INDEX

Internet Draft
draft-ietf-trade-xmlmsg-requirements-00.txt
Expires: July 2000

Commerce One
January 2000
David Burdett

Requirements for XML Messaging
Version 1.0 Release 00

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Distribution of this document is unlimited. Please send comments to the TRADE working group at <ietf-trade@lists.elistx.com >, which may be joined by sending a message with subject "subscribe" to <ietf-trade-request@lists.elistx.com>.

Discussions of the TRADE working group are archived at http://www.elistx.com/archives/ietf-trade.

Abstract

This specification contains requirements for a generic approach to the reliable, resilient, secure, tamper resistant, authenticated exchange of XML or other electronic documents over insecure transport mechanisms.

The specification provides the requirements and framework for a set of related specifications on XML Messaging covering:
- Requirements for XML Messaging (this paper)
- Common XML Message Elements
- Document Exchange and Transactions
- Reliable Messaging
- Secure Messaging
- Document Choreography Definitions
- Common Document Exchanges
- Transport Protocol Supplements

Although much work has been carried out on the other parts of the XML Messaging specification described above, they are sill in development

Table of Contents

Table of Figures

1. Background

Many Internet protocols consist of an exchange of documents over the Internet, for example sending a Purchase Order and receiving a Purchase Order Acknowledgement in return.

Many of these protocols use or plan to use XML to represent those documents. However irrespective of the way in which documents are represented, there are common requirements that are independent of the purpose of the protocols, the information they are carrying or the processes that create or accept those documents.

The XML Messaging specification provides a generic approach to the reliable, resilient, secure, tamper resistant, authenticated exchange of XML or other electronic documents over insecure, unreliable transport mechanisms. More specifically it provides, a series related specifications that cover:

- wrapping a document
- identifying a document
- describing the processing of a document
- reporting errors in a document
- digitally signing documents to provide tamper resistance and security
- exchanging a series of documents to carry out a process or service
- authenticating a document so that the intended recipient knows:
  - who sent and/or authorized the document, and therefore that
  - the document should be processed and a response returned
- providing a record of the success or failure of a process/service
- securely relating a series of sub-services together to create a larger, more complex service where the start of one sub-service is dependent on the authorized, successful completion of another
- providing an audit-trail of the execution of a set of services that can be used after the event as evidence that those services were carried out
- inquiring on the success or failure of a service initiated by sending a document
- determining if a failed service can be restarted and then restarting it where possible

- detecting whether or not a system or server is up-and-running and able to accept documents
- authenticating a user or server that is providing a service.

1.1 Structure

This specification is one of a set of related specifications on XML Messaging covering:

- Requirements for XML Messaging (this paper)
- Common XML Message Elements [XMLMSG-CME]. This defines the XML elements and attributes used to construct Messages that conform to XML Messaging
- Document Exchanges and Transactions [XMLMSG-DET]. This defines standard templates for exchanging documents between parties that can be used to implement transactions that support different types of services and processes.
- Reliable Messaging [XMLMSG-RM]. This defines how to exchange messages in a way that is reliable, robust and resilient and results in "guaranteed once-only" message delivery
- Secure Messaging [XMLMSG-SM]. This describes how digital signatures and other methods such as [SSL] may be used to ensure the tamper resistance and authenticated exchange of messages
- Document Choreography Definitions [XMLMSG-DCD]. This describes how the sequence in which documents are exchanged may be defined, agreed between the parties involved and then used dynamically to determine the way in which a particular type of business service or process is carried out
- Common Document Exchanges [XMLMSG-CDE]. This defines a number of Common Document Exchanges that are generally applicable to many situations
- Transport Protocol Supplements. These are a set of specifications that describe how Messages that conform to XML Messaging specifications are transported over a various transport protocols such as [HTTP], [SMTP], etc.

A compliant XML Messaging specification does not need to implement all the specifications highlighted above. An explanation of the how these relate is contained in section 6 XML Messaging Compliance Levels.

Currently only this document is fully defined and published.

1.2 History

The ideas in this specification are largely based on the ideas

contained within the Internet Open Trading Protocol [IOTP]. IOTP development started in early 1998 and is available as an Informational RFC (currently in RFC Editor Queue) from the IETF. Several implementations of IOTP have been developed.

1.3 Objectives of Document

The objectives of this document are to:

- define the terminology used by XML Messaging, and
- provide the reader with an overall understanding of XML Messaging so that they can better understand the relationships between this and the other XML Messaging Specifications identified above
- highlight the requirements that other XML Messaging specifications must cover.

It is recommended that this specification is read first.

1.4 Principles and Assumptions

The following principles and assumptions have been applied in developing this specification:

- [XML] will be used to define any data required to support XML Messaging
- XML Messaging shall support the exchanging of documents in any digital format
- the data used by XML Messaging will be defined using:
  - the W3C XML Schema language [XDSL], and
  - an [XML] Data Type Definition (DTD)
- Schema and DTD definitions will be placed in a repository such as those being developed by [XML.org].

1.5 Document Structure

This document contains the following sections:

- XML Messaging Definitions and Terminology. This defines the terminology used by XML Messaging

- Standard Transactions. This describes four standard Transactions that are built using XML messaging that are generally applicable:
  - Service Availability Inquiry
  - Transaction Status Inquiry
  - Quality of Service Negotiation
  - User/Server Authentication
- Restart, Recovery and Idempotency. This describes how XML Messaging

can be used to ensure the completion of a transaction and provide guaranteed once-only delivery

- Security Considerations. This describes some of the considerations around whether or not digital signatures should be used with XML Messaging

- XML Messaging Compliance Levels. Compliance with XML Messaging levels. This describes how the different parts of the XML Messaging specification may be used in combination to meet different needs.

- XML Messaging Examples. This illustrates the structure of a "Message" and illustrates its use with five example message flows

- References. This lists the other documents and standards used by XML Messaging

[Note] In several part of the specification "notes" such as this are placed to provide additional background, advice or guidance as an aid to understanding. They are non-normative parts of this specification.
[Note End]

1.6 Terminology

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

2. XML Messaging Definitions and Terminology

This section defines the main terminology used by XML Messaging. It is divided into four parts:

- Documents, Parties, Messages and Document Exchanges

- Services and Transactions,

- Lower Level Data Definitions that are used by the above, and

o Miscellaneous Definitions that don't fall into any of the above categories.

[Note] Definitions of words or phrases in this section that start with a capital letter (for example Document Exchange) cab usually be found defined elsewhere within this section.
[Note End]

2.1 Documents, Parties, Messages and Document Exchanges

2.1.1 Overview

This section describes how Parties, such as buyers and suppliers, customers and merchants, can exchange Documents contained in Messages in order to execute Services.

All the Documents and other data in a Message are contained within an outermost Message Envelope.

A Message can optionally include a Digital Signature so that:

o the identity of the party sending the message can be authenticated

o any changes to the message can be detected.

Services are requested by sending one or more Documents in a Request Message to a Party who then:

o processes the Request Message by carrying out a Service and

o generates a Response Message to indicate the result.

At a minimum a Document Exchange consists of a Request Message and a Response Message although there might be additional Exchange Messages between the Request and the Response Message.

Documents may also be sent as in a One Way message. In this case no business or application level response is provided.

The receipt of a Message may be acknowledged by sending a Message Acknowledgement back to the sender of the Message.

Error Messages are used to report permanent or transient problems or errors in a Message.

More detail is provided below.

2.1.2 A Document

A Document is any data that can be represented in a digital form.

Examples of Documents include:

o a set of XML Elements

o an XML Document

o an HTML Document

o a word processing file

o a Adobe Acrobat PDF file

o a binary file.

2.1.3 Party

A Party is a company, organization, individual or other entity that can generate, receive or authorize Documents and their associated Messages.

