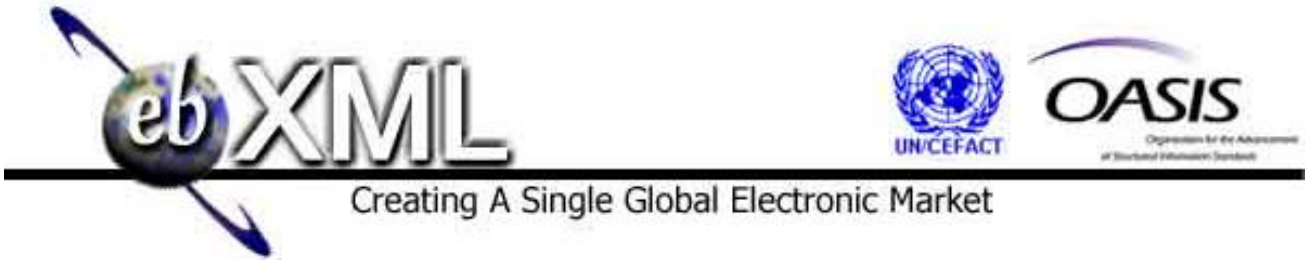


Messaging Service Specification

ebXML Transport, Routing & Packaging

Version 0.21c

26 September 2000

1 Status of this Document

This document specifies an ebXML DRAFT for the eBusiness community.

Distribution of this document is unlimited.

The document formatting is based on the Internet Society's Standard RFC format converted to Microsoft Word 2000 format.

This version:
http://www.ebxml.org/working/project_teams/…

***Latest version*:**
http://www.ebxml.org/…

Previous version:
http://www.ebxml.org/…..

2 ebXML participants

The authors wish to acknowledge the support of the members of the Transport, Routing and Packaging Project Team who contributed ideas to this specification by the group's discussion e-mail list, on conference calls and during face-to-face meetings.

Ralph Berwanger – bTrade.com
Jonathan Borden - Author of XMTP
Jon Bosak – Sun Microsystems
Marc Breissinger - webMethods
Dick Brooks – Group 8760
Doug Bunting - Ariba
David Burdett - Commerce One
Len Callaway – Drummond Group, Inc.
David Craft – VerticalNet
Philippe De Smedt - Viquity
Lawrence Ding - WorldSpan
Rik Drummond - Drummond Group, Inc.
Christopher Ferris – Sun Microsystems
Maryann Hondo - IBM
John Ibbotson - IBM
Ian Jones – British Telecommunications
Ravi Kacker – Kraft Foods
Nick Kassem – Sun Microsystems
Henry Lowe - OMG
Jim McCarthy - webXI
Bob Miller - GSX
Dale Moberg - Sterling Commerce
Joel Munter – Intel
Farrukh Najmi – Sun Microsystems
Kathy Spector – Extricity
Nikola Stojanovic
Gordon Van Huizen – Process Software
Martha Warfelt - DaimlerChrysler
Prasad Yendluri – Vitria
Jim Hughes - Fujitsu
Masayoshi Shirmamura – Fujitsu
Akira Ochi – Fujitsu

3 Table of Contents

4 Introduction

4.1 Summary of Contents of Document

This specification defines an that describes how to securely and reliably exchange messages between two parties. It includes descriptions of:

- the *ebXML Message* structure used to encapsulate (package) *ebXML Message* payloads for transport between parties, and
- the behavior of the messaging service that sends or receives those messages.

No assumption or dependency is made relative to communication protocol or type of payload. The specifications contained here are both payload and communication protocol neutral.

Terms in *Italics* are defined in the ebXML Glossary of Terms [Glossary]. Terms listed in ***Bold Italics*** represent the element and/or attribute content of the XML *ebXML Message* Header. Terms listed in `Courier` font relate to MIME components.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [Bra97].

Note that the force of these words is modified by the requirement level of the document in which they are used.

> MUST: This word, or the terms “REQUIRED” or “SHALL”, means that the definition is an absolute requirement of the specification.
>
> MUST NOT: This phrase, or the phrase “SHALL NOT”, means that the definition is an absolute prohibition of the specification.
>
> SHOULD: This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
>
> SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

4.2 Audience

The target audience is software developers who will implement ebXML-messaging services.

4.3 Related Documents

The following set of related specifications will be delivered in phases:

- **ebXML Messaging Service Specification** (this document) **-** defines the structure of the messages and the behavior of messaging services software. This will include:
  - definitions of the messages

- behavior of the messaging service software
- reliable messaging
- message security
- extensibility and versioning

- **ebXML Trading Partner Specification** (under development) - defines how one party can discover and/or agree upon the information that party needs to know about another party prior to sending them a message that complies with this specification
- **ebXML Messaging Service Interface Specification** (to be developed) - defines an interface that may be used by software to interact with an ebXML Messaging Service
- **ebXML Messaging Services Security Specification** (in development) – defines the security mechanisms necessary to negate anticipated, selected threats
- **ebXML Messaging Services Requirements Specification** – defines the requirements of the Messaging Services

5 Design Objectives

5.1 Goals/Objectives/Requirements/Problem Description

The design objectives and goals are to define a Messaging Service (MS) to support XML based electronic business between small, medium and large enterprises. This is intended to be a low cost solution. It is the intention of the Transport, Routing and Packaging Project Team to keep this specification as simple and succinct as possible while remaining robust. Every item in this specification has been prototyped by the ebXML Proof of Concept team in order to ensure the clarity and succinctness of this specification. This specification is organized around the following two topics:

- **Packaging Specification** - A description of how to package an *ebXML Message* and associated parts. This section includes specifications for the various structures and containers. The Packaging Specification is a standard MIME multipart/related structure with two parts: XML Message Headers and Payload. The payload may be any type of data that MIME RFC 2045 and related IETF MIME extensions may support. The XML based Message Header elements and their structure were chosen after reviewing several current transports, both international and proprietary, to ensure that the appropriate header elements were included in the specification
- **Message Headers** - A specification of the structure and composition of the information necessary for an ebXML Messaging Service to successfully generate or process an ebXML compliant message.

