www.w3.org/TR/1999/REC-xml-names-19990114. The latest version is available at http://www.w3.org/TR/REC-xml-names.

Namespaces in XML 1.1

Namespaces in XML 1.1, Richard Tobin, Andrew Layman, Tim Bray, and Dave Hollander, Editors. World Wide Web Consortium, 04 Febuary 2004. This version is http://www.w3.org/TR/2004/REC-xml-names11-20040204. The latest version is available at http://www.w3.org/TR/xml-names11/.

XML Information Set (Second Edition)

XML Information Set (Second Edition), Richard Tobin and John Cowan, Editors. World Wide Web Consortium, 04 Feb 2004. This version is http://www.w3.org/TR/2004/REC-xml-infoset-20040204. The latest version is available at http://www.w3.org/TR/xml-infoset.

XML Schema Part 1: Structures Second Edition

XML Schema Part 1: Structures Second Edition, David Beech, Murray Maloney, Henry S. Thompson, and Noah Mendelsohn, Editors. World Wide Web Consortium, 28 October 2004. This version is http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/. The latest version is available at http://www.w3.org/TR/xmlschema-1/.

XML Schema Part 2: Datatypes Second Edition

XML Schema Part 2: Datatypes Second Edition, Ashok Malhotra and Paul V. Biron, Editors. World Wide Web Consortium, 28 October 2004. This version is http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/. The latest version is available at http://www.w3.org/TR/xmlschema-2/.

Assigning Media Types to Binary Data in XML

Assigning Media Types to Binary Data in XML, Ümit Yalçınalp and Anish Karmarkar, Editors. World Wide Web Consortium, 02 November 2004. This version is http://www.w3.org/TR/2004/WD-xml-media-types-20041102. The latest version is available at http://www.w3.org/TR/xml-media-types.

RFC 2119

Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Editor. IETF, March 1997. This RFC is available at http://www.ietf.org/rfc/rfc2119.txt.

RFC 2387

The MIME Multipart/Related Content-type, E. Levinson, Editor. IETF, August 1998. This RFC is available at http://www.ietf.org/rfc/rfc2387.txt.

RFC 2557

MIME Encapsulation of Aggregate Documents, such as HTML (MHTML), J. Palme, A. Hopmann and N. Shelness, Editors. IETF, March 1999. This RFC is available at http://www.ietf.org/rfc/rfc2557.txt.

RFC 2392

Content-ID and Message-ID Uniform Resource Locators, E. Levinson, Editor. IETF, August 1998. This RFC is available at http://www.ietf.org/rfc/rfc2392.txt.

RFC 2396

Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding and L. Masinter, Editors. IETF, August 1998. This RFC is available at http://www.ietf.org/rfc/rfc2396.txt.

RFC 2732

Format for Literal IPv6 Addresses in URL's, R. Hinden, B. Carpenter and L. Masinter, Editors. IETF, December 1999. This RFC is available at http://www.ietf.org/rfc/rfc2732.txt.

B.2 Informative References

Canonical XML Version 1.0

Canonical XML Version 1.0, John Boyer, Editor. World Wide Web Consortium, 15 March 2001. This version is http://www.w3.org/TR/2001/REC-xml-c14n-20010315. The latest version is available at http://www.w3.org/TR/xml-c14n.

Exclusive XML Canonicalization Version 1.0

Exclusive XML Canonicalization Version 1.0, Joseph Reagle, Donald E. Eastlake 3rd, and John Boyer, Editors. World Wide Web Consortium, 18 July 2002. This version is http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/. The latest version is available at http://www.w3.org/TR/xml-exc-c14n.

XML Encryption Syntax and Processing

XML Encryption Syntax and Processing, Joseph Reagle and Donald Eastlake, Editors. World Wide Web Consortium, 10 December 2002. This version is http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/. The latest version is available at http://www.w3.org/TR/xmlenc-core/.

SOAP Version 1.2 Part 1: Messaging Framework

SOAP Version 1.2 Part 1: Messaging Framework, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, *et. al.*, Editors. World Wide Web Consortium, 24 June 2003. This version is http://www.w3.org/TR/2003/REC-soap12-part1-20030624/. The latest version is available at http://www.w3.org/TR/soap12-part1/.

SOAP Version 1.2 Part 2: Adjuncts

SOAP Version 1.2 Part 2: Adjuncts, Henrik Frystyk Nielsen, Noah Mendelsohn, Jean-Jacques Moreau, *et. al.*, Editors. World Wide Web Consortium, 24 June 2003. This version is http://www.w3.org/TR/2003/REC-soap12-part2-20030624/. The latest version is available at http://www.w3.org/TR/soap12-part2/.

C Acknowledgements (Non-Normative)

This specification is the work of the W3C XML Protocol Working Group.

Participants in the Working Group are (at the time of writing, and by alphabetical order): David Fallside (IBM), Tony Graham (Sun Microsystems), Martin Gudgin (Microsoft Corporation, formerly of DevelopMentor), Marc Hadley (Sun Microsystems), Gerd Hoelzing (SAP AG), John Ibbotson (IBM), Anish Karmarkar (Oracle), Suresh Kodichath (IONA Technologies), Yves Lafon (W3C), Michael Mahan (Nokia), Noah Mendelsohn (IBM, formerly of Lotus Development), Jeff Mischkinsky (Oracle), Jean-Jacques Moreau (Canon), Mark Nottingham (BEA Systems, formerly of Akamai Technologies), David Orchard (BEA Systems, formerly of Jamcracker), Herve Ruellan (Canon), Jeff Schlimmer (Microsoft Corporation), Pete Wenzel (SeeBeyond), Volker Wiechers (SAP AG).