Examples of a Party include:

o a Merchant

o a Customer

o a Lawyer

o a Bank

A Party is also used to refer to systems or servers that are carrying out Services or other processes on behalf of a Party.

2.1.4 From, To and Authorizing Parties

FROM AND TO PARTIES

A Party that sends a Message is the "From" Party. A Party that is the intended recipient of a Message is the "To" Party. Every Message MUST

identify the "From" and the "To" Parties although Parties MAY identify themselves as anonymous.

Examples of a "From" Party include:

o a buyer that sends a Purchase Order to a supplier

o a supplier that sends an Invoice to a buyer.

Examples of a "To" Party include:

o the customer that receives a bank statement from their bank

o the travel agent that receives a booking for a flight

ANONYMOUS PARTIES

Anonymous parties MAY be used where the identity of the sender of a message is not required for the service associated with the message to be carried out. Examples of anonymous parties include:

o an individual that is querying availability of seats on a flight

o a company that is searching for the list prices of a product from a number of suppliers.

AUTHORIZING PARTIES

Messages MAY also contain one or more Authorizing Parties. The Authorizing Parties are Parties that authorize the "To" Party to act upon the Message that is received.

The Authorizing Party will frequently be the "From" party but this MAY NOT always be the case. If no Authorizing Parties are present in a Message then the "From" party is the Authorizing Party.

Examples of document exchanges where more than the "From" party are required to authorize an action include:

o a payment request a consumer makes to a bank to get a refund on a credit card payment. The authorizing parties are:
  - the consumer that is requesting the refund, and
  - the merchant that is instructing the bank to make the refund

o a transport firm that is being requested by a company to deliver goods that have been ordered from a supplier and payment made to a bank. The authorizing parties are:
  - the company that is requesting the delivery
  - the bank that authorizes that payment had been made
  - the supplier that specifies what goods should be delivered

[Note] In the above examples, the consumer/company is sending a message to one party to ask them to carry out a service on behalf of another party where the information that is being

sent is being passed through the consumer/company. Frequently Digital Signatures may be used to protect the information. See section 7.2.4, Transaction with Multiple Document Exchanges for an example of this can be used.
[Note End]

2.1.5 Message

A Message is data that is sent from one Party to another. A Message MUST consist of:

o a Message Header

o Message Routing Information

o zero or more Digital Signatures, and

o zero or more Documents

All the data in a Message is contained within a Message Envelope.

Examples of a Message include:

o a Purchase Order that is sent by a buyer to a supplier

o an Invoice that is sent by the supplier back to the buyer

- o a request to make a payment of $50 sent to a Credit Card acquirer
- o the authorization received from a Credit Card acquirer as a result of making a payment
- o Status Data indicating the success or failure of a Service
- o a multiple document message, e.g.
  - a design specification for a product including the Microsoft Word document and several JPEG files that illustrate the product's appearance
  - a batch of Purchase Orders that are being sent at the same time to the same destination

[Note] A Message need not contain any documents since the Message Header can contain all the data necessary to indicate the purpose of the Message.
[Note End]

2.1.6 Message Header

A Message Header contains the additional data that needs to be associated with a Document so that it can be sent to and successfully processed by a Party. The Message Header MUST contain:

- o Transaction Identity data to identify the set of Messages that are related to one another through one or more Document Exchanges
- o Message Identity data to enable the Message to be identified and referenced within the Transaction
- o Organization Data that describes:
  - who the Message is From
  - who the Message is sent To
- o Action Data to indicate the type of Service that is being sent the message and the intention behind sending it

Depending on the purpose of the message, a Message Header MAY also contain:

- o Status Data that describes the results of carrying out a Service.
- o a Message Manifest to identify the documents, other than the Message Header and Message Routing Information, that are contained within the same Message Envelope
- o Related Transaction Data that identifies other Transactions to which this Transaction is related

2.1.7 Message Routing Information

The Action Data and Organization Data together identify, logically, the Sender and Recipient of the Message, the type of Service that should process it and the intention behind sending the message.

This information is used to determine the [URL] to which the Message

should be physically sent. The rules used to determine the URL that is used for this purpose SHALL NOT form part of this specification.

The resultant URL MUST then be placed inside the Message Routing Information prior to sending the message to the URL that was determined.

The URL to which the Message is sent may not be the final destination of the Message and may be some "hub" process that forwards the Message to yet another location. Each "hop" along the chain of destinations results in another entry in the Message Routing Information.

The Message Routing Information may be digitally signed so that the intent of sending the message at the time and in the manner indicated can be authenticated.

[Note] Message Routing Information is held separately from other Documents in the Message, such as the Message Header, since:

- it would break any digital signature on those documents as Message Routing Information is extended each time the Message is forwarded to a new URL
- it makes it easier to separate the processing logic that determines the logical destination (e.g. the business application) from the physical destination. As a result business applications need not be aware of changes to network designs that result in changes to URLs that to which Messages are sent,
- if a particular URL is not accepting Messages for some reason, then an alternative URL may be used without invalidating any signatures (see section 4. Restart, Recovery and Idempotency)

The Message Routing Information may also be extended when a Message is routed beyond its apparent ultimate destination by processes that are not made publicly available.

[Note End]

2.1.8 Digital Signature

A Digital Signature is a cryptographic signature over data contained in a Message, or elsewhere that are addressable via [URI]s, that may permit:

- the identity of the sender of the message to be authenticated, and
- enables changes in the signed data to be detected

The XML Messaging specifications SHALL NOT specify any of the trust relationships or certificate authority issues associated with Digital Signatures. This is the responsibility of the designers of systems that use XML Messaging.

Use of Digital Signatures by XML Messaging is OPTIONAL.

Digital Signatures within XML Messaging are implemented using the IETF/W3C Digital Signature standard [XMLDSIG].

Examples of the use of Digital Signatures include:

- o demonstrating that a Request Message to initiate a Service is authorized
- o proving the authenticity and validity of an assertion contained in a document. For example, "these goods are guaranteed against defect for one year"

[Note] Signatures are optional since:
- o some Messages may not need any security
- o some Documents might contain their own Signatures that make the need for Signatures at the Message level unnecessary

Signatures are not contained in the Message Header since:

- o implementations of XML Messaging may require that Signatures sign the Message Header
- o it makes it easier for Documents to make use of message-level signatures without understanding the structure of a Message Header

[Note End]

2.1.9 Message Envelope

A Message Envelope is the outermost container for a Message. It MUST be either:

- o an XML Document with pre-defined XML tags, or
- o a multi-part MIME message

2.1.10 Request Message

A Request Message is a Message sent from one Party to a second Party with the intent that the second Party act upon the data in the Request Message by carrying out a Service.

A Request Message MAY also contain:

- o Status Data that describes the end process state of earlier Document Exchanges on which start of the Service being requested may depend

Examples of a Request Messages include:

- o requesting the processing of a new purchase order
- o requesting that a previous purchase order is changed
- o requesting a refund of a payment as a result of a problem
- o requesting the review of a legal document
- o requesting information on the services a business provides.

2.1.11 Response Message

A Response Message is a Message that is generated by the Party that received a Request Message. It is produced as a result of carrying out the requested Service. It is the last Message in a Document Exchange unless the Message contains errors.

Response Messages are sent back to the sender of the Request Message.

A Response Message MUST contain:

- a Reference to the immediately preceding Message (e.g. a Request Message) that the Response Message is responding to, and
- Status Data that describes the end process state of the Document Exchange.

Examples of Response Messages include:

- an acknowledgement of receipt of a Purchase Order
- an Invoice generated as a result of processing a Purchase Order
- a receipt for a payment that was made
- an opinion on a legal document that was received
- a description of the services provided by a business

[Note] There is a distinction between a Response Message and an Error Message (see section 2.1.18) below.
[Note End]

2.1.12 One Way Message

A One Way Message is a Message sent from one Party to another where a business level or application response to the message is not required.