Appendices to the specification cover:

- Appendix A Schemas and DTD Definitions
- Appendix B Examples
- Appendix C Candidate Packaging Technologies and Selection Process
- Appendix D MIME Type discussion
- Appendix E Communication Protocol Envelope Mappings
- Appendix F Detailed list of the Messaging Services Requirement Phases

5.2 Caveats and Assumptions

The specification is the first in a series of phased deliverables. This version of the specification does not address message security, extensibility, service interface, reliability, and versioning. These are being developed as separate documents and will be included in later versions of this document or as additional services specifications to the ebXML Message Services Specification.

It is assumed that the reader has an understanding of transports, MIME and XML.

6 System Overview

This document defines the enveloping and *ebXML Message* header structure used to transfer *ebXML Message*s over a data communication mechanism. This document provides sufficient detail to develop software for the packaging, exchange and processing of *ebXML Message*s.

6.1 What ebXML Messaging Services does

ebXML Messaging Services (MS) defines, robust yet basic functionality necessary to transfer messages between two ebXML Message Services using various existing transport protocols. The ebXML Messaging Service will perform in a manner which will allow for reliability, persistence of messages, security, and extensibility.

6.2 Where ebXML Messaging Services May Be Implemented

The ebXML Messaging Services is expected to be implemented in environments requiring a robust, low cost solution to enable electronic business.

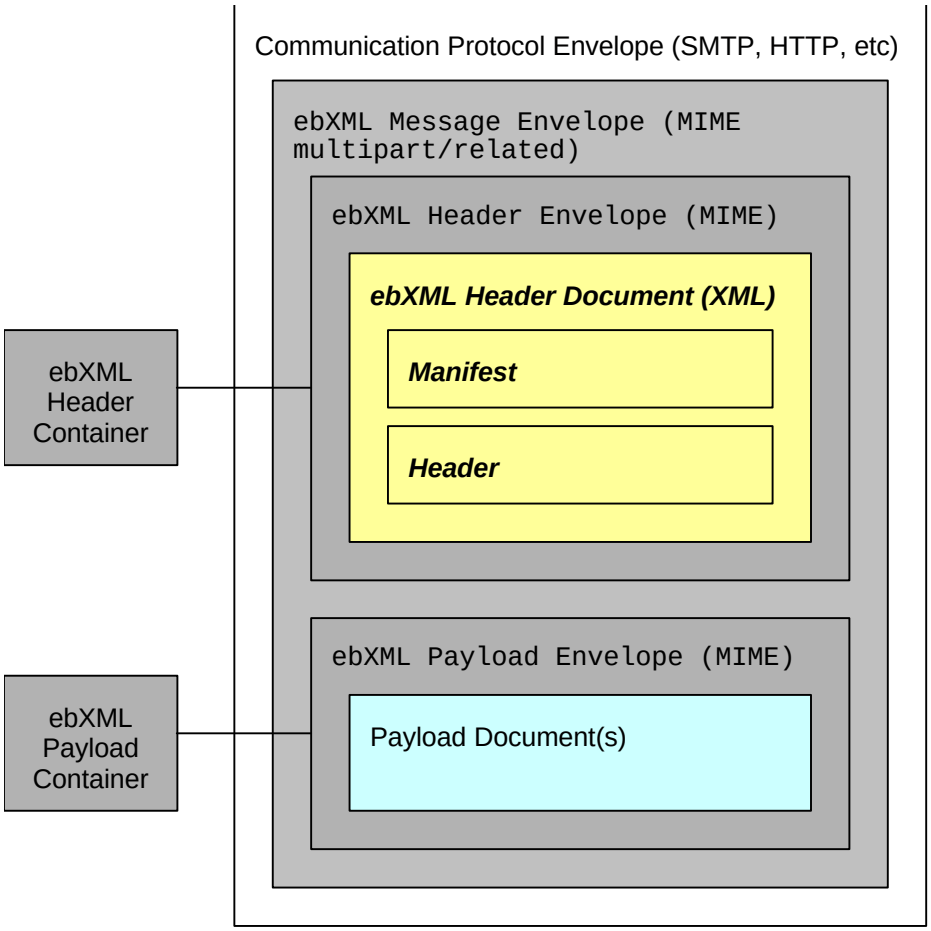
7 Definition and Scope

7.1 Packaging Specification

7.1.1 ebXML Message Structure

An *ebXML Message* consists of:

- an outer Communication Protocol Envelope, such as HTTP or SMTP,
- an inner communication “protocol independent” e*bXML Message Envelope*, specified using MIME multipart/related, that contains the two main parts of the Message:
  - an ebXML Header Container that is used to envelope one ebXML Header Document, and
  - an optional, single *ebXML Payload Container* that MUST be used to envelope the real payload (transferred data) of the Message



Note: The `Courier` font is used to represent MIME components. Items shown in ***bold italics*** represent XML items.

Figure 7-1 ebXML Message Structure

An *ebXML Header Envelope* and an *ebXML Payload Envelop* are constructed of standard, MIME components.

An *ebXML Header (or Payload) Document* is the content of the standard MIME part and is:

- an XML document in an ***ebXML Header***, or
- an XML or some other document for the ebXML Payload

Any special considerations for the usage the *ebXML Message Envelope* in HTTP and SMTP transports are described in Appendix E.

7.1.2 MIME usage Conventions

Values associated with MIME header attributes are valid in both quoted and unquoted form. For example, the forms `type="ebxml"` and `type=ebxml` are both valid.

7.2 ebXML Message Envelope

The MIME structured *ebXML Message Envelope* is used to identify the message as an ebXML compliant structure and encapsulates the header and payload in MIME body parts. It MUST conform to [RFC2045] and MUST contain two MIME headers:

- `Content-Type`
- `Content-Length`

7.2.1 Content-Type

The MIME `Content-Type` MUST be set to `multipart/related` for all *ebXML Message Envelope*s. See Appendix C for selection rationale. For example:

```
Content-Type: multipart/related;
```

The MIME `Content-Type` header contains four attributes:

- `type`
- `boundary`
- `version`
- `charset`

7.2.1.1 type Attribute

The MIME `type` attribute is used to identify the *ebXML Message Envelope* as an ebXML compliant structure. It conforms to a MIME XML Media Type [XMLMedia] and MUST be set to `"application/vnd.eb+xml"`. For example:

```
type="application/vnd.eb+xml"
```

