XML Protocol (XMLP) Requirements

W3C Working Draft 19 March 2001

This version:
http://www.w3.org/TR/2001/WD-xmlp-reqs-20010319/
Latest version:
http://www.w3.org/TR/xmlp-reqs/
Previous version:
http://www.w3.org/TR/2000/WD-xp-reqs-20001219/
Editors:

Vidur Apparao, Netscape, vidur@netscape.com
Alex Ceponkus, Bowstreet, aceponkus@bowstreet.com
Paul Cotton, Microsoft, pcotton@microsoft.com
David Ezell, Hewlett Packard, david_e3@verifone.com
David Fallside, IBM, fallside@us.ibm.com
Martin Gudgin, DevelopMentor, marting@develop.com
Oisin Hurley, IONA Technologies, ohurley@iona.com
John Ibbotson, IBM, ibbotson@uk.ibm.com
R. Alexander Milowski, Lexica, LLC, alex@milowski.com
Kevin Mitchell, XMLSolutions, mailto:kevin.mitchell@xmls.com
Jean-Jacques Moreau, Canon, moreau@crf.canon.fr
Eric Newcomer, IONA Technologies, eric.newcomer@iona.com
Henrik Frystyk Nielsen, Microsoft, frystyk@microsoft.com
Mark Nottingham, Akamai Technologies, mnot@akamai.com
Waqar Sadiq, Vitria Technology Inc., wsadiq@vitria.com
Stuart Williams, Hewlett Packard, skw@hplb.hpl.hp.com
Amr Yassin, Philips Research, amr.f.yassin@philips.com

Abstract

This document describes the XML Protocol Working Group's requirements for the XML Protocol (XMLP) specification.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This is the second W3C Working Draft of the XML Protocol requirements document. It is a chartered

deliverable of the XML Protocol Working Group (WG), which is part of the XML Protocol Activity. The Working Group has agreed to publish this document, although this document does not necessarily represent consensus within the Working Group about XMLP requirements. This new version contains an updated glossary, some new requirements and usage scenarios.

Discussion of this document takes place on the public <xml-dist-app@w3.org> mailing list (Archives) per the email communication rules in the XML Protocol Working Group Charter.

This is a public W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". A list of all W3C technical reports can be found at http://www.w3.org/TR/.

Table of Contents

---

1 Notations

The following terminology and typographical conventions have been used in this document.

Each requirement and scenario has a three digit number with a prefix indicating the status as follows:

- A "**DR**nnn" notation indicates a requirement that the WG is actively considering (**has not** reached *rough* consensus within the WG)
- An "**R**nnn" notation indicates a requirement that the WG is not actively considering at present (**has** reached *rough* consensus within the WG)
- A "**DS**nnn" notation indicates a usage scenario that the WG is actively considering (**has not** reached *rough* consensus within the WG)

- An "**S**nnn" notation indicates a usage scenario that the WG is not actively considering at present (**has** reached *rough* consensus within the WG)

The numbers used to identify requirements are arbitrary and does not imply any ordering or significance.

The document includes several verbatim quotes from the XML Protocol WG Charter which provide context for the requirements. The quoted text is *emphasized* and prefixed with "*Charter*".

Editorial notes are indicated with yellow background (may not appear in all media) and prefixed with "Ednote".

2 Relationship to WG Charter

The XML Protocol WG Charter has two sections describing what is in-scope and what is out-of-scope of the problem space defined for the WG. The WG considers all the requirements in section 4 to be in-scope per the Charter.

Reviewers and readers should be familiar with the XML Protocol WG Charter because it provides the critical context for the requirements and any discussion of them.

3 Requirements on Requirements

R900

The XMLP requirements must include usage scenarios that describe how XMLP is used in various environments (see section 8). The set of usage scenarios must represent the expected range of XMLP's use. The scenarios must be used as design cases during the development of XML Protocol, and it must be possible to determine whether or not the XML Protocol design enables each scenario. In addition, the usage scenarios are intended to help a technically competent person understand the role of XMLP.

4 Requirements

4.1 General Requirements

Charter: The envelope and the serialization mechanisms developed by the Working Group may not preclude any programming model nor assume any particular mode of communication between peers.

R500

The specification will make reasonable efforts to support (but not define) a broad range of programming models suitable for the applications intended for XMLP.

R501

The specification will make reasonable efforts to support (but not define) a broad range of protocol bindings between communicating peers (see also section 4.6).

R502

The specification developed by the Working Group must support either directly or via well

defined extension mechanisms different messaging patterns and scenarios. The specification will directly support One-way and Request-response patterns as part of permanently and intermittently connected scenarios. The specification will not preclude the development of other patterns at either the application or transport layers. Examples of such patterns may include publish-subscribe or multicast delivery. All patterns and scenarios will be described by relevant usage scenarios (see section 8).

R503

The Working Group will coordinate with the W3C XML Activity through the XML Coordination Group (W3C members only) and shall use available XML technologies whenever possible. If there are cases where this is not possible, the reasons must be documented thoroughly.

R504

The specification developed by the Working Group shall be as lightweight as possible keeping parts that are mandatory to the minimum. Optional parts of the specification should be orthogonal to each other allowing non-conflicting configurations to be implemented.