Example uses of One Way Messages include:

- a newsletter that contains updated information about a company or group
- a Request for Quotation (RFQ) for the provision of a service or product

2.1.13 Document Exchange

A Document Exchange is a generic term for either a Simple Document Exchange or a Multiple Round Trip Document Exchange.

2.1.14 Simple Document Exchange

A Simple Document Exchange consists of:

o a Request Message sent from one party to a second party, and

o the Response Message that is returned as a result.

Examples of instances of a Simple Document Exchange include:

o a Purchase Order sent by a buyer to a seller and the acknowledgement from the seller of its receipt

o a Purchase Order sent by a buyer to a seller and the Invoice that is sent back as a result of fulfilling the order

o sending a document for review by a lawyer followed by the legal opinion that is sent back as a result

2.1.15 Multiple Round Trip Document Exchange

A Multiple Round Trip Document Exchange consists of:

o a Request Message sent from one party to a second party, followed by

o a series of Exchange Messages that are exchanged between the two parties until finally

o the second party generates and sends a Response Message back to the first party.

Examples of Multiple Round Trip Document Exchanges include:

o the exchange of messages required to make a payment using payment method protocols such as [SET] or [Mondex]

o the exchange of messages required to negotiate an agreement on terms and conditions

o the exchange of messages required to send a large document between two locations in several smaller parts.

2.1.16 Exchange Message

An Exchange Message is a Message that is sent between one Party and another after the sending of the initial Request Message and before the sending of the final Response Message.

An Exchange Message MUST also contain a Reference to the immediately preceding Message that the caused the Exchange Message to be generated.

Examples of Exchange Messages include:

o intermediate messages that are part of a Payment Protocol

o a counter offer to an offer made as part of a negotiation.

2.1.17 Acknowledgement Message

The sender of any Message, apart from an Acknowledgement Message, MAY request that receipt of the Message be acknowledged by recipient sending an Acknowledgement Message back to the Party that sent the original Message.

The purpose of an Acknowledgment Message is to provide positive confirmation that a Message has been received.

Acknowledgement Messages should be sent immediately upon receipt of the Message before the Message is processed and before any business or application level response (i.e. a Response Message or Exchange Message) is sent.

Acknowledgement Messages may be digitally signed to help prove their authenticity.

[Note] Acknowledgement Messages are particularly useful in an asynchronous exchange of Messages where there may be extended periods of time between the receipt of a Message and the generation of another business or application level Message in response.
[Note End]

2.1.18 Error Message

An Error Message is a Message that reports on a problem in an earlier Message that prevents the earlier Message from being processed in a normal way.

An Error Message MUST also contain Error Data to describe the problem that has been found.

Examples of an Error Message include:

- an Error Message reporting that an XML document was invalid or did not conform to its XML schema

- an Error Message reporting a Transient Error that the Server processing a Message is busy and therefore the original Message should be resent at a later point in time.

[Note] Error Messages report abnormal conditions which will require some type of unusual handling that might (or might not) permit recovery from the error. This is distinct from a Response Message that is used when a process completes normally.

However even though a process completes "normally" it might not be successful. For example, if a Purchase Order is sent to a supplier, the goods requested might be "out of stock".

This means that the request failed although the request was not in error and no Error Message should be produced.

[Note End]

2.2 Services, Service Types and Transactions

2.2.1 Overview

A Service is the implementation, by a Party, of a set of processes that conform to a particular Service Type.

Each Service Type has associated with it a set of Document types that it can accept as input and a set of Document types that it can produce as output.

The same Document type may be sent with different intentions. For example a purchase order might be either sent as either a "new" purchase order or as a "change" purchase order.

Sending a Document to a Service with a particular intention will result in an Document Exchange where documents are swapped in a sequence defined according to the rules defined in a Document Choreography.

The Document Choreographies and One Way Messages associated with a Service Type fully define that type of Service's interactions with the outside world.

The definition of a Service Type may also make use of Sub Services to implements it's functionality. Each Sub-Service is a Service in its own right. So, at the lowest level, all definitions of a Service Type consist of Document Exchanges.

The preceding description describes terms that define the rules associated with the sending or receiving of documents by a Service. The actual sending of real documents are called Transactions.

More detail is provided below.

2.2.2 Service and Service Type

A Service is carried out by a Party as a result of receiving a Request Message containing Documents sent with a particular intention.

The behavior of a Service in terms of the Documents it can accept or generate is defined by its Service Type.

Service Types may be defined by and be proprietary to an individual Party. However it is more likely that "standard" Service Types will

be defined to meet common requirements, for example, to submit a new Purchase Order or to make a payment.

Examples of Service Types include descriptions of:

- a Purchasing Service that enables a customer to purchase goods on-line
- an Order Processing Service that processes an Order and generates a response as a result
- a Payment Service that accepts a payment and provides a receipt
- a Fulfillment service that fulfills an order at the request of a Merchant.

2.2.3 Sub-Service

A Sub-Service is a Service that is executed at the request of and as part of another Service.

Examples of Sub-Services include:

- a payment service that occurs as part of a purchase
- a tax calculation service that calculates the tax due as part of an order processing service.

The Sub-Services in a Service will have dependencies between them. These dependencies MAY be:

- Serial. One Sub-Service MUST start only after the completion of another Sub-Service
- Alternative. One Sub-Service MAY be executed as an alternative to another
- Iterative Loop. A Sub-Service MAY be repeated a variable number of times
- Conditional. The execution of a Sub-Service is conditional on the state of another Service. This may be used in conjunction with Serial, Alternative and Iterative Loop dependencies.
- Parallel. A Sub-Service MAY execute in Parallel with another Service
- Concurrent. A Sub-Service MUST Execute at the same time as another Sub-Service.

An example of a simple Service with Sub-Services is a Purchase Service that consists of three Sub-Services:

- an Offer Service that conveys an Offer for sale of goods. This Sub-Service has no dependencies and therefore starts first
- a Payment Service that carries out the Payment which has a Serial dependency on the Offer Service
- a Delivery Service that delivers the Digital Goods, that has a Serial dependency on the Payment Service

2.2.4 Document Choreography

A Document Choreography is a description of the dependencies that control the sequence and choices in which Documents may be exchanged within a Document Exchange.

Document Choreographies MUST be either:

- Explicit, in that the Document Choreography is contained as data in one of the Messages in a Transaction, or
- Implicit, in that the Document Choreography is defined in some other document or specification, for example a protocol specification.

An example of a Document Choreography would be a new Purchase Order document choreography where:

- a new Purchase Order is sent to a supplier, then
- one of the following documents is returned as a result:
  - a Purchase Order Acknowledgement to indicate that the document has been received and is being processed, or
  - an Error Document to indicate there was a technical error in the Purchase Order, or
  - a Cancel Document to indicate that the Purchase Order will not be processed.

2.2.5 Transaction

A Transaction is an instance of the execution of a Service with a particular intention.

Examples of a Transaction include:

- a Purchase Transaction that buys an Company Report for $20. It consists of three Sub-Service instances:
  - an Offer Service instance to buy the Company Report for $20
  - a Payment Service instance that accepts a Payment for $20 using a credit card, and finally
  - a Delivery Service instance that delivers the Company Report as an HTML web page.
- a Buying Service that consists of the following Sub-Services:

  - three Price Negotiation Service instances that negotiate the price of a Photocopier
  - a Purchase Order Service instance that places the order for the Photocopier.

2.3 Lower Level Data Definitions

2.3.1 Overview

This sub-section contains definitions of the lower level data

elements referenced in the previous two sub-sections. It covers:

- Transaction Identity Data
- Message Identity Data
- Organization Data
- Status Data
- Action Data
- Error Data
- Cancel Data

2.3.2 Transaction Identity Data

Transaction Identity Data uniquely identifies and describes a Transaction.