7.2.1.2 boundary Attribute

The MIME `boundary` attribute is used to identify the body part separator used to identify the start and end points of each body part contained in the message. The MIME `boundary` SHOULD be chosen carefully to insure that it does not occur within the content area of a body part see [RFC 2045] for guidance on how to do this. For example:

```
boundary:="-------8760"
```

7.2.1.3 version Attribute

The MIME `version` attribute is used to identify the particular version of *ebXML Message Envelope* being used. All message headers SHOULD USE "0.21". For example:

```
version="0.21"
```

7.2.1.4 charset Attribute

The MIME `charset` attribute is used to identify the character set used to create the message. The list of valid values can be found at http://www.iana.org/. The MIME default charset value is "iso-8859-1". For example:

```
charset="iso-8859-1"
```

7.2.2 Content-Length

The MIME `Content-Length` header is a decimal value used to identify the total number of `OCTETS` contained in all constituent message body parts, including body part boundaries. Example:

```
Content-Length: 9841
```

7.2.3 ebXML Message Envelope Example

An example of a compliant *ebXML Message Envelope* header appears as follows:

```
Content-Type: multipart/related; type="application/vnd.eb+xml" "boundary:="-------8760"
charset="iso-8859-1" Content-Length: 9841
```

7.3 ebXML Header Container

The *ebXML Header Container* is a MIME body part that MUST consist of:

- one XML based ***ebXML Header Envelope***, and
- one XML ***ebXML Header Document***

The XML compliant ***ebXML Header Document*** is described in section 8 of this document.

The following rules apply:

- the *ebXML Header Container* MUST be the first MIME body part in the *ebXML Message.*
- there MUST be one and only one *XML ebXML Header Document* in each *ebXML Message*. However, an *ebXML Payload Container* may be a completely encapsulated *ebXML Message*.

The MIME based *ebXML Header Envelope* conforms to [RFC 2045] and MUST consist of three MIME headers:

- `Content-ID`
- `Content-Length`
- `Content-Type`

The *ebXML Header Document* within the content portion of the MIME container MAY be enhanced during transport, provided it has not been digitally signed. Any change in the size of the *ebXML Header Document* MUST be reflected in `Content-Length` attribute of the *ebXML Message Envelope* and *ebXML Header Envelope*.

7.3.1 Content-ID

The `Content-ID` MIME header identifies this instance of an ebXML Message header body part. The value for `Content-ID` SHOULD be a unique identifier, in accordance with RFC 2045. For example:

```
Content-ID: <2000-0722-161201-123456789@ebxmlhost.realm>
```

7.3.2 Content-Length

The MIME `Content-Length` header is a decimal value used to identify the total number of `OCTETS` contained in the *ebXML Header Container* MIME body part. For example:

```
Content-Length: 4208
```

7.3.3 Content-Type

The MIME `Content-Type` for an ebXML header is identified with the value "`application/vnd.eb+xml`". `Content-Type` MUST contain two attributes:

- `version`, and
- `charset`

7.3.3.1 version Attribute

- The MIME `version` attribute indicates the version of the ebXML Messaging Service Specification to which the *ebXML Header Document* conforms. For example:

```
version="1.0";
```

7.3.3.2 charset Attribute

The MIME `charset` attribute identifies the character set used to create the message. The list of valid values can be found at http://www.iana.org/.

The MIME `charset` attribute SHALL be equivalent to the encoding attribute of the *ebXML Header Document* (see section 8.1). For maximum interoperability it is RECOMMENDED that [UTF-8] be used. Note: this is not the default for MIME. For example:

```
charset="UTF-8"
```

7.3.4 ebXML Header Container Example

The following represents an example of an *ebXML Header Envelope* and *ebXML Header Document*:

```
Content-ID: ebxmlheader-123    ---------------|                          |
Content-Length: 2048                          |   MIME ebXML             |
Content-Type: application/vnd.eb+xml ---------|   Header Envelop         |  ebXML
                                                                         |  Header
<ebXMLHeader>                    -------------|                          |  Container
     <Manifest>........                       |   XML ebXML Header       |
     </Manifest>                              |   Document               |
     <Header>........                         |                          |
     </Header>                                |                          |
</ebXMLHeader>                   -------------|                          |
```

A complete example of an *ebXML Header Container* is presented in Appendix B.

7.4 ebXML Payload Container

If the *ebXML Message* contains a payload, then a single *ebXML Payload Container* MUST be used to envelop it.

If there is no payload within the *ebXML Message* then the *ebXML Payload Container* MUST not be present.

The contents of the *ebXML Payload Container* MUST be identified by the *Message Manifest* element within the *ebXML Header Document* (see section 8.3).

If the *Message Manifest* is an empty XML element then an ebXML Payload Container MUST NOT be present in the *ebXML Message*.

If an *ebXML Payload Container* is present then it MUST conform to MIME [RFC2045] and MUST consist of:

- a MIME header portion - the *ebXML Payload Envelope*, and
- a content portion - the payload itself which may be of any valid MIME type.

The *ebXML MIME Payload Envelope*, MUST consist of three MIME headers:

- `Content-ID`
- `Content-Length`
- `Content-Type`

The ebXML Messaging Service Specification makes no provision, nor limits in any way the structure or content of payloads. Payloads MAY be a simple-plain-text-object or complex nested multipart objects. This is the implementer's decision.

7.4.1 Content-ID

The `Content-ID` MIME Header is used to uniquely identify an instance of an *ebXML Message* payload body part. The value for `Content-ID` SHOULD be a unique identifier, in accordance with MIME [RFC 2045]. For example:

```
Content-ID: <2000-0722-161201-987654321@ebxmlhost.realm>
```

7.4.2 Content-Length

The MIME `Content-Length` header is a decimal value used to identify the total number of `OCTETS` contained in the content portion of the *ebXML Payload Container*. For example:

```
Content-Length: 5012
```

7.4.3 Content-Type

The MIME `Content-Type` for an ebXML payload is determined by the implementer and is used to identify the type of data contained in the content portion of the *ebXML Payload Container*. For example:

```
Content-Type: application/xml
```

7.4.4 Example of an ebXML MIME Payload Container

The following represents an example of an *ebXML MIME Payload Envelope* and a payload:

```
Content-ID: ebxmlpayload-123      -------------|                            |
Content-Length: 4096                           | ebXML MIME                 |
Content-Type: application/xml     -------------| Payload Envelope           |  ebXML
                                               |                            |  Payload
<Invoice>                         -------------|                            |  Container
    <Invoicedata>........                      | Payload                    |
    </Invoicedata>                             |                            |
</Invoice>                        -------------|                            |
```

A complete example of the ebXML Payload Container is presented in Appendix B.