R505

The specification must be suitable for use between communicating parties that do *not* have *a priori* knowledge of each other.

R506

The specification must focus on the encapsulation and representation of data being transferred between parties capable of generating and/or accepting an XMLP envelope.

4.2 Simplicity and Stability

Charter: Focus must be put on simplicity and modularity and must support the kind of extensibility actually seen on the Web. In particular, it must support distributed extensibility where the communicating parties do not have a priori knowledge of each other.

Charter: Simplicity is a key element in making distributed systems easy to understand, implement, maintain, and evolve. Modularity and layering are two important design principles for achieving simplicity. Although simplicity can only be measured in relative terms, the Working Group must ensure that the complexity of any solution produced is comparable to that of other current and widespread Web solutions.

Charter: Another important aspect of simplicity is ease of deployment. The Working Group will look at various ways of deploying XML Protocol in a manner that is compatible with the existing Web infrastructure.

Over the years, many different companies and individuals have proven the ability to design and implement workable open protocols for distributed computing that operate largely within organizational boundaries. The design center for XMLP must include the interoperation of systems across organizational boundaries. The aim is to exploit Web philosophy and Web design principles in order to help foster widespread decentralized computing on the Web.

R307

XMLP must be suitable for widespread use across organizational boundaries in support of the

application usage scenarios supplied elsewhere in this document (see section 8). This suitability requirement implies simplicity in the language of the XMLP specification, which itself describes a technology that is simple to understand and to implement correctly (see also R301, R301a). Although simplicity can only be measured in relative terms, the Working Group should ensure that the complexity of any solution produced is comparable to that of other current and widespread Web solutions.

R308

Since XMLP is intended to be a foundation protocol, its definition should remain simple and stable over time. Explicit use of modularity and layering in the resulting design will help assure longevity. Such a framework will allow subsequent extension of the design while leaving the foundation of the design intact. (R300 and R302 relate to stability).

Requirements for simplicity and stability arise in the context of the specification documents and in the context of the protocol technologies being defined.

Simplicity in XMLP implies that many potentially important features are out of scope for XMLP proper. However, the XML Protocol Working Group recognizes that providing consistent ways to support these out of scope features will help keep XMLP stable.

Examples of such features are 1) message authentication and encryption (perhaps using SMIME, SSL, or digital signatures), 2) sessions and transactions (possibly by providing globally unique identifiers for messages), and 3) service definition and discovery. Facilities to support features like these may resemble SOAP/1.1 facilities such as the "Header" element.

4.2.1 The XMLP Specification Documents

R300 (absorbs old DRs: DR023, DR053, DR088)

The requirements that XMLP support the use of layering and be modular, extensible, and transport independent imply that there is an architectural design model behind XMLP. This architecture and the extensibility framework must be explicitly defined (R308 references modularity, R302 and R700 series reference extensibility, R502 and R600 reference transport neutrality).

In this context, layering refers to both XMLP's support of XMLP modules (the layer(s) "above") as well as the capability of XMLP to define services required (the layer(s) "below") for implementation across a variety of underlying protocols

R301

The XMLP specifications should be clear and easy to understand. This clarity implies that considerable editorial effort will be required in the structuring of the narrative through both outline/overview and normative reference material.

R301a

The XMLP specification must clearly identify conformance requirements in a way that enables the conformance of an implementation of the specification to be tested (see also the W3C Conformance requirements (W3C members only)).

4.2.2 The XMLP Technologies

R302 (Absorbs old DR's: DR107)

XMLP must support extensibility of vocabulary between communicating parties in a way that allows for decentralized extensibility without prior agreement. The WG must demonstrate through usage scenarios that the solution supports decentralized extensibility in a modular and layered manner (see section 8).

To date the web has been enormously successful because it has enabled the creators of web services adapt the user interfaces they provide to human users of the web. A goal of XMLP is to achieve similar levels of evolvability, extensibility and adaptability for interfaces between web services.

R304

XMLP should facilitate the creation of simple applications. Simple applications are often characterized by message exchange patterns such as one-way (or event), and two-way (or synchronous) request response interactions. The specification should make such simple exchange applications as easy as possible to create and to use.

R306 (Absorbs old DRs: DR090)

XMLP and applications of XMLP must be easy to deploy—especially in systems already supporting XML technologies like XML namespaces and XML schemas.

The ease with which XMLP applications can be deployed will be crucial to the success of XMLP. The design of the protocol architecture must be sensitive to the issues arising in the full spectrum of deployment environments ranging from resource constrained embedded devices (appliances) through high performance service engines.

R309

XMLP should support applications that operate on resource constrained devices.

4.3 Data Encapsulation and Evolvability

Charter: For two peers to communicate in a distributed environment, they must first agree on a unit of communication. The XML Protocol Working Group must define such a unit by defining an encapsulation language that allows for applications to independently introduce extensions and new features. In this context, the following requirements for extensions and features must be met:

- *They are or can be orthogonal to other extensions.*
- *They can be deployed automatically and dynamically across the Web with no prior coordination and no central authority.*
- *The sender can require that the recipient either obeys the semantics defined by an extension or aborts the processing of the message.*

R701a Requirement for Encapsulation

The XMLP specification must define the concept of an envelope or outermost syntactical construct or structure within which all other syntactical elements of the message must be enclosed. The envelope must be described with XML Schema.