Examples of the data contained in Transaction Identity Data include:

- an identifier that globally uniquely identifies the Transaction (a GUID)
- the version number of XML Messaging that is being used
- the date/time the Transaction started

2.3.3 Message Identity Data

Message Identity Data is data that enables a Message to be uniquely identified and referenced within a Transaction.

Examples of the data contained in Message Identity Data include:

- a reference that uniquely identifies the Message within a Transaction

- the date/time the message was created
- the software that generated the Message

2.3.4 Message Manifest

The Message Manifest contains references to the other documents, apart from the Message Routing Information document, that are contained within the same Message Envelope.

The purpose of the Message Manifest is to facilitate locating and validating that all required Documents contained within the Message Envelope are present.

Examples of the types of documents that might be referenced by a Message Manifest include:

o a Purchase Order

o a Purchase Order and a picture of the requested goods

o a Purchase Order and a digital signature

[Note] The Message Manifest does not contain a reference to the Message Routing Information since this information will typically be added after the rest of the documents contained in the Message Envelope have been created and therefore it's reference number will not be known.

The reference cannot be added later if the Message Header has been digitally signed since it would break the signature.

Finally, there may be benefit in adopting the Message Manifest developed by the IETF/W3C digital signature group [XMLDSIG] as the standard to be used here since essentially they serve the same purpose.
[Note End]

2.3.5 Related Transaction Data

Related Transaction Data optionally links one Transaction to other related Transactions. The linkage may be either by:

o specifying the GUID of the other transactions as specified on that Transactions, Transaction Identity Data or,

o using some other reference, outside the scope of XML Messaging by which the transaction may be identified

Examples of the use of Related Transaction Data include:

o referring to an earlier transaction where the terms and conditions associated with the transaction were negotiated

o raising a request to fix a problem associated with some earlier transaction that had occurred

o making a query on the status of another transaction.

2.3.6 Organization Data

Organization Data is data that describes and identifies a Party.

The purpose of Organization Data is to provide information:

o to enable the identity of the Organization to be determined

o that allows the Organization to be contacted when and if necessary

o to determine who a Message is from and who it is sent to.

Examples of information contained within Organization Data include:

- Name and Address
- Individuals and contacts
- Reference numbers, e.g. [DUNS] numbers

[Note] The amount of information provided in Organization Data can vary from:
- a simple reference number, through
- a comprehensive address, for example to describe a Supplier who is selling tangible goods, to
- minimal or none, for example if a Consumer is buying a page of digital information for 2 cents.

[Note End]

2.3.7 Action Data

Action Data indicates the type of Service that is being sent a Message and the intention behind sending it.

Examples of information contained within Action Data include:

- the identifier of the Service that should process the Message
- the intention in sending the Message, for example to request a service
- whether or not the Action is for test purposes only.

2.3.8 Status Data

Status Data provides information on the current state of a Transaction. It MUST be sufficient, on its own, to indicate success or failure of the Transaction.

Examples of the information contained within Status Data include:

- a code to indicate success, failure and various other states
- a reference to the Request Message that initiated the Transaction
- a reference to the Organization Data that carried out the Service

[Note] A standard definition of the results of a Transaction enables that information to be used to authorize the start of a later Service within a Document Choreography without providing details of the first Service to the Party that is carrying out the later Service.

As a result privacy is improved in that a Party is only provided with the information they need, to carry out a requested service.

[Note End]

2.3.9 Error Data

Error Data contains data that describes an Error contained in a Message. It MUST contain:

o a reference to the Message(s) that are in Error

If the error is not a Transient Error then the Error Data MUST also contain:

o a reference to the part(s) of the Message that are in Error

2.3.10 Cancel Data

Cancel Data sent by one party indicates that to the other Party that the remainder of the transaction will not be carried out. Cancel Data MAY be sent:

o by the Party that generated a Request Message to the Party that is acting on the message - to indicate that the requested action should not be completed

o by the Party that received a Request Message back to the party that generated the Request Message to indicate that the Request Message was unacceptable in some way and therefore no Response Message will be generated.

[Note] One of the main uses of sending a Message containing Cancel Data is to prevent the sender of a Request Message going into "recovery mode" since no Response Message was received within the expected timeframe (see 4.2 Recovering from failed Message delivery).

Also note that:
o sending a Message containing Cancel Data will fail if the Response Message has already been sent
o it may not be possible (or allowable) to cancel the processing of some Request Messages after they have been sent.

[Note End]

2.4 Miscellaneous Definitions

This section covers a number of miscellaneous definitions used earlier in this section. It covers:

o Document Encoding

o Audit Trails

o Transport Mechanism

o Identical Elements

2.4.1 Document Encoding

Some Documents may exist either in their native form or transformed

into a directly equivalent form through the use of a Document Encoding.

Examples of a Document Encoding include:

o [BASE64] encoding

o [MIME] encoding.

2.4.2 Audit Trail

An Audit Trail is a record of a Transaction that allows the processing carried out by a Transaction to be traced.

There are three mechanisms in XML Messaging that support the production of audit trails:

o data on each Message Identity Information that references the earlier Message that is being responded to

o Status Data from an earlier Message that shows how the start of one Document Exchange is dependent on an earlier one, and

o digital signatures on the above two sets of information to make the audit trail non-refutable.

See the XML Messaging Examples below for examples of an audit trail.

2.4.3 Transport Mechanism

A Transport Mechanism is a protocol, such as [HTTP] or [SMTP], that is used to physically transport a Message between two Parties.

XML Messaging is designed so that any Transport Mechanism may be used.

[Note] Although [HTTP] and [SMTP] are two likely transport mechanisms, literally any transport mechanism is possible, including:
o sending a Message as a fax
o putting the Message on a floppy disk and sending it in the post
[Note End]

2.4.4 Identical Elements

There are a number of instances where it is necessary to compare XML elements from one Message with elements from another to determine if they are the same. For example the way in which the Messages that belong to the same Transaction are identified is that they have "identical" Transaction Identification Data elements.

However some Transport Mechanisms can introduce additional characters into documents which means that a simple character-by-character comparison of elements will not provide reliable results.

Therefore, in XML Messaging, identification of identical elements MUST be determined by checking that their [DOM-HASH] values are identical.

3. Standard Transactions

The specification for XML Messaging defines four "standard" transactions:

- Service Availability Inquiry, that checks whether or not a Service is up-and-running
- Transaction Status Inquiry, that checks on the status of a transaction,
- User/Server Authentication, that enables one Party to authenticate another, and
- Protocol Negotiation, that allows two parties to agree which choreography and documents they will use to carry out a service.

The Service Availability and Transaction Status Inquiry transactions MAY be used to assist in re-start and recovery (see section 4).

Each of these are described in more detail below.

3.1 Service Availability Inquiry

In outline a Service Availability transaction consists of a Simple Document Exchange:

- a Service Availability Request document, sent by one Party to another, and
- a Service Availability Response document, that is sent in return

The Service Availability Response indicates whether or not the Service is available and provides basic information on some of properties of the Service such as scheduled "down time" and anticipated Response Times.

Service Availability Requests MAY be signed. System operators can use this to decide whether or not to generate a Service Availability

Response.

It is RECOMMENDED that a Party that is conducting Transactions with other Parties periodically sends Service Availability Requests to those other Parties to make sure that they are up and running.

3.2 Transaction Status Inquiry

The Transaction Status Inquiry consists of a Simple Document Exchange:

- a Transaction Status Inquiry Request document, that specifies the globally unique identifier of a Transaction, and
- a Transaction Status Inquiry Response document, that indicates the current state of the transaction. The state can include:
  - Not Known
  - Not Yet Started
  - In Progress
  - Completed Ok
  - Failed
  - Canceled
  - Process Error

The Transaction Status Inquiry document also indicates the identifier of the last Message that was received. This is used in recovering from failed message delivery (see section 4.2).