8 ebXML Header Document

The *ebXML Header Document* is a single [XML] document with a number of principal header-elements within it where each principal header-element is a separate XML element.
In general, separate principal-header elements are used where:

- different software is likely to be used to generate that header-element,
- the structure of the header element might vary independently of the other header-elements, or
- the data contained in the header-element MAY need to be digitally signed separately from the other header-elements.

8.1 XML Prolog

The XML prolog for the ebXML Message header document SHALL contain the encoding attribute which SHALL be equivalent to the `charset` attribute of the MIME `Content-Type` of the ebXML Message Header Container (see section 7.3.3.27.3.3.2). It is RECOMMENDED that UTF-8 be used explicitly although this is one of the default values assumed if none is specified.

NOTE: The encoding attribute is OPTIONAL in the XML version 1.0 specification [4], however, it is mandatory for the ebXML message header to ensure no conflicts occur with the `charset` attribute of the MIME Content-Type of the container and to ensure maximum interoperability.

An example follows

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

8.2 Root Element

The root element of the *XML ebXML Header Document* is named ***ebXMLHeader***. It is comprised of three XML attributes and two subordinate elements.

The first attribute is the namespace declaration (***xmlns***) (see [XML Namespace] that has a REQUIRED value of "`http://www.ebxml.org/namespaces/messageHeader`".

The second attribute is the ***Version*** attribute. This attribute is required. Its purpose is to provide for future versioning capabilities. It has a default value of '1.0'.

The last of the ***ebXMLHeader*** attributes is the ***MessageType*** attribute. Its purpose is to enable ebXML-aware software to distinguish between normal and communication protocol-specific messages, such as acknowledgment and error messages. The ***MessageType*** is an enumeration consisting of three possible values:

- ***Normal*** – the ebXML Payload Container contains data that has been provided to the ebXML Messaging Service by the software that called it
- ***Acknowledgment*** – a ebXML Messaging Service-specific acknowledgment message.
- ***Error*** – an ebXML Messaging Service-specific error message.

The ***ebXMLHeader*** element MUST contain the following two elements:

- **Manifest** - contains a list of references to he other parts of the Message. This includes references to the documents, which comprise the *Payload* of the *Message.*
- **Header** - contains the information REQUIRED by the recipient to process the message. The message originator creates this information to which additional information MAY be added.

The ***Header*** and ***Manifest*** are REQUIRED elements in every *Message*.

The following is a sample ***ebXMLHeader*** document fragment demonstrating the overall structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<ebXMLHeader xmlns="http://www.ebxml.org/namespaces/messageHeader"
        Version="1.0" MessageType="Normal">
        <Manifest>...</Manifest>
        <Header>...</Header>
</ebXMLHeader>
```

8.3 XML Manifest

The required ***Manifest*** element is a composite element consisting of zero or more ***DocumentReference*** elements. Each ***DocumentReference*** element identifies data associated with the message, whether included as part of the message, or remote resources accessible via a URL. The ***Manifest*** SHALL be the first subordinate element in the ***ebXMLHeader***. It identifies the payload document(s) contained in the *ebXML Message Container*. The purpose of the ***Manifest*** is to make it easier to directly extract a particular document associated with the Message. See also section ???

8.3.1 XML DocumentReference

The ***DocumentReference*** element is a composite element consisting of two required subordinate elements as follows:

- ***DocumentDescription*** - an optional textual description of the document/resource
- ***DocumentLabel*** - a code that enables the purpose of the referenced document to be determined without retrieving it
- ***DocumentId*** - a URL of the `Content-ID` of a MIME body part, as defined in [RFC2392], representing payload data, or a remote URL to some external resource.

The following fragment demonstrates a typical ***Manifest*** for a message with a single payload MIME body part:

```
<Manifest>
        <DocumentReference>
                <DocumentLabel>PurchaseOrder</DocumentLabel>
                <DocumentId>cid:0987654321</DocumentId>
        </DocumentReference>
</Manifest>
```

8.4 XML Header

The ***Header*** element immediately follows the ***Manifest*** element. It is required in all ***ebXMLHeader*** documents. The ***Header*** element is a composite element comprised of the following required subordinate elements:

- ***From*** – the logical address of the sender of the message.
- ***To*** – the logical address of the intended recipient of the message.
- ***TPAInfo*** – a composite set of information which relates to the *Trading Partner Agreement* under which the message is governed
- ***MessageData*** – a composite set of information which uniquely identifies the *Message*

- ***ReliableMessagingInfo*** - information which identifies the degree of reliability with which the message SHOULD be delivered

The following fragment demonstrates the structure of the ***Header*** element of the ***ebXMLHeader*** document:

```
<Header>
        <From>...</From>
        <To>...</To>
        <TPAInfo>...</TPAInfo>
        <MessageData>...</MessageData>
        <ReliableMessagingInfo>...</ReliableMessagingInfo>
</Header>
```

8.4.1 From and To

The ***From*** element identifies the *Party* which originated the message. It is a logical identifier, which MAY take the form of a URN. An example of this would be a DUNS number. The ***From*** element consists of a ***PartyId*** element.

The ***To*** element identifies the intended recipient of the message. As with ***From***, it is a logical identifier which is comprised of a ***PartyId*** element.

The **PartyId** element has a single attribute; ***context*** and a text value. The purpose of the context attribute is to provide a context for the text value of the ***PartyId*** element. The following fragment demonstrates usage of the ***From*** and ***To*** elements of the ***ebXMLHeader***.