R701b Requirement for Encapsulation

The XMLP specification must also define a processing model that defines what it means to properly process an XMLP envelope or produce a fault. This processing model must be independent of any extensions carried within the envelope. The processing model must apply equally to intermediaries as well as ultimate destinations of an XMLP envelope.

R700a Requirement for Extensibility

The XMLP specification must define a mechanism or mechanisms that allow applications to submit application-specific content or information for delivery by XMLP. In forming the standard for the mechanisms, the XMLP specification may consider support for:

- carrying application specific payloads inside the XMLP envelope,
- referring to application specific payloads outside the XMLP envelope,
- carrying nested XMLP envelopes as application specific data within the XMLP envelope,
- referring to XMLP envelopes as application specific data outside the XMLP envelope

Regarding the handling of binary data in particular, the XML Protocol WG Charter has the following to say:

Charter: Note that XML Namespaces provide a flexible and lightweight mechanism for handling language mixing as long as those languages are expressed in XML. In contrast, there is only very rudimentary support (base-64 encodings etc.) for including data languages expressed in binary formats. Such formats include commonly used image formats like PNG, JPEG etc. Although it is inconceivable to imagine a Web without such data formats, it is not considered a priority of this Working Group to solve this problem. This is in part because other organizations (e.g. ebXML and RosettaNet) are already addressing the issue using an approach based on MIME multipart. The Working Group can consider solutions proposed by other groups as a matter of low priority, if there is sufficient interest.

R700b Requirement for Extensibility

To manage the mechanisms, the XMLP specification must define a set of directives which will unambiguously indicate to an XMLP processor which extensions are optional and which are mandatory so that it can:

- process all of the extensions in an XMLP envelope or fail,
- process a subset of the extensions in an XMLP envelope or fail.

R700c Requirement for Extensibility

In both cases above, the XMLP processor must fail in a standard and predictable fashion.

R702 Requirement for Evolution

The XMLP specification must define the concept of protocol evolution and define a mechanism or mechanisms for identifying XMLP revisions. This mechanism or mechanisms must ensure that an XMLP processor, by simple inspection of an XMLP envelope, may determine whether or not the envelope is compatible with its processing ability. The specification must define the concepts of backwards compatible and backwards incompatible evolution.

R703a Requirement for Encapsulation of Error Information

The XMLP specification must define a means to convey error information as a fault. The capability of XMLP carrying a fault message must not depend on any particular protocol

binding.

R703b Requirement for Encapsulation of Status

The XMLP specification must define a mechanism or mechanisms to allow the transfer of status information within an XMLP message without resort to use of XMLP fault messages or dependence on any particular interaction model.

4.4 Intermediaries

Charter: Intermediaries are essential parts of building distributed systems that scale to the Web. Intermediaries can act in different capacities ranging from proxies, caches, store-and-forward hops, to gateways. Experience from HTTP and other protocols has shown that intermediaries cannot be implicitly defined but must be an explicit part of the message path model for any data encapsulation language. Therefore, the Working Group must ensure that the data encapsulation language supports composability both in the vertical (within a peer) as well as in the horizontal (between peers).

Because XMLP separates the message envelope from the transport binding, two types of intermediaries are possible; transport intermediaries and processing intermediaries.

4.4.1 Transport Intermediaries

Transport intermediaries are interposed by a transport binding, as part of the message exchange pattern that it implies. They do not define a processing model for messages; they only operate as part of the transport binding, as a message routing mechanism and cannot be addressed from within an XMLP envelope.

R803

XMLP must not preclude the use of transport bindings that define transport intermediary roles such as store-and-forward, proxy and gateway.

4.4.2 Processing Intermediaries

Processing intermediaries are full XMLP processors; they process the message, but are not the ultimate recipient of it. They may be colocated with transport intermediaries, using them as a routing mechanism, or they may use in-message routing mechanisms.

R811

XMLP must define and accommodate processing intermediaries.

To enable the interposition of processing intermediaries into the message path, two core requirements must be met:

R806

Targeting - XMLP must define mechanisms that allow XMLP processors, including intermediaries, to identify XMLP blocks which they are eligible to process.

R808

Reporting - XMLP must enable the generation of status and/or error messages by processing

intermediaries, and enable propagation and proper identification of status and/or error messages through processing intermediaries.

In addition

R802

XMLP must also enable processing intermediaries to locate and process XMLP blocks intended for them without processing the entire message.

4.5 Data Representation

Charter: With the introduction of XML and Resource Description Framework (RDF) schema languages, and the existing capabilities of object and type modeling languages such as Unified Modeling Language (UML), applications can model data at either a syntactic or a more abstract level. In order to propagate these data models in a distributed environment, it is required that data conforming to a syntactic schema can be transported directly, and that data conforming to an abstract schema can be converted to and from XML for transport.