Transaction Status Inquiry Request documents MAY be digitally signed. This means that system operators can use this to determine if an inquiry is valid.

Transaction Status Inquiry Response documents MAY also be digitally signed. This means that the recipient of the Response Message can check its authenticity.

3.3 User/Server Authentication

The User/Server Authentication Transaction enables one Party to authenticate another. It uses the following Document Exchange:

- Authentication Request document. This is sent to the Party to be authenticated. It contains:
  - "challenge data" to which the Party being authenticated should respond, and
  - the authentication method that should be used, this can vary from providing a simple password, providing a digital signature or using some proprietary authentication protocol
- Authentication Response document. This is sent by the Party being authenticated to the Party requesting authentication. It contains the results of the processing the authentication data

Authentication Request documents may optionally be digitally signed. This MAY be used by the user/server being authenticated to determine whether or not the Authentication Request is authorized.

3.4 Quality of Service Negotiation

The same service may be implemented in a variety of ways and with differing levels or timeliness of response and using different transport protocols. The Quality of Service Negotiation transaction enables two parties to agree which, of the possible alternatives, may be used.

Two of the uses of Quality of Service Negotiation are:

- to determine the transport protocols that the Service supports
- to determine whether the Service will be implemented synchronously or asynchronously
- to agree the various "timeouts" used in recovering from failed message delivery (see section 4.2).

It consists of the following Document Exchange:

- Quality of Service Negotiation Request document. This is sent by the Party that wants to discover information about the Service. It contains a prioritized list that describes:
  - the protocols supported (e.g. HTTP, SMTP, etc) - whether messages are exchanged synchronously or asynchronously
  - the preferred time after which "recovery from failed message delivery processing" would start.
- Quality of Service Negotiation Response document. This is returned by the Party that received the Request. It contains, for example:
  - the selected protocol (if any),
  - the preferred time after which recovery from failed message delivery processing should start,
  - whether or not the data is exchanged synchronously
  - the validity of the Service Negotiation Response document
  - details of any planned outages or non-availability of the Service.

A single Quality of Service Negotiation transaction may not result in a successful response. In this case the sender of the original Request Message, may vary the content of the Quality of Service Negotiation Request document in order to find a combination that is acceptable.

Implementers may want to cache the results of a Quality of Service Negotiation Response document for later use

It is not necessary for a Quality Of Service Negotiation to be carried out before sending messages between two parties. However its use should maximize the probability of any particular transaction completing in a normal way.

4. Restart, Recovery and Idempotency

Restart and Recovery in XML Messaging provides a standard approach to recovering from the failure of a Message to get through to its intended recipient.

It uses the Service Availability Transaction (see section 3.1) and the Transaction Status Inquiry Transaction (see section 3.2) to determine what the current situation is and therefore what action should be taken to recover.

When to start the restart/recovery process will vary depending on the transport protocol and characteristics of the service. The Quality of Service Negotiation transaction (see section 3.4) MAY be used to determine when this should be.

The following is an overview. More detail is provided in [XMLMSG-RM].

4.1 Idempotency

Duplicate Messages can be received by a Service for a number of reasons:

- the Transport Protocol sometimes causes duplicate Messages to be delivered
- the sender of the Message sent the message twice. Either by accident, for example there is a bug in the Sender's software; or by design, for example no response was received to the first message so the message was resent.

In XML Messaging restart and recovery, it is assumed that if a Recipient receives an "identical" duplicate message (see section 2.4.4), then:

- the duplicate message is not processed
- the message that was generated in response to the original message is resent.

4.2 Recovering from failed Message delivery

Prior to sending a Message (either a Request Message or an Exchange Message) to another Party the timeframe in which recovery/restart should commence should be negotiated using a Quality of Service Negotiation Transaction (see section 3.4).

Once the Message is sent, the Sender of the Message will only discover a failure in delivery by realizing that the Message expected in response has not been received within the expected time frame.

The Message expected in response may be:

o an Acknowledgement Message (see 2.1.17 Acknowledgement Message), or

o a Message that is a business or application level response to the original Message, this can be:
  - a Response Message
  - an Exchange Message
  - an Error Message, or
  - a Cancel Message

However the Sender does not know why no Message was received in response. The Sender may also not want to re-send their Message without knowing the results of sending the earlier message. They therefore need to find out the current situation so that they can work out what to do to recover.

The following approach is RECOMMENDED as a guideline:

1)Carry out a Transaction Status Inquiry for the Transaction

2)If the result of the Inquiry is "Not Known", then re-send the original Message as the Message did not get through on the first try

3)If the result of the Inquiry is "In Progress" then wait a further period of time, as a Message in response should be received in due course once its processing is complete. If no Message in response is received after a period of time then repeat from step 1.

4)If the result of the Inquiry is any other Transaction Status (e.g. "Completed Ok", "Failed", etc.) then the first Message was received and processed but the Message in Response did not get through. In this case re-send the first Message which should result in the re-sending of the previously generated Message in Response. If the Message in Response does not arrive after the expected period of time then repeat from step 1.

5)If no response to the Transaction Status Inquiry is received, then it is possible that either the Transaction Status Inquiry transaction failed, or that the Service that should respond to the Inquiry is not working. In this case carry out a Service Availability Transaction for the Service.

6)If the Service Availability Transaction does not work then in all probability the Service is not working. In this case either:

  a) wait some period of time and retry the Service Availability Transaction until it does respond. Once the Service Availability Transaction works then repeat from step 1 to determine what recovery action to take, or

  b) choose an alternative URL to send the Message to (if one is available) and send the Message to that URL then wait for a response. In this case the failure to send the message to the

original destination should be recorded in the Message Routing Information.

7)If, after a number of re-tries, there is still no response from sending a Service Availability Request, and there are no alternative URLs that can be used then carry out whatever action is appropriate such as notify an operator for human intervention and carry out no further recovery actions until requested by the operator.

5. Security Considerations

This section considers, from an IETF perspective how XML Messaging addresses security. The section covers:

o digital signatures

o message authenticity, and

o data privacy.

5.1 Digital Signatures

The data contained within XML Messages can be digitally signed for the usual reasons such as:

o authenticating the identity of the sender of a message,

o enables changes in the signed data to be detected,

o providing a mechanism to support non-repudiation of a request for or response to the execution of a Service.

However, the scope of this specification SHALL be limited to defining how digital signatures can be used by XML Messaging and excludes the trust relationships that use of signatures may apply.

5.1.1 Determining whether to use digital signatures

The use of digital signatures within XML Messaging is entirely optional. XML Messaging can work successfully either with or without the use of digital signatures.

Ultimately it is up to the Parties involved to decide whether the XML Messages they use will include signatures. For example if a Sender of a Message omits a Digital Signature then the recipient of the Message will need to decide whether carrying out the requested Service without signatures is an acceptable risk. If senders of Request Messages discover that requests without signatures are not being accepted, then they will either:

o start using signatures,

o find a method of working which does not need signatures, or

o accept a lower volume and value of transactions.

5.1.2 Reasons for using Digital Signatures

A non-exhaustive list of the reasons why digital signatures might be used are described below.

A MERCHANT WANTS TO DEMONSTRATE THAT THEY CAN BE TRUSTED

If, for example, a merchant generates an offer to carry out a trade and includes it in an XML Message with a Signature that uses a certificate from a trusted third party, known to the Consumer, then the Consumer can check the signature and certificate and so more reasonably rely on the offer being from the actual organization the merchant claims to be.

In this case signatures using asymmetric (PKI) cryptography are likely to be required since the Merchant would not want other organizations to be able to generate an equivalent signature.

A MERCHANT WANTS TO GENERATE A RELIABLE RECORD OF A TRANSACTION

For example, with appropriate trust hierarchies, digital signatures could be checked by the Consumer to determine:

- o if a record of a purchase will be accepted by tax authorities as a valid record of a transaction, or
- o if some warranty, for example from a "Better Business Bureau" or similar was being provided.

A PARTY NEEDS TO KNOW THAT A REQUEST MESSAGE IS UNALTERED AND AUTHORISED

The data that constitutes a Request Message may have been provided by someone other than the sender of the Message. For example, in [IOTP], details of how much to pay is sent to the Consumer by a Merchant and then forwarded to the Payment Handler that is to accept the payment in a payment request message.

If the request is not signed, the Consumer could change the amount due by, for example, removing a digit.

If the Payment Handler has no access to the original payment information provided by the Merchant, then, without signatures, the Payment Handler cannot be sure that the data has not been altered.

Similarly, if the payment information is not digitally signed, the Payment Handler cannot be sure who is the Merchant that is requesting the payment.

A PARTY WANTS TO PROVIDE A NON-REFUTABLE RECORD OF THE COMPLETION STATUS OF A SERVICE

If a Response Message is digitally signed, then the recipient of the Response Message can later use the data in the Response Message to

prove that the service was carried out. This could be used, for example, to enable a customer to claim a refund of a purchase if required.

5.1.3 Reasons to not use Signatures

A non-exhaustive list of the reasons why digital signatures might not be used follows.

A SINGLE PARTY IS CARRYING OUT ALL SERVICES

One of the reasons for using digital signatures is so that one Party can determine if data has been changed by an intermediate Party that is forwarding the data.

However if the Party that needs to check for authenticity has access to all the necessary data, then it might be possible to compare, for

example, the forwarded information with the original.

Access to the data necessary could be realized by, for example, one Party carrying out all the different roles on the same system, or where the roles are carried out on different systems but the systems can communicate in some way.

THE PROCESSING COST OF THE CRYPTOGRAPHY IS TOO HIGH

If a payment is being made of only a few cents, the cost of carrying out all the cryptography associated with generating and checking digital signatures on a request to make the payment or a receipt contained in a payment response might make the whole transaction uneconomic.

ANYONE CAN REQUEST THE SERVICE

The service that is being requested, for example to inquire on the public list price of a product, may be open to anyone. In this case, there is little benefit in signing the request as it is likely that a response will always be generated.

5.2 Message Authenticity

XML Messaging provides two mechanisms by which the authenticity of the Parties involved may be assured:

- digital signatures
- user/server authentication

Digital signatures contained in Messages MAY be checked to ensure that the sender of the document is who they appear to be.

User/server authentication (see section 3.3) may be used to authenticate the sender of or recipient for any message. This can occur either:

- before sending a message to authenticate a recipient, or
- after receiving a message to authenticate its sender.

In addition transport level methods MAY be used to determine the authenticity of messages. For example:

- by checking the certificates associated with the use of secure channels such as [SSL/TLS], or
- using a communication method (e.g. a telephone line) that is dedicated to the two parties and is thought sufficiently trustworthy.

Transport Level authenticity methods SHALL NOT be defined by this specification. A secure transport protocol MAY also remove the need to use digital signatures.

5.3 Data Privacy

Privacy of information is provided by sending Messages using a secure channel such as [SSL/TLS]. Use of a secure channel with XML Messaging is OPTIONAL.

Any elements within documents that are transmitted using XML Messaging MAY be encrypted. However, the procedure used to encrypt XML elements or documents SHALL NOT be defined within this specification.

6. XML Messaging Compliance Levels

The purpose of this section is to indicate the different ways in which the various parts of the XML Messaging Specification may be used together in a compliant way.

XML Messaging Specification is split into several parts for a number of reasons:

- so that implementers of XML Messaging may select the just the parts of XML Messaging that they need, for example, some implementations may not need to use digital signatures,
- to facilitate the development and evolution of individual sections in parallel,
- to encourage re-use of the XML Messaging Specifications by other protocol and standard developers.

The following examples illustrate ways in which the different parts of XML Messaging may be combined.

6.1 Data Element Compliance

Data Element Compliance is the lowest level of compliance which simply requires use of some of the XML element definitions contained in the Common XML Messaging Elements specification [XMLMSG-CME]. The message elements MUST be used with the semantics as defined in the specification.

6.2 Message Level Compliance

Message Level Compliance uses all the data elements defined in the Common XML Messaging Elements Specification [XMLMSG-CME] to construct XML Messages that can be sent between two Parties. However use of XML Messages in a Response-Request based Document Exchange is not required.

6.3 Document Exchange and Transaction Level Compliance

Document Exchange Compliance exchanges messages in the standard ways defined within Document Exchange and Transactions [XMLMSG-DET]. This uses XML Messages to support Transactions consisting of either:

o a Simple Document Exchange

o a Multiple Round Trip Document Exchange, or

o a One Way Message

This MUST be used with:

o Message Level Compliance (see section 6.2), and

o a transaction protocol (e.g. HTTP)as defined in the protocol's Transaction Protocol Supplement (see section o).

6.4 Reliable Messaging Compliance

Reliable Messaging Compliance uses the processes and procedures defined in Reliable Messaging [XMLMSG-RM] to achieve reliable, robust and resilient message delivery.

This MUST be used with Document Exchange and Transaction Compliance (see section 6.3) and MAY be used with any combination of:

o Secure Messaging Compliance (see section 6.5),

o Document Choreography Compliance (see section 6.6).

6.5 Secure Messaging Compliance

Secure Messaging Compliance uses the processes and procedures defined in Secure Messaging [XMLMSG-SM] to ensure the tamper resistant and authenticated exchange of messages.

This MUST be used with Document Exchange and Transaction Compliance (see section 6.3) and MAY be used with any combination of:

o Reliable Messaging Compliance (see section 6.4),

o Document Choreography Compliance (see section 6.6).

6.6 Document Choreography Compliance

Document Choreography Compliance uses the processes and procedures defined in Document Choreography Definitions [XMLMSG-DCD] that defines how the sequence in which documents are exchanged may be defined.

This MUST be used with Document Exchange and Transaction Compliance (see section 6.3) and MAY be used with any combination of:

o Reliable Messaging Compliance (see section 6.4),

o Secure Messaging Compliance (see section 6.5).

6.7 Transport Protocol Supplement Compliance

XML Messages may be transported over a variety of transport protocols such as [HTTP] or [SMTP]. How this is done for a particular transport protocol is defined in a Transport Protocol Supplement for the protocol.

Transport Protocol Supplement Compliance means that an implementation conforms with the Transport Protocol Supplement specification.

6.8 Common Document Exchange and Transaction Compliance

Common Document Exchange Compliance means that an implementation supports one or more of the document exchanges defined as standard.

Examples of candidate standard document exchanges/transactions include:

o a Transaction Status Inquiry, that allows the current status of any transaction to be discovered,

o a Large Message Delivery Transaction, that supports the delivery of large messages/documents by splitting them up into several smaller parts,

o a Message Trace Inquiry that enables the sender of a message to discover where the message has been forwarded to,

o a Service Functionality Inquiry that enables the functions that a Service supports to be discovered such as Reliable Messaging, Security and Signatures, Transport Protocols supported, Large Message Delivery capability, etc

These common document exchanges/transactions MUST be used with Document Exchange and Transaction Compliance (see section 6.3) and MAY be used with any combination of:

o Reliable Messaging Compliance (see section 6.4),

o Secure Messaging Compliance (see section 6.5),

o Document Choreography Compliance (see section 6.6).

7. XML Messaging Examples

This section is in two parts:

o an illustration of the structure of a Message that conforms to the ideas in this specification, and

o a series of examples that illustrate how XML Messaging could be used. The examples provided are:
  - Simple Document Exchange
  - Simple Document Exchange with Errors
  - Canceling a Transaction
  - Transaction with Multiple Document Exchanges
  - Relating Two Transactions

7.1 XML Message Structure

In outline, an XML Message has the structure shown in the figure below. Note that spaces have been included in the "element" names for ease of reading.

<MessageEnvelope ...>	o Outer wrapper can be an XML Document or a Multipart MIME message.
<MessageHeader ...>	o Message Header contains additional data required to send the document(s) successfully. Content depends on whether it is a Request Message, Exchange Message, Response Message or an Error Message.
<MessageType ...>	o Required in every Message. It indicates the type of the Message. The different types of Message are: Request, Response, Exchange, Acknowledgement, OneWay, Cancel, and Error.
<TransactionIdentityData ...> ...	o Required in every Message. It identifies a set of related

```
      </TransactionIdentityData>       Messages.
      <MessageIdentityData ...>      o Required in every Message. It
        ...                            identitifies and describes an
      </MessageIdentityData>           individual message within a
                                       Transaction
      <From ...>                     o Required in every Message. It
        ...                            identifies, logically, the sender
      </From>                          of the message. The sender may be
                                       anonymous.
      <To ...>                       o Required in every Message. It
        ...                            identifies, logically, the
```



```
      </To>                            recipient of the Message. The
                                       recipient may be anonymous in
                                       which case the recipient is
                                       identified at the Transport
                                       Protocol level.
      <AuthorizationData ...>        o Optional. Contains information
        ...                            that identifies who is
      </AuthorizationData>             authorizing that the recipient of
                                       the Message should act on the
                                       Message.
      <ServiceType ... >             o Required in every Message. It
        ...                            indicates the type of Service to
      </ServiceType>                   which the Message is associated.
      <Intent ... >                    Required in every Message. It
        ...                            indicates the reason why Message
      </Intent>                        is being sent to the Service.
      <Status ... >                    On Response, Acknowledgement and
        ...                            Error Messages it rovides an
      </Status>                        indication of the current state
                                       of the Document Exchange. On
                                       Request Messages it is used to
                                       indicate the end state of a
                                       preceding Document Exchange.
      <MessageManifest ...>          o Required in every Message that
        ...                            contains more than just a Message
      </MessageManifest>               Header and Message Routing
                                       Information. Identifies what
                                       other documents/signatures were
                                       sent with the message.
      <RelatedTransactionData...>    o Optionally links one Transaction
        ...                            to some earlier Transaction
      </RelatedTransactionData>
    </MessageHeader>
    <MessageRoutingInfo ...>         o Required within every message.
      ...                              Maps the logical destination to
    </MessageRoutingInfo>              the physical destination (e.g. a
                                       URL). Contains a history of the
                                       route taken by the message in
                                       order to reach its final
                                       destination.
    <Signatures ...>                 o Zero or more. A Digital signature
      ...                              can sign any data in the Message,
    </Signatures>                    the Transaction, or elsewhere,
                                       for example on the web.
                                   o The actual document(s) that are
    ... DOCUMENT(S) ...                to be sent to a party. Documents
                                       may be encoded for transportation
                                       for example using [Base64].
```

```
 </MessageEnvelope>
```

Figure 1 XML Messaging, Message Structure

If the Message Envelope is implemented using multi-part MIME, then the Message Header, Signatures and the "Documents" would each be contained in separate multi-part MIME parts.

If the Message Envelope is implemented using an XML-Document, then non-XML "documents" would be wrapped in thin XML wrapper so that they can be uniquely identified by the Message Manifest.

7.2 XML Messaging Examples

7.2.1 Simple Document Exchange

At its simplest, a Transaction consists of just one Service that uses just one Simple Document Exchange that consists of:

- a Request Message sent from one party to a second party, and
- the Response Message that is returned as a result.

The figure below provides an example of what these messages could contain and how they could be processed.

```
 Party 1
     |  Party 2
STEP |     |
 1.          Party 1 generates the documents that need to be sent,
             places them in an XML Message together with a Message
             Header and (optional) digital signature and sends it to
             the Party 2.

     1 --> 2 Request Message:
             o Message Header
               - Message Type (Request)
               - Transaction Identity Data
               - Message Identity Data
               - From (Party 1)
               - To (Party 2)
               - Service Type
               - Intent
               - Authorization Data
               - Message Manifest
             o Message Routing Info
             o Signature(s) (Optional)
             o Document(s)

 2.          Party 2 that received the Request Message carries out the
             following processes:
             o checks the "To" data to check the Message intended for
               them;
             o checks the Service Type and Intent to make that they
```

can carry out the request;

- o if required, checks the "From" data to make sure that the sender is known
- o checks the Authorization Data (if present) to make sure the action is authorized (note some requests may not need authorization);
- o checks the Signature, if present, to make sure the data has not changed and the sender can be identified, then
- o if everything is OK, checks the Document(s) for errors, then
- o if no errors in the Document(s) then carries out the requested action using the data contained in the Document(s)
- o once the action is complete, then generates a Response Message and sends it back to Party 1.

1 <-- 2 Response Message:

- o Message Header:
  - Message Type (Response)
  - Transaction Identity Data
  - Message Identity Data
  - From (Party 2)
  - To (Party 1)
  - Service Type
  - Intent
  - Message Manifest
  - Status Data
- o Message Routing Info
- o Signature(s) (Optional);
- o Document(s)

3. Party 1 that has received the Response Message then carries out the following processes:

- o Checks the Transaction Identity Data and Message Identity Data to make sure the Response Message refers to the Transaction/Request Message that they sent earlier
- o checks the Signature, if present, to make sure the data has not changed and is the sender of the message can be authenticated
- o checks the Status Data to determine that the Service was carried out by the anticipated Organization and whether the service succeeded or failed
- o if everything is OK, then checks the Document(s) for Errors,
- o if everything is still OK, then carries out whatever processing of the Document(s) is necessary.

Figure 2 Simple Document Exchange Message Flow

[Note] In the above example the sequence of the processing by the described for Party 1 or Party 2 is indicative of the processing required. Sequences that provide the same result are equally acceptable.

[Note End]

7.2.2 Simple Document Exchange with Errors

In the previous example, it was assumed that no errors were found. If errors are found then a slightly different message flow would occur. The example below builds on the previous example in Figure 2 Simple Document Exchange Message Flow.

```
 Party 1
     |  Party 2
STEP |     |
 1.          Party 1 generates the documents that need to be sent,
             places them in an XML Message together with a Message
             Header and (optional) digital signature and sends it to
             the Recipient.

     1 --> 2 Request Message:
             o Message Header
               - Message Type (Request)
               - Transaction Identity Data
               - Message Identity Data
               - From (Party 1)
               - To (Party 2)
               - Service Type
               - Intent
               - Message Manifest
             o Message Routing Info
             o Signature(s) (Optional)
             o Document(s)

 2.          Party 2 that receives the Request Message detects an error
             and so sends an Error Message back to Party 1.

     1 <-- 2 Error Message:
             o Message Header
               - Message Type (Error)
               - Transaction Identity Data
               - Message Identity Data
               - From (Party 2)
               - To (Party 1)
               - Service Type
               - Intent
               - Message Manifest
             o Message Routing Info
             o Signature(s) (Optional)
             o Document(s)
               - Error Data

 3.          Party 1 that received the Error Message:
             o Checks the Transaction Identity Data and Message
               Identity Data to make sure the Message refers to the
               Transaction/Request Message that they sent
```



```
             o checks the Signature, if present, to make sure the data
               has not changed
             o determines that there was an Error and takes the
```

appropriate action.

Figure 3 Document Exchange with Request Message Errors

Errors can also occur in a Response Message. In this case the Message flow would be as below.

```
 Party 1
     |  Party 2
STEP |     |
 1.          Party 1 generates the documents that need to be sent,
             places them in an XML Message together with a Message
             Header and (optional) digital signature and sends it to
             Party 2.

     1 --> 2 Request Message:
             o Message Header
               - Message Type (Request)
               - Transaction Identity Data
               - Message Identity Data
               - From (Party 1)
               - To (Party 2)
               - Service Type
               - Intent
               - Message Manifest
             o Message Routing Info
             o Signature(s) (Optional)
             o Document(s)

 2.          Party 2 that receives the Request Message successfully
             processes the Request Message and generates and sends a
             Response Message back to Party 1.

     1 <-- 2 Response Message:
             o Message Header:
               - Message Type (Response)
               - Transaction Identity Data
               - Message Identity Data
               - From (Party 2)
               - To (Party 1)
               - Service Type
               - Intent
               - Message Manifest
               - Status Data
             o Message Routing Info
             o Signature(s) (Optional);
             o Document(s)

 3.          Party 1 that has received the Response Message identifies
             an error in the some part of the Response Message and
```



```
             therefore sends an Error Message back to the Sender of the
             Response Message.

     1 --> 2 Error Message:
             o Message Header
               - Message Type (Error)
               - Transaction Identity Data
               - Message Identity Data
```

```
                - From (Party 1)
                - To (Party 2)
                - Service Type
                - Intent (Error)
                - Message Manifest
              o Message Routing Info
              o Signature(s) (Optional)
              o Document(s) (Optional)
                - Error Data

 4.           Party 2 that received the Error Message:
              o Checks the Transaction Identity Data and Message
                Identity Data to make sure the Message refers to a
                Transaction/Request Message that they are aware of
              o checks the Signature, if present, to make sure the data
                has not changed
              o determines that there was an Error in the Response
                Message they had sent and takes the appropriate action.
```

Figure 4 Document Exchange with Response Message Errors

7.2.3 Canceling a Transaction

Transaction cancellation occurs in a similar way to generating errors. For example if a Recipient of a Request Message wants to cancel a transaction they would do this as illustrated in the figure below.

```
 Party 1
     |  Party 2
STEP |     |
 1.          Party 1 generates the documents that need to be sent,
             places them in an XML Message together with a Message
             Header and (optional) digital signature and sends it to
             Party 2.

     1 --> 2 Request Message:
             o Message Header
               - Message Type (Request)
               - Transaction Identity Data
               - Message Identity Data
               - From (Party 1)
               - To (Party 2)
               - Service Type
```



```
               - Intent
               - Message Manifest
             o Message Routing Info (Optional)
             o Signature(s) (Optional)
             o Document(s)

 2.          Party 2 that received the Request Message checks the
             message but decides they want to cancel the Transaction
             for some reason and so sends a Cancel Message back to the
             Sender of the Request Message.

     1 <-- 2 Cancel Message:
             o Message Header
```

```
                - Message Type (Cancel)
                - Transaction Identity Data
                - Message Identity Data
                - From (Party 2)
                - To (Party 1)
                - Service Type
                - Intent (Cancel)
                - Message Manifest (Optional)
                 o Message Routing Info (Optional)
                 o Signature(s) (Optional)
                 o Document(s) (Optional)
```

7.2.4 Transaction with Multiple Document Exchanges

Transactions can also consist of multiple Document Exchanges as illustrated in the following example that contains four Document Exchanges that are part of three Services. The Services, the associated Intent and the resultant Document Exchanges are as follows

```
 o Service: Supplier Order Processing
   - Intent: Stock Availability Check
       Document: Stock Availability Request
       Document: Stock Availability Response
   - Intent: New Purchase Order
       Document: Purchase Order
       Document: Invoice
       Document: Payment Brand List

 o Service: Payment Service
   - Intent: Make Payment
       Document: Payment Request
       Document: Payment Instrument
       Document: Payment Response

 o Service: Delivery Goods
   - Intent: Request Delivery
       Document: Delivery Request
       Document: Delivery Response
```


Note that this example also illustrates how digital signatures can be used to provide an Audit Trail and authorize the execution of each Service.

More details are provided below together with explanations on how signatures are used to link each Document Exchange instance together.

```
 Buyer Supplier /
     |  Payment Handler
STEP |     |
 1.          Buyer generates a Stock Availability Request, digitally
             signs it, and sends it to the Supplier to check for
             availability.

     B --> S Request Message (Stock Availability Request):
             o Message Header
               - Message Type (Request)
               - Transaction Identity Data
```

```
                - Message Identity Data
                - From (Buyer)
                - To (Supplier)
                - Service Type (Supplier Order Processing)
                - Intent (Stock Availability Check)
                - Authorization Data:
                    Buyer Data
                    ref. to Signature1
                - Message Manifest
              o Message Routing Info
              o Signatures
                - Signature1, Signs:
                    Message Header
                    Stock Availability Request
              o Documents
                - Stock Availability Request
```

2. The Supplier checks the Stock Availability Request (see example earlier for what is involved in checking a request) and generates a Stock Availability Response to send back to the Buyer. The response is digitally signed.

```
    B <-- S Response Message (Stock Availability Response):
              o Message Header
                - Message Type (Response)
                - Transaction Identity Data
                - Message Identity Data
                - From (Supplier)
                - To (Buyer)
                - Service Type (Supplier Order Processing)
                - Intent (Stock Availability Check)
                - Message Manifest
                - Status Data (Stock Availability Check)
              o Message Routing Info
              o Signatures
                - Signature2, Signs:
```



```
                    Message Header
                    Stock Availability Response
                    Signature1 in Stock Availability Request Message
              o Document
                - Stock Availability Response
```

3. The Buyer checks the Stock Availability Response Message and finds it OK. As a result generates a Purchase Order Document and sends it to the Supplier. The Authorization Data includes Status Data on the Stock Availability, to indicate that the Purchase Order is linked to the earlier Stock Availability Check Document Exchange.

```
    B --> S Request Message (Purchase Order):
              o Message Header
                - Message Type (Request)
                - Transaction Identity Data
                - Message Identity Data
                - From (Buyer)
                - To (Supplier)
                - Service Type (Supplier Order Processing)
                - Intent (New Purchase Order)
                - Message Manifest
```

```
                - Authorization Data:
                    Buyer Data
                    Status Data (Stock Availability Check)
                    ref. to Signature3
              o Message Routing Info
              o Signatures
                - Signature3, signs:
                    Message Header
                    Purchase Order
                    Signature2 (on Stock Availability Check Response
                    Message)
              o Documents
                - Purchase Order

 4.           The Supplier checks the Purchase Order Request Message and
              generates an Invoice, and a list of Payment Brands that
              the Supplier accepts, signs the documents and sends them
              back to the Buyer.

     B <-- S Response Message (Purchase Order Response):
              o Message Header
                - Message Type (Response)
                - Transaction Identity Data
                - Message Identity Data
                - From (Supplier)
                - To (Buyer)
                - Service Type (Supplier Order Processing)
                - Intent (New Purchase Order)
                - Message Manifest
                - Status Data (New Purchase Order)
              o Message Routing Info
```



```
              o Signatures
                - Signature4, signs
                    Message Header
                    Invoice
                    Brand List
                    Signature3 (on Purchase Order Request Message)
              o Documents
                - Invoice
                - Brand List

 5.           The Buyer checks the Purchase Order Response Message and
              finds it OK. As a result generates a Payment Request,
              provides information on the Payment Instrument to use,
              digitally signs them sends them to the Payment Handler
              (another Party) that is to accept the payment on behalf of
              the Supplier.

     B --> P Request Message (Payment Request):
              o Message Header
                - Message Type (Request)
                - Transaction Identity Data
                - Message Identity Data
                - From (Buyer)
                - To (Payment Handler)
                - Service Type (Payment)
                - Intent (Make Payment)
                - Message Manifest
                - Authorization Data
```

```
                    Buyer Data
                    Supplier Data
                    Status Data (New Purchase Order)
                    ref. to Signature4 (from Purchase Order Response
                    Message)
                    ref to Signature5
              o Message Routing Info
              o Signatures
                - Signature4 (copied from Purchase