```
<From>
        <PartyId context="DUNS">12345</PartyId>
</From>
<To>
        <PartyId context="DUNS">54321</PartyId>
</To>
```

8.4.2 TPAInfo

The ***TPAInfo*** element follows the ***From*** and ***To*** elements in the ***Header*** element structure. The ***TPAInfo*** element is a composite set of information that relates to the *Trading Partner Agreement* under which the message is governed. The ***TPAInfo*** element has four subordinate elements as follows:

- ***TPAId*** – a URI which identifies the *Trading Partner Agreement* which governs the processing of the message
- ***ConversationId*** – a URI which identifies the set of related messages that make up a conversation between two ***Parties***
- ***ServiceInterface*** – Identifies the Service Interface that SHOULD act on the payload in the message. It is unique within the domain of the ***Party*** to which the message is being sent. URN's MAY be considered suitable for the element content.
- ***Action*** – Identifies a process within a Service Interface, which processes the Message. ***Action*** SHALL be unique within the Service Interface in which it is defined.

The following example fragment demonstrates the usage of the ***TPAInfo*** element.

```
<TPAInfo>
```

```
        <TPAId context = "tpadb">12345678</TPAId>
        <ConversationId context = "tpadb">987654321</ConversationId>
        <ServiceInterface>QuoteToCollect</ServiceInterface>
        <Action>NewPurchaseOrder</Action>
</TPAInfo>
```

8.4.3 MessageData

The required ***MessageData*** element follows the ***TPAInfo*** element. The purpose of the ***MessageData*** element is to provide a means of identifying an *ebXML Message*. It is a composite element that contains the following three elements:

- ***MessageId*** – a unique identifier for the message conforming to [RFC2392]. The "local part" of the identifier is implementation dependent.
- ***TimeStamp*** – a value representing the time that the message header was created conforming to [ISO-8601]. The format of CCYYMMDDTHHMMSS.SSSZ is used. This time format is Coordinated Universal Time (UTC).
- ***RefToMessageId*** – an optional reference to an earlier *ebXML Message*. If there is no earlier message then the element MUST be empty. If element is not empty then it MUST contain the value of the ***MessageId*** of the earlier related *ebXML Message*.

The following example demonstrates the usage of the ***MessageData*** element.

```
<MessageData>
    <MessageId>UUID-2</MessageId>
    <TimeStamp>20000725T121905.000Z</TimeStamp>
    <RefToMessageId>UUID-1</RefToMessageId>
</MessageData>
```

8.4.4 ReliableMessagingInfo

The last element of the ***ebXMLHeader*** is the ***ReliableMessagingInfo*** element. This element identifies the degree of reliability with which the message will be delivered. This element has a single attribute, ***DeliverySemantics.*** This attribute is an enumeration, which may have one of the following values:

- "AtMostOnce" – reliable messaging semantics, which specifies that the Service Interface handler will receive a given message no more than once.
- "Unspecified" – reliable delivery semantics are not specified.

```
<ReliableMessagingInfo>
    <DeliverySemantics>AtMostOnce</DeliverySemantics>
</ReliableMessagingInfo>
```

9 References

9.1 Normative References

[Glossary] ebXML Glossary, see ebXML Project Team Home Page

[ISO 8601] International Standards Organization Ref. ISO 8601 Second Edition, Published 1997

[RFC 2392] IETF Request For Comments 2392. Content-ID and Message-ID Uniform Resource Locators. E. Levinson, Published August 1998

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

[TRPREQ] ebXML Transport, Routing and Packaging: Overview and Requirements, Version 0.96, Published 25 May 2000

[UTF-8] UTF-8 is an encoding that conforms to ISO/IEC 10646. See [XML] for usage conventions.

[XML Namespace] Recommendation for Namespaces in XML, World Wide Web Consortium, 14 January 1999, http://www.w3.org/TR/REC-xml-names

[XMLMedia] IETF Internet Draft on XML Media Types. See http://www.imc.org/draft-murata-xml Note. It is anticipated that this Internet Draft will soon become a RFC. Final versions of this specification will refer to the equivalent RFC.

[XML] Extensible Mark Up Language. A W3C recommendation. See http://www.w3.org/TR/1998/REC-xml-19980210 for the 10 February 1998 version.

9.2 Non-Normative References

[XMTP] XMTP - Extensible Mail Transport Protocol http://www.openhealth.org/documents/xmtp.htm

10 Disclaimer

The views and specification expressed in this document are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this design.

11 Contact Information

Team Leader

Name	Rik Drummond
Company	Drummond Group, Inc.
Street	5008 Bentwood Crt.
City, State, Postal Code	Fort Worth, Texas 76132
Country	USA
Phone:	817.294.7339
EMail:	rik@drummondgroup.com

Vice Team Leader

Name	Chris Ferris
Company	Sun Microsystems
Street	One Network Drive
City, State, Postal Code	Burlington, MA 01803-0903
Country	USA
Phone:	(781) 442-3063
EMail:	chris.ferris@sun.com

Team Editor

Name	David Burnett
Company	CommerceOne
Street	4400 Rosewood Drive 3rd Fl, Bldg 4
City, State, Postal Code	Pleasanton, CA 94588
Country	USA
Phone:	(925) 520-4422 or (650) 623-2888
EMail:	david.burdett@commerceone.com

Document Editing Team

Name	Ralph Berwanger
Company	bTrade.com
Street	2324 Gateway Drive
City, State, Postal Code	Irving, TX 75063
Country	USA
Phone:	(972) 580-2900
EMail:	rberwanger@btrade.com

Name	Ian Jones
Company	British Telecommunications
Street	Enterprise House, 84-85 Adam Street
City, State, Postal Code	Cardiff, CF241XF
Country	UK
Phone:	+44 29 2072 4063
EMail:	ian.c.jones@bt.com

Name	Martha Warfelt
Company	DaimlerChrysler Corporation
Street	800 Chrysler Drive
City, State, Postal Code	Auburn Hills, MI
Country	USA
Phone:	(248) 944-5481
EMail:	maw2@daimlerchrysler.com

Appendix A Schemas and DTD Definitions

The following are definitions for validation of the *ebXML Message* header structure.

A.1 XML Header DTD

```
<?xml version ="1.0"?>
<schema xmlns = "http://www.w3.org/1999/XMLSchema">

<!ELEMENT ebXMLHeader  (Manifest , Header )>
<!ATTLIST ebXMLHeader  Version     CDATA  #FIXED '1.0'
                  MessageType CDATA  #FIXED 'Normal' >
<!ELEMENT Manifest  (DocumentReference )+>
<!ELEMENT DocumentReference  (Document Description?, DocumentLabel , DocumentId )>
<!ELEMENT DocumentDescription  (#PCDATA )>
<!ATTLIST DocumentDescription e-dtype NMTOKEN  #FIXED 'string' >
<!ELEMENT DocumentLabel  (#PCDATA )>
<!ATTLIST DocumentLabel  e-dtype NMTOKEN  #FIXED 'string' >
<!ELEMENT DocumentId  (#PCDATA )>
<!ATTLIST DocumentId  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT Header  (From , To , TPAInfo , MessageData , ReliableMessagingInfo )>
<!ELEMENT TPAInfo  (TPAId , ConversationId , ServiceInterface , Action )>
<!ELEMENT ServiceInterface  (#PCDATA )>
<!ATTLIST ServiceInterface  e-dtype NMTOKEN  #FIXED 'string' >
<!ELEMENT Action  (#PCDATA )>
<!ATTLIST Action  e-dtype NMTOKEN  #FIXED 'string' >
<!ELEMENT TPAId  (#PCDATA )>
<!ATTLIST TPAId  context CDATA    'Undefined'
            e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT ConversationId  (#PCDATA )>
<!ATTLIST ConversationId  context CDATA    'Undefined'
                  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT MessageData  (MessageId , TimeStamp , RefToMessageId )>
<!ELEMENT RefToMessageId  (#PCDATA )>
<!ATTLIST RefToMessageId  e-dtype NMTOKEN  #FIXED 'uuid' >
<!ELEMENT TimeStamp  (#PCDATA )>
<!ATTLIST TimeStamp  e-dtype NMTOKEN  #FIXED 'dateTime' >
<!ELEMENT MessageId  (#PCDATA )>
<!ATTLIST MessageId  e-dtype NMTOKEN  #FIXED 'uuid' >
<!ELEMENT From  (PartyId )>
<!ELEMENT To  (PartyId )>
<!ELEMENT PartyId  (#PCDATA )>
<!ATTLIST PartyId  context CDATA    'Undefined'
              e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT ReliableMessagingInfo EMPTY>
```

```
<!ATTLIST ReliableMessagingInfo  DeliverySemantics  (AtMostOnce | Unspecified )  #FIXED 'Unspecified' >
```

A.2 XML Header Schema Definition

```
<?xml version ="1.0"?>
<schema xmlns = "http://www.w3.org/1999/XMLSchema">
        <element name = "ebXMLHeader">
                <complexType content = "elementOnly">
                        <sequence>
                                <element ref = "Manifest"/>
                                <element ref = "Header"/>
                        </sequence>
                        <attribute name="Version" use="fixed" value="1.0" type="string"/>
                        <attribute name="MessageType" use="fixed" value="Normal" type = "string"/>
                </complexType>
        </element>
        <element name = "Manifest">
                <complexType content = "elementOnly">
                        <sequence minOccurs = "0" maxOccurs = "unbounded">
                                <element ref = "DocumentReference"/>
                        </sequence>
                </complexType>
        </element>
        <element name = "DocumentReference">
                <complexType content = "elementOnly">
                        <sequence minOccurs = "1" maxOccurs = "unbounded">
                                <element ref = "DocumentDescription" />
                                <element ref = "DocumentLabel"/>
                                <element ref = "DocumentId"/>
                        </sequence>
                </complexType>
        </element>
        <element name = "DocumentLabel" type = "string">
        </element>
        <element name = "DocumentId" type = "uri">
        </element>
        <element name = "Header">
                <complexType content = "elementOnly">
                        <sequence>
                                <element ref = "From"/>
                                <element ref = "To"/>
                                <element ref = "TPAInfo"/>
                                <element ref = "MessageData"/>
                                <element ref = "ReliableMessagingInfo"/>
                        </sequence>
                </complexType>
        </element>
        <element name = "BusinessServiceInterface" type = "string">
        </element>
        <element name = "Action" type = "string"/>
        <element name = "TPAId">
                <complexType base = "uri" content = "textOnly">
                        <attribute name="context" use="default" value="Undefined" type = "string"/>
                </complexType>
        </element>
        <element name = "ConversationId">
                <complexType base = "uri" content = "textOnly">
                        <attribute name = "context" use = "default" value = "Undefined" type = "string"/>
                </complexType>
        </element>
        <element name = "MessageData">
                <complexType content = "elementOnly">
                        <sequence>
                                <element ref = "MessageId"/>
                                <element ref = "TimeStamp"/>
                                <element ref = "RefToMessageId"/>
                        </sequence>
```

```
                </complexType>
        </element>
        <element name = "RefToMessageId" type = "uuid">
        </element>
        <element name = "TimeStamp" type = "dateTime">
        </element>
        <element name = "MessageId" type = "uuid">
        </element>
        <element name = "From">
                <complexType content = "elementOnly">
                        <sequence>
                                <element ref = "PartyId"/>
                        </sequence>
                </complexType>
        </element>
        <element name = "To">
                <complexType content = "elementOnly">
                        <sequence>
                                <element ref = "PartyId"/>
                        </sequence>
                </complexType>
        </element>
        <element name = "PartyId">
                <complexType base = "uri" content = "textOnly">
                        <attribute name = "context" use = "default" value = "Undefined" type = "string"/>
                </complexType>
        </element>
        <element name = "ReliableMessagingInfo">
                <complexType content = "empty">
                        <attribute name = "DeliverySemantics" use = "fixed" value = "Unspecified">
                                <simpleType base = "ENUMERATION">
                                        <enumeration value = "AtMostOnce"/>
                                        <enumeration value = "Unspecified"/>
                                </simpleType>
                        </attribute>
                </complexType>
        </element>
        <element name = "TPAInfo">
                <complexType content = "elementOnly">
                        <sequence>
                                <element ref = "TPAId"/>
                                <element ref = "ConversationId"/>
                                <element ref = "BusinessServiceInterface"/>
                                <element ref = "Action"/>
                        </sequence>
                </complexType>
        </element>
</schema>
```

Appendix B Examples

The following are complete examples of *ebXML Message*s showing the structure as defined in this specification.

B.1 Complete Example of an ebXML Message Envelope using multipart/related Content-Type sent via HTTP POST

```
POST /ebxmlhandler HTTP/1.1
Accept: multipart/related
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Group 8760 InsideAgent
Host: localhost:9090
Connection: Keep-Alive
Content-Type: multipart/related; type=application/vnd.eb+xml; version=0.1;
boundary=----------------------------7d02a82e5f8
Content-Length: 9293

------------------------------7d02a82e5f8
Content-ID: ebxmlheader-9981
Content-Length: 211
Content-Type: application/vnd.eb+xml; charset="UTF-8";

<?xml version="1.0" encoding="UTF-8"?>
<ebXMLHeader xmlns = “http://www.ebxml.org/namespaces/messageHeader"
          Version = "1.0"
          MessageType = "Normal">
   <Manifest>
      <DocumentReference>
         <DocumentLabel>Purchase Order Request Action</DocumentLabel>
         <DocumentId>
            cid:uid@originator-domain [C-ID of the payload MIME part]
         </DocumentId>
      </DocumentReference>
   </Manifest>
   <Header>
      <From>
         <PartyId context = "DUNS">requester-DUNS-number</PartyId>
      </From>
      <To>
         <PartyId context = "DUNS">responder-DUNS-number</PartyId>
      </To>
      <TPAInfo>
```

```
        <TPAId context = "tpadb">
          /requester-DUNS-number/responder-DUNS-number/PIP3A4/1.1
        </TPAId>
        <ConversationId context = "CreatePurchaseOrder">
          uid@requester-domain
        </ConversationId>
        <BusinessServiceInterface>
          Seller Service
        </BusinessServiceInterface>
        <Action version="1.1">Purchase Order Request Action</Action>
      </TPAInfo>
      <MessageData>
        <MessageId>uid@requester-domain</MessageId>
        <TimeStamp>CCYYMMDDThhmmss.sssZ</TimeStamp>
        <RefToMessageId>Not Applicable</RefToMessageId>
      </MessageData>
      <ReliableMessagingInfo DeliverySemantics = "Unspecified"/>
    </Header>
</ebXMLHeader>
----------------------------7d02a82e5f8
Content-ID: ebxmlpayload-9981
Content-Length: 7517
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v2.5  - http://www.xmlspy.com -->
<HITISMessage xmlns="" Version="1.0">
            <Header OriginalBodyRequested="false" ImmediateResponseRequired="true">
                        <FromURI>http://www.pms.com/HITISInterface</FromURI>
                        <ToURI>http://www.crs.com/HITISInterface</ToURI>
                        <ReplyToURI>http://www.pms.com/HITISInterface</ReplyToURI>
                        <MessageID>1234567890</MessageID>
                        <OriginalMessageID>1234567890</OriginalMessageID>
                        <TimeStamp>1999-11-10T10:23:44</TimeStamp>
                        <Token>1234-567-8901</Token>
                        <!--Token to be assigned in response to HITISRegister-->
            </Header>
            <Body>
            <HITISOperation OperationName="CommissionEventsUpdate">
                                    <BodyPartstuffgoeshere/>

            </HITISOperation>
            </Body>
</HITISMessage>

----------------------------7d02a82e5f8--
```

B.2 Complete Example of an ebXML Message Envelope using multipart/related Content-Type sent via SMTP

The default Content-transfer-encoding type of 7BIT is being used in this message.

```
From dick@8760.com Sun May  7 17:01:14 2000
Received: from granger.mail.mindspring.net by alpha2000.tech-comm.com; (8.8.5/1.1.8.2/05Jun95-1217PM)
          id RAA32702; Sun, 7 May 2000 17:01:13 -0500 (CDT)
Received: from gamma (user-33qt10l.dialup.mindspring.com [199.174.132.21])
          by granger.mail.mindspring.net (8.9.3/8.8.5) with SMTP id SAA11942
          for <ebxmlhandler@8760.com>; Sun, 7 May 2000 18:11:14 -0400 (EDT)
From: "Dick Brooks (E)" <dick@8760.com>
To: <ebxmlhandler@8760.com>
Subject: OTA Commission Event
Date: Sun, 7 May 2000 17:07:38 -0500
Message-ID: <NDBBIOBLMLCDOHCHIKMGKEEIDAAA.dick@8760.com>
MIME-Version: 1.0
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300
Content-Length: 8081
Content-Type: multipart/related; type="application/vnd.eb+xml"; version="0.1"; charset="iso-8859-1"; boundary="---
=_NextPart_000_0005_01BFB846.BF7FABA0"

------=_NextPart_000_0005_01BFB846.BF7FABA0
Content-Type: application/vnd.eb+xml
Content-ID: ebxmlheader-9000
Content-Length: 272

<?xml version="1.0" encoding="UTF-8"?>
<ebXMLHeader xmlns = “http://www.ebxml.org/namespaces/messageHeader"
          Version = "1.0"
          MessageType = "Normal">
   <Manifest>
      <DocumentReference>
         <DocumentLabel>Purchase Order Request Action</DocumentLabel>
         <DocumentId>
            cid:uid@originator-domain [C-ID of the payload MIME part]
         </DocumentId>
      </DocumentReference>
   </Manifest>
   <Header>
      <From>
         <PartyId context = "DUNS">requester-DUNS-number</PartyId>
      </From>
```

```
      <To>
         <PartyId context = "DUNS">responder-DUNS-number</PartyId>
      </To>
      <TPAInfo>
         <TPAId context = "tpadb">
            /requester-DUNS-number/responder-DUNS-number/PIP3A4/1.1
         </TPAId>
         <ConversationId context = “CreatePurchaseOrder">
            uid@requester-domain
         </ConversationId>
         <BusinessServiceInterface>
            Seller Service
         </BusinessServiceInterface>
         <Action version=”1.1”>Purchase Order Request Action</Action>
      </TPAInfo>
      <MessageData>
         <MessageId>uid@requester-domain</MessageId>
         <TimeStamp>CCYYMMDDThhmmss.sssZ</TimeStamp>
         <RefToMessageId>Not Applicable</RefToMessageId>
      </MessageData>
      <ReliableMessagingInfo DeliverySemantics = "Unspecified"/>
   </Header>
</ebXMLHeader>
------=_NextPart_000_0005_01BFB846.BF7FABA0
Content-Type: text/xml
Content-ID: ebxmlpayload-9000
Content-Length: 7515

<?xml version="1.0" encoding="UTF-8"?>
<HITISMessage xmlns="" Version="1.0">
            <Header OriginalBodyRequested="false" ImmediateResponseRequired="true">
                        <FromURI>http://www.pms.com/HITISInterface</FromURI>
                        <ToURI>http://www.crs.com/HITISInterface</ToURI>
                        <ReplyToURI>http://www.pms.com/HITISInterface</ReplyToURI>
                        <MessageID>1234567890</MessageID>
                        <OriginalMessageID>1234567890</OriginalMessageID>
                        <TimeStamp>1999-11-10T10:23:44</TimeStamp>
                        <Token>1234-567-8901</Token>
            </Header>
            <Body>
                        <HITISOperation OperationName="CommissionEventsUpdate">
                                    <BodyPartstuffgoeshere/>
                        </HITISOperation>
            </Body>
</HITISMessage>
------=_NextPart_000_0005_01BFB846.BF7FABA0--
```

Appendix C Candidate Packaging Technologies and Selection Process

The packaging sub-group began its investigation of packaging technologies by identifying the technologies currently used for business-to-business message exchange or were being developed for this purpose. The following packaging technologies were identified:

- MIME - currently in use by companies exchanging business transactions using E-mail and HTTP
- XML - currently used by RosettaNet and Microsoft (BizTalk and SOAP) and others

C.1 Selection Process

Each candidate technology was evaluated based on its ability to meet the requirements listed in the section titled "Packaging and other Requirements" in this document. When necessary, specific parties were contacted to provide details describing how a technology was being used to meet specific requirements. The following parties were contacted to provide expert insight:

- Microsoft - David Turner, regarding use of XML packaging in BizTalk
- Develop Mentor - Don Box, regarding use of XML packaging in SOAP
- Vitria - Prasad Yendluri, regarding use of XML packaging in RosettaNet
- Jonathan Borden - author of [XMTP], an XML to MIME transformation tool

The packaging sub-group considered the inputs of people from the ebXML Transport mailing list as well as the parties listed above, before making a selection.

C.2 MIME

Multipurpose Internet Mail Extensions (MIME) is an international standard created by the Internet Engineering Task Force. It has been implemented by numerous software vendors across the globe and has been used to exchange mixed type payloads, including XML, for several years. MIME was designed purely as a packaging (enveloping) solution to allow the transport of mixed payloads using Internet E-mail (SMTP). MIME is also being used by other transport technologies as a packaging technology, most notably HTTP.

C.3 XML

eXtensible Markup Language (XML) version 1.0 is a technical specification holding a RECOMMENDED status created by the World Wide Web Consortium. It has been implemented by numerous software vendors across the globe and has been used to describe a broad spectrum of document structures from very simple to very complex. XML is a very flexible markup language that can be used to represent virtually any type of document. XML can be used solely for packaging (enveloping) documents of any type, providing the data can be "transformed" into "legal" XML.
In some cases, XML documents MUST be placed into transport specific "envelopes" before being transported. For example, XML data MUST be placed in a MIME envelope when being transported via SMTP or HTTP.

C.4 Conclusion

The packaging sub-group examined the capabilities of both XML and MIME relative to the list of packaging requirements above. It's important to note that neither technology met all of the ebXML requirements and in the end it was the packaging sub-groups assessment of which technology

came closest to meeting ALL of the ebXML requirements that determined which technology SHOULD be used.
MIME was chosen to serve as the ebXML packaging technology, over XML, based on the information contained in following table:

Reason	**Requirement(s) Satisfied**
There is no formal packaging recommendation within IETF or W3C, based on XML. If ebXML were to choose XML as a packaging technology it would be required to define an XML packaging specification and submit this to IETF or W3C for adoption as a formal standard.	to not reinvent the wheel - re-use where possible [TRPREQ]
XML requires that binary and other types of payload data including XML documents be base64 encoded in order to be encapsulated within a XML root document. Base64 encoding ensures that no illegal XML characters exist within a document and recursive XML documents are "hidden". Base64 encoding imposes a significant processing overhead and results in larger messages, which affect both transmission and processing times. Base64 encoding of binary data is required of MIME content when being transported by SMTP, but this is a transport level requirement, not a requirement imposed by MIME. Binary data can be packaged and transported without alteration when using MIME over HTTP	Minimize intrusion to payload (special encoding or alteration) Low processing overhead
At the time of defining this specification there is no industry standard way to package an encrypted message, or portion of a message, using XML.	All or part of the documents in a message MAY be encrypted prior to sending [TRPREQ]
MIME could be used in conformance within existing IETF recommendations, no additions or changes are initially required to produce a functional envelope.	to not reinvent the wheel - re-use where possible [TRPREQ]

Appendix D MIME Type discussion

Three MIME media types were considered to serve as Content-Type for the *ebXML Message Envelope*:

- Multipart/related
- Multipart/Mixed
- Multipart/form-data

The group selected the multipart/related media type to serve as the preferred message envelope Content-Type.

Note:

There was some discussion over the similarities of multipart/related and multipart/mixed, both of which appear to offer similar capabilities and both could meet stated requirements. However, the group converged on multipart/related, believing it to be more semantically appropriate for ebXML.

There was significant discussion over whether to support multipart/form-data as an alternate Content-Type for message-envelope, due to the large installed base of web browsers that support this Content-Type.

It was determined that multipart/related was a more generic Content-Type than multipart/form-data and the multipart/related Content-Type is the preferred Content-Type for ebXML Message Envelopes. Multipart/form-data Content-Type is typically associated with HTTP/HTML web forms, whereas multipart/related can be associated with any type of data.

Additionally, due to limitations in their handling of multipart ebXML payloads it was determined that existing web browsers are unable to support the full breadth of functions needed to package complex *ebXML Message*s containing multipart payloads. Therefore browser vendors are encouraged to add support for the ebXML enveloping standard as specified in this document.

Appendix E Communication Protocol Envelope Mappings

This section provides rules and definitions for the completion of the Communication Protocol Envelope Mappings for HTTP [RFC xxxx], SMTP [RFC xxxx] and FTP [RFC xxxx].

E.1 HTTP

To be completed

E.2 SMTP

To be completed

E.3 FTP

To be completed

Copyright Statement