Charter: The Working Group should propose a mechanism for serializing data representing non-syntactic data models in a manner that maximizes the interoperability of independently developed Web applications. Furthermore, as data models change, the serialization of such data models may also change. Therefore it is important that the data encapsulation and data representation mechanisms are designed to be orthogonal.

Charter: Examples of relationships that will have to be serialized include subordinate relationships known from attachments and manifests. Any general mechanism produced by the Working Group for serializing data models must also be able to support this particular case.

R400

The XMLP data encapsulation and data representation mechanisms must be orthogonal.

R401

The XMLP data representation must support using XML Schema simple and complex types.

R402

The XMLP data representation must be able to serialize data based on data models not directly representable by XML Schema simple and complex types. These data models include object graphs and directed labeled graphs. It must be possible to reconstruct the original data from the data representation.

R403

Data serialized according to the XMLP data representation may contain references to data outside the serialization. These references must be Uniform Resource Identifiers (URIs).

R404

The XMLP data representation must be able to encode arrays which may be nested.

4.6 Protocol Bindings

Charter: A mechanism for using HTTP transport in the context of an XML Protocol. This does not mean that HTTP is the only transport mechanism that can be used for the technologies developed, nor that support for HTTP transport is mandatory. This component merely addresses the fact that HTTP transport is expected to be widely used, and so should be addressed by this Working Group.

Charter: Mapping onto existing application layer protocols may lead to scalability problems, security problems and semantic complications when the application semantics defined by those protocols interfere with the semantics defined by an XML Protocol. The WG may consider issuing a warning about the possible problems of reusing non-safe "transports" like SMTP and others. A mapping onto transport services other than HTTP will only be started if enough interest is shown and time is available.

Charter: General transport issues were investigated by the HTTP-NG Activity, which designed a general transport mechanism for handling out-of-order delivery of message streams between two peers. While we do strongly encourage work to be undertaken in this area, it is expected that work in this area will be done in collaboration with the IETF and not as part of this Working Group

R600

The XMLP specification must not mandate any dependency on specific features or mechanisms provided by a particular transport protocol beyond the basic requirement that the transport protocol must have the ability to deliver the XMLP envelope as a whole unit. This requirement does not preclude a mapping or binding to a transport protocol taking advantages of such features. It is intended to ensure that the basic XMLP specification will be transport neutral.

R604

The XMLP specification must consider the scenario where an XMLP message may be routed over possibly many different transport or application protocols as it moves between intermediaries on the message path. This requirement implies it must be possible to apply many transport or application protocol bindings to the XMLP message without information loss from the XMLP message content.

R608

The XMLP binding mechanism should not preclude the possibility of constructing bindings to protocols that provide a security mechanism.

Typical examples of such protocols are SSL providing a secure channel,and S/MIME which provides a secure wrapper. It should be possible to specify XMLP bindings for such security protocols.

R609

The XMLP specification may mandate the use of a specific character encoding, such as UTF-8, at some point in the future.

The Working Group is aware of the complexity resulting in the use of a large set of character encodings and is actively seeking feedback in this area. Until all the feedback has been evaluated, the Working Group will not make a decision in favor of restriction.

R612

The XMLP specification must provide a normative description of the default binding of XMLP to HTTP. This binding, while normative, is not to be exclusive. The binding provided by the Working Group will respect the semantics of HTTP and will demonstrate that it can co-exist with existing HTTP/1.0 and HTTP/1.1 implementations.

4.7 Convention for RPC

Charter: A convention for the content of the envelope when used for RPC (Remote Procedure Call) applications. The protocol aspects of this should be coordinated closely with the IETF and make an effort to leverage any work they are doing

R200

XMLP must contain a convention for representing calls and replies between RPC (Remote Procedure Call) applications and services. The conventions must include the following:

1. Complete and unique identification, by means of URI syntax, of the program, service or object and procedure or method to be called.
2. Enable support for matching response messages to request messages for cases in which matching is not provided by the underlying protocol binding.
3. The ability to specify the parameters to a call in a request message and the results of a call in a reply messages.
4. Provisions for specifying errors in a reply message (see also 703a and 703b)

Where possible, an attempt will be made to leverage any related work done by the IETF.

R201

The RPC conventions within XMLP should use the Data Representation model discussed in section 4.5 to represent parameters to a call in the request message and results of the call in the reply message. It must be convenient to create straightforward mappings of the data types to a wide variety of widely deployed programming languages and object systems.

R202

XMLP should allow applications to include custom encodings for data types used for parameters and results in RPC messages.

5. Requirements from other W3C WGs

These are requirements submitted by other W3C Working Groups and Activities.

Ednote: These are the verbatim received texts. The WG has not made any decisions regarding these requirements.

5.1 XForms Requirements

These are the requirements that the XML Protocol WG has received from the (W3C Members only) XForms WG:

XForms models the data to be obtained from the user, specifies how a user interface for obtaining the data is declared using XHTML markup, and finally specifies how the populated data is shipped backed to the server. The [SEND] subgroup is responsible for the interactions between the XForms client and the backend server.

The work on [SEND] could be a replacement for the various methods for posting data to an HTTP server such as application/x-www-form-urlencoded or multipart/form-data.

Requirements:

1. An XForms client needs to send and receive well-formed XML data that has been defined through the XForms specification. For example, XML data will be "sent" when the user agent is done filling out an XForm or XML data will be "received" when a server ships out initial values for populating a form.
2. An XForms client needs to send/receive partially completed XML data to/from the server for persistence. This functionality will allow a user agent to "save" or "load" a form in progress. Therefore, the XML data may not fully conform to a schema when only partially completed.
3. An XForms client needs to be able to send/receive arbitrary binary content along with the XML data. This will be used to support features such as the "file upload" feature available in many WWW browsers. There needs to be support for both 'in-band' (i.e. the binary data is within the XML data in an XML compatible encoding such as base64) and 'out-of-band' data (i.e. the binary data is available at some other location, and the XML data refers to the other location).

5.2 P3P Requirements

These are the requirements that the XML Protocol WG has received from the P3P WG:

- It must be possible to associate a P3P Privacy Policy with an XMLP message.

6 Glossary

For a description of fundamental Web concepts including resources and resource manifestations, see the "Web Characterization Terminology & Definitions Sheet" W3C Working Draft.

6.1 Protocol Layering Concepts

XMLP is a framework which can accommodate an open-ended set of XMLP modules defining a large variety of functions and services. Typical functions and services defined by XMLP modules can range from generic mechanisms for handling security, caching, routing, and eventing to specific functions like submitting a purchase order.

While XMLP itself is intended to be as simple and lightweight as possible, XMLP modules can be designed and composed to perform arbitrarily complex operations allowing the core protocol to remain simple.

XMLP itself can be layered on top of a variety of underlying protocols that can help facilitate the transfer of XMLP messages.

XMLP

The formal set of conventions governing the format and processing rules of an XMLP message and basic control of interaction among applications generating and accepting XMLP messages

for the purpose of exchanging information along an XMLP message path.

XMLP block

The syntactic construct or structure defined in an XMLP module. XMLP blocks are processed by XMLP handlers.

XMLP handler

An XMLP handler is responsible for processing XMLP Blocks targeted at it according to any rules defined in the corresponding XMLP module.

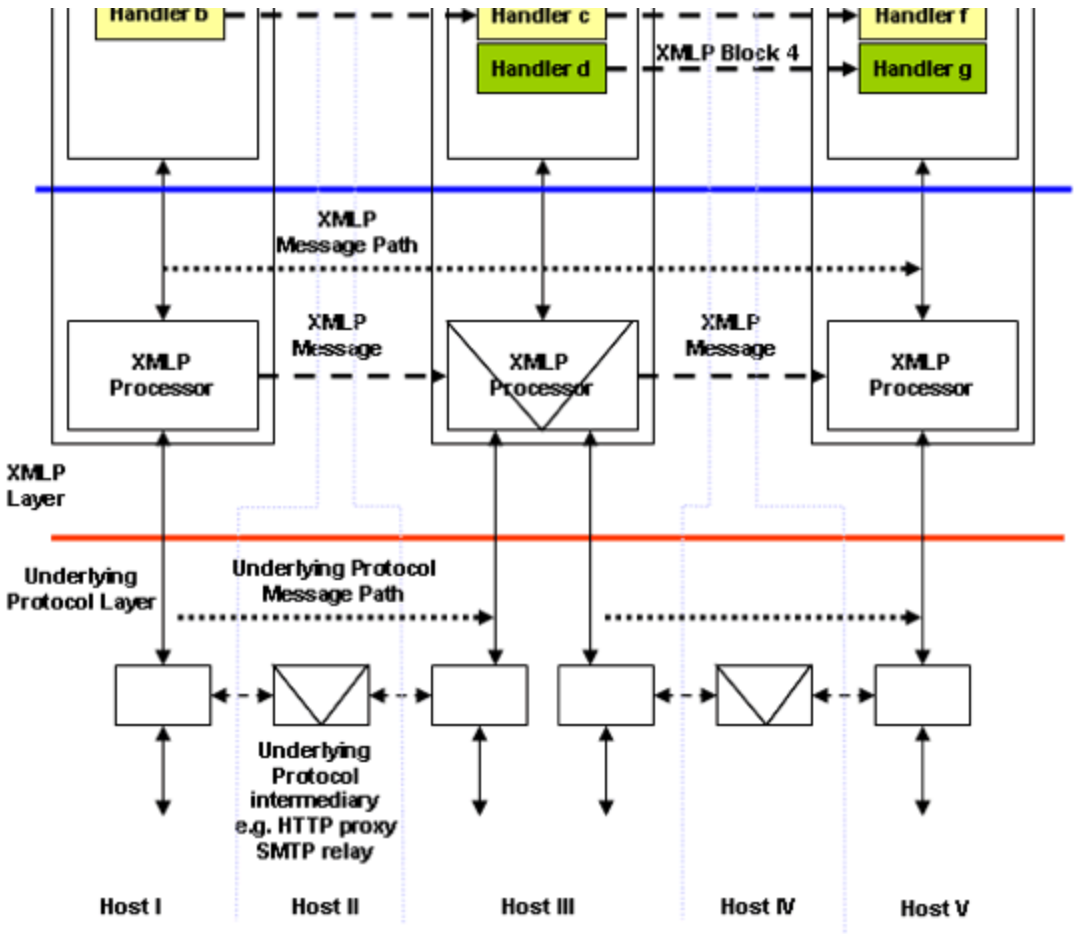
XMLP module

An XMLP module is a basic unit for the definition of extensions to XMLP. An XMLP module encapsulates the definition of one or more related XMLP blocks and their associated processing rules. These processing rules are realised in one or more XMLP handlers.

XMLP binding

The formal set of rules for carrying an XMLP message within or on top of another protocol for the purpose of transmission. Typical XMLP bindings include carrying an XMLP message within an HTTP message, or on top of TCP.

The XMLP layering model is illustrated on Figure 1.



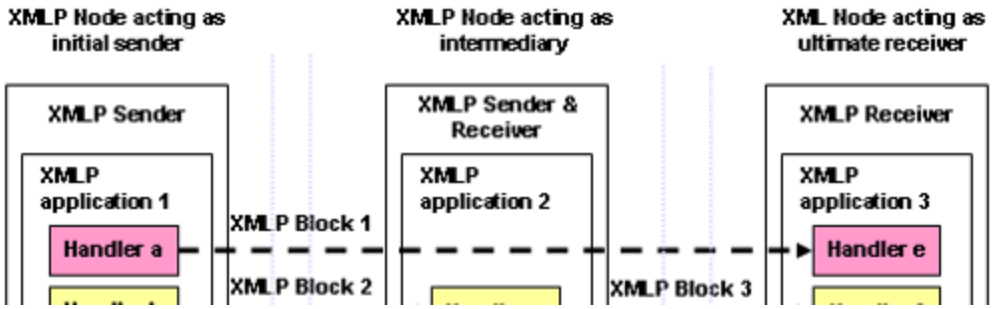


Figure 1: XMLP Layering Model

6.2 Data Encapsulation Concepts

The XMLP data encapsulation model describes how XMLP blocks defined by XMLP modules can be carried within an XMLP message, which is the fundamental unit of communication in XMLP. The following diagram illustrates how an XMLP message is composed.

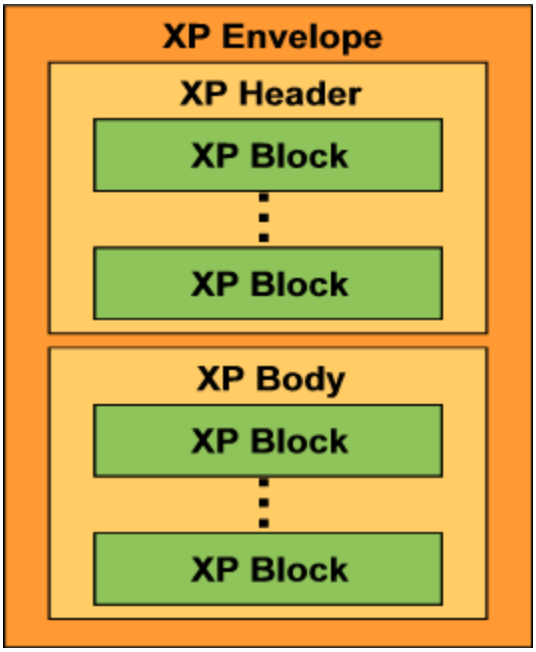


Figure 2: Encapsulation model illustrating the parts of an XMLP message

An XMLP message is composed of an XMLP envelope which contains an XMLP header and an XMLP body each of which can contain zero, one or more XMLP blocks.

XMLP message

An XMLP message is the basic unit of communication between peer XMLP processors.

XMLP processor

An XMLP Processor processes an XMLP message according to the formal set of conventions defined by XMLP. It is responsible for enforcing the rules that govern the exchange of XMLP messages and accesses the services provided by the underlying protocols through XMLP bindings. An XMLP processor is responsible for invoking local XMLP Handlers and providing the

services of the XMLP layer to those XMLP handlers.

Non-compliance with XMLP conventions or failure in an XMLP handler can cause an XMLP processor to generate an XMLP fault (see also XMLP receiver and XMLP sender).

XMLP envelope

The outermost syntactical construct or structure of an XMLP message defined by XMLP within which all other syntactical elements of the message are enclosed.

XMLP header

A collection or zero or more XMLP blocks which may be targeted at any XMLP receiver within the XMLP message path

XMLP body

A collection or zero, or more XMLP blocks targeted at the ultimate XMLP receiver within the XMLP message path.

XMLP fault

A special XMLP block which contains fault information generated by an XMLP processor or handler.

6.3 Message Sender and Receiver Concepts

The XMLP message path model is defined in terms of XMLP senders and XMLP receivers who can generate and accept XMLP messages respectively (see Figure 1). Behind each XMLP receiver is an XMLP processor that processes the message according to the rules of XMLP.

A important part of the XMLP message path model is the concept of XMLP intermediaries. Intermediaries contain both an XMLP receiver and an XMLP sender which allows them to forward a message on behalf of the previous sender.

Note: In some interactions, more complicated message path models may be required to encapsulate the semantics of multi-party interactions like for example "fan-out" or "fan-in" models. Such models can be built using the basic XMLP message path model provided that the semantics of message "split" and "merge" are provided by higher layer semantics.

XMLP node

An XMLP Node is an encapsulation of XMLP handlers and their associated XMLP processor.

XMLP sender

An XMLP Sender is an XMLP Node that transmits an XMLP Message.

XMLP receiver

An XMLP Receiver is an XMLP Node that accepts an XMLP Message.

XMLP message path

The set of XMLP senders and XMLP receivers through which a single XMLP message passes. This includes the initial XMLP sender, zero or more XMLP intermediaries, and the ultimate XMLP receiver.

initial XMLP sender

The XMLP sender that originates an XMLP message as the starting point of an XMLP message path.

XMLP intermediary

An XMLP intermediary is both an XMLP receiver and an XMLP sender, target-able from within an XMLP message. It processes a defined set of blocks in an XMLP message along an XMLP message path. It acts in order to forward the XMLP message towards the ultimate XMLP receiver.

ultimate XMLP receiver

The XMLP receiver that the initial sender specifies as the final destination of the XMLP message within an XMLP message path. An XMLP message may not reach the ultimate recipient because of an XMLP fault generated by an XMLP processor or an XMLP Handler along the XMLP message path.

The relationship between an XMLP sender and an XMLP processor and an XMLP receiver and an XMLP processor respectively is illustrated in Figure 1.

6.4 Data Representation Concepts

XMLP data model

A set of abstract constructs that can be used to describe common data types and link relationships in data defined by XMLP modules.

XMLP data encoding

The syntactic representation of data described by the XMLP data model within one or more XMLP blocks in an XMLP message.

7 Other Terms

Ednote: A list of commonly used terms that are not defined by the WG.

8 Usage Scenarios

Usage scenarios are intended to provide representative examples of situations where XMLP might be applicable. The purpose of usage scenarios is to help ensure that XMLP is capable of dealing with applications and services actually seen in the Web. Hence, usage scenario specifications should be at a coarse-grain level of an end user's desired XML document/message interchange, rather than at a detailed, implementation or transport specific level. Usage scenarios often make assumptions about the specific environments in which the use cases are described that the requirements cannot.

In other words, the requirements are explicitly targeted to the design of XMLP; usage scenarios are targeted to systems in which XMLP is most likely part of an overall solution. Not all requirements need to be referenced by an example usage scenario, since, in addition to higher-level, application specific requirements for use, there are internal, architectural requirements independent of any specific higher-level use (e.g., using XML, schemas, and namespaces imposes certain requirements irrespective of use).

S1 Fire-and-forget to single receiver

A sender wishes to send an unacknowledged message to a single receiver (e.g. send a stock price update every 15 minutes)

Note: S1 Originates from splitting the ebXML use case 1.1 into 2 scenarios (S1 and S2).

S2 Fire-and-forget to multiple receivers

A sender wishes to send unacknowledged messages to a set of receivers (e.g. send a stock price update every 15 minutes)

Note: S2 Originates from splitting the ebXML use case 1.1 into 2 scenarios (S1 and S2). Note that S2 may be decomposed into Multiple instances of S1 under the control of some ?higher level? process such as multicast or publish/subscribe.

S3 Request-response

Two parties wish to conduct electronic business by the exchange of business documents. The sending party packages one or more documents into a request message which is then sent to the receiving party. The receiving party then processes the message contents and responds to the sending party. Examples of the sending party's documents may be purchase order requests, manufacturing information and patient healthcase information. Examples of the receiving party's responses may include order confirmations, change control information and contractual acknowledgements.

S4 Remote Procedure Call (RPC)

The sender invokes the service by passing parameters that are serialised into a message for transmission to the receiving server.

S5 Request with acknowledgement

A sender wishes to reliably exchange data with a receiver. It wishes to be notified of the status of the data delivery to the receiver. The status may take the form of:

1. The data has been successfully delivered to the receiver, or
2. Some failure has occurred which prevents the sucessful delivery to the receiver.

Note: This scenario does not imply that reliable message delivery will be supported by the XMLP core specification.

S6 Request with encrypted payload

A sender wishes to exchange data with a receiver and has agreed to encrypt the payload. The

sending and receiving applications agree on the encryption methodology. Data is encrypted by the originating application and sent to the receiver via XMLP.The data reaches the receiving application untouched, and may then be decrypted in the agreed-upon manner.

S7 Third part intermediary

A blind auction marketplace serves as a broker between buyers and suppliers. Buyers submit their requirements to the marketplace hub, which broadcasts this information to multiple suppliers. Suppliers respond to the marketplace hub where the information is logged and ultimately delivered to the buyer.

S8 Conversational message exchange

Two partners are engaged in a long-running process which involves multiple message exchanges. Examples of such processes may be complex supply chain management, dynamic manufacturing scheduling or information retrieval. There may be multiple instances of the same process in progress between the same two partners.

S10 Message header and payload encryption

Two trading partners engaged in a message exchange may agree to cryptographically sign and verify either the message header, the routing header(s) and/ or the payload. The sender or originating application may perform the siging of the payload. The sending message handler signs the message header. A routing header may be appended to the message header. The routing header may also be signed by a message service handler.

S11 Communication via multiple intermediaries

An intermediary forwards a message to the ultimate receiver on behalf of an initial sender. The initial sender wishes to enforce the non-repudiation property of the route. Any intermediate message service handler that appends a routing message must log the routing header information. Signed routing headers and the message headers must be logged at the message handler which passes the message to the ultimate receiver to provide the evidence of non-repudiation.

DS17 Asynchronous messaging

A sender sends a message asynchronously to a receiver expecting some response at a later time. The sender tags the request with an identifier allowing the response to be correlated with the originating request. The sender may also tag the message with an identifier for another service (other than the originating sender) which will be the recipient of the response.

S19 Sending non-XML data

A digital camera wishes to transmit image data over a wireless link using XMLP to a remote server. The binary image data (non-XML) accompanies the message. The digital camara represents a situation in which connections from the receiver to the sender may not be permitted due to device limitations or firewalls.

S20 Multiple asynchronous responses

An application requests some information from a server, which is returned at a later time in multiple responses. This can be because the requested information was not available all at once (e.g., distributed web searches). (based on mail)

S21 Incremental parsing/processing of XMLP messages

An XMLP sender generates a lengthy XMLP message that is incrementally transmitted and received by an XMLP receiver. The XMLP receiver employs an XMLP handler that can incrementally process the body as it is received (e.g., employing a SAX-style XML parser on the body as it arrives). Note that the entire message need not be present at one time at any point in its existence.

This would be particularly helpful for memory-limited processors. It is also very efficient for services which are consistent with incremental, real-time transformations of the data, direct archiving of received data, etc. It would also be useful in scenarios in which voluminous body data can be directly transduced into application data structures or events by an XMLP (module) processor. In particular, there is no need for the explicit construction of a DOM model of the data. Support for XMLP data models might still be possible even with incremental processing if the models are incrementally constructible (copied in its entirety from mail)

S23 Event notification

An application subscribes to notifications of certain named events from an event source. When such events occur, notifications are sent back to the originating application (first party notification) or to another application (third party notification). For example, an application can subscribe to notification of various aspects of a printer's status (e.g., running out of paper, ink etc.). The notifications of such events could be delivered to a management application (based on: See item 2 of mail)

DS24 Caching

Some applications may wish to make caching possible for latency, bandwidth use or other gains in efficiency. To enable this, it should be possible to assign cacheability in a variety of circumstances. For example, "read" caching might be used to store messages at intermediaries for reuse in the response phase of the request/response message exchange pattern. Such caching might be on the scope of an entire message, an XMLP module, or scoped to individual XMLP module elements.

Similarly, "write" caching may be useful in situations when a request message in a request/response message exchange pattern (as well as similar messages in other message exchange patterns) does not need to be immediately forwarded or responded to. Such cachability might be scoped by different methods, as outlined above.

Cacheability scoped by different elements might be associated by an attribute to the target element, through use of XML Query or XPath to describe the target elements in a header, or implied by the document schema, for example.

Cacheability mechanisms applied to messages, bodies or elements might include time-to-live (delta time), expiry (absolute time), entity validation, temporal validation, subscription to invalidation services, and object update/purge.

Finally, some applications may be capable of describing the dependencies and relationships between message elements. For example, a response element may be applicable to a wide range of requests; it would be beneficial to describe this element's relationship with request elements, so that it may satisfy a wide range of requests in an economical fashion. Similarly, the presence of a particular element may be a trigger for a cacheability mechanism to be applied to another element, such as validation or invalidation (see also mail from archives)

S805 Routing

A developer wishes to force an explicit message path through certain intermediaries - for instance, he might use an anonymizing intermediary to make a call to a specified remote service without allowing the target service to track the identity/IP of the caller. In this case, the intermediary is responsible for calling the target service and returning the results to the caller, using its own authentication credentials if any are required by the target service.

S807 Tracking

A service provider wishes to track incoming messages to see exactly which processing intermediaries have touched it by the time it arrives at its destination. It therefore requires a tracking extension to be included by all clients, and by any processing intermediaries along the message paths from the clients to the server.

S809 Caching with Expiration

BizCo updates their online price catalog every morning at 8AM. Therefore, when remote clients access their XMLP inventory service, clients and intermediaries may cache the results of any price queries until 8AM the next day.

S810 QoS

An XMLP sender (not necessarily the initial XMLP sender) wants the XMLP message to be handled with specific quality of service as it traverses the XMLP message path to include multiple XMLP Processing intermediaries. Information in the XMLP message is used to select appropriate QoS mechanisms (e.g., RSVP, Diffserv, MPLS, etc.). Selection of QoS may be constrained by QoS policies, Service Level Agreements (SLAs), Service Level Specifications (SLS).

A Acknowledgments

The WG thanks all participants of the xml-dist-app@w3.org mailing list (archives) for directly and indirectly contributing to this document.

B References

1. XML Protocol Activity
2. XML Protocol Working Group
3. XML Protocol Working Group Charter