Previous participants were: Yasser alSafadi (Philips Research), Bill Anderson (Xerox), Vidur Apparao (Netscape), Camilo Arbelaez (webMethods), Mark Baker (Idokorro Mobile, Inc., formerly of Sun Microsystems), Philippe Bedu (EDF (Electricite De France)), Olivier Boudeville (EDF (Electricite De France)), Carine Bournez (W3C), Don Box (Microsoft Corporation, formerly of DevelopMentor), Tom Breuel (Xerox), Dick Brooks (Group 8760), Winston Bumpus (Novell, Inc.), David Burdett (Commerce One), Charles Campbell (Informix Software), Alex Ceponkus (Bowstreet), Michael Champion (Software AG), David Chappell (Sonic Software), Miles Chaston (Epicentric), David Clay (Oracle), David Cleary (Progress Software), Dave Cleary (webMethods), Ugo Corda (Xerox), Paul Cotton (Microsoft Corporation), Fransisco Cubera (IBM), Jim d'Augustine (Excelon Corporation), Ron Daniel (Interwoven), Glen Daniels (Macromedia), Doug Davis (IBM), Ray Denenberg (Library of Congress), Paul Denning (MITRE Corporation), Frank DeRose (TIBCO Software, Inc.), Mike Dierken (DataChannel), Andrew Eisenberg (Progress Software), Brian Eisenberg (DataChannel), Colleen Evans (Sonic Software), John Evdemon (XMLSolutions), David Ezell (Hewlett Packard), James Falek (TIBCO Software, Inc.), Eric Fedok (Active Data Exchange), Chris Ferris (Sun Microsystems), Daniela Florescu (Propel), Dan Frantz (BEA Systems), Michael Freeman (Engenia Software), Dietmar Gaertner (Software AG), Scott Golubock (Epicentric), Mike Greenberg (IONA Technologies), Rich Greenfield (Library of Congress), Hugo Haas (W3C), Mark Hale (Interwoven), Randy Hall (Intel), Bjoern Heckel (Epicentric), Frederick Hirsch (Zolera Systems), Erin Hoffmann (Tradia Inc.), Steve Hole (MessagingDirect Ltd.), Mary Holstege (Calico Commerce), Jim Hughes (Fujitsu Limited), Oisin Hurley (IONA Technologies), Yin-Leng Husband (Hewlett Packard, formerly of Compaq), Ryuji Inoue (Matsushita Electric Industrial Co., Ltd.), Scott Isaacson (Novell, Inc.), Kazunori Iwasa (Fujitsu Limited), Murali Janakiraman (Rogue Wave), Mario Jeckle (DaimlerChrysler Research and Technology), Eric Jenkins (Engenia Software), Mark Jones (AT&T), Jay Kasi (Commerce One), Jeffrey Kay (Engenia Software), Richard Koo (Vitria Technology Inc.), Jacek Kopecky (Systinet), Alan Kropp (Epicentric), Julian Kumar (Epicentric), Peter Lecuyer (Progress Software), Tony Lee (Vitria Technology Inc.), Michah Lerner (AT&T),

Bob Lojek (Intalio Inc.), Henry Lowe (OMG), Brad Lund (Intel), Matthew MacKenzie (XMLGlobal Technologies), Murray Maloney (Commerce One), Richard Martin (Active Data Exchange), Alex Milowski (Lexica), Kevin Mitchell (XMLSolutions), Nilo Mitra (Ericsson), Ed Mooney (Sun Microsystems), Dean Moses (Epicentric), Highland Mary Mountain (Intel), Don Mullen (TIBCO Software, Inc.), Rekha Nagarajan (Calico Commerce), Raj Nair (Cisco Systems), Masahiko Narita (Fujitsu Limited), Mark Needleman (Data Research Associates), Art Nevarez (Novell, Inc.), Eric Newcomer (IONA Technologies), Henrik Nielsen (Microsoft Corporation), Conleth O'Connell (Vignette), Kevin Perkins (Compaq), Jags Ramnaryan (BEA Systems), Andreas Riegg (DaimlerChrysler Research and Technology), Vilhelm Rosenqvist (NCR), Marwan Sabbouh (MITRE Corporation), Waqar Sadiq (Vitria Technology Inc.), Rich Salz (Zolera Systems), Krishna Sankar (Cisco Systems), George Scott (Tradia Inc.), Shane Sesta (Active Data Exchange), Lew Shannon (NCR), John-Paul Sicotte (MessagingDirect Ltd.), Miroslav Simek (Systinet), Simeon Simeonov (Macromedia), Aaron Skonnard (DevelopMentor), Nick Smilonich (Unisys), Seumas Soltysik (IONA Technologies), Soumitro Tagore (Informix Software), James Tauber (Bowstreet), Anne Thomas Manes (Sun Microsystems), Lynne Thompson (Unisys), Patrick Thompson (Rogue Wave), Jim Trezzo (Oracle), Asir Vedamuthu (webMethods), Randy Waldrop (WebMethods), Fred Waskiewicz (OMG), David Webber (XMLGlobal Technologies), Ray Whitmer (Netscape), Stuart Williams (Hewlett Packard), Yan Xu (DataChannel), Amr Yassin (Philips Research), Susan Yee (Active Data Exchange), Jin Yu (MartSoft Corp.).

The people who have contributed to discussions on xml-dist-app@w3.org are also gratefully acknowledged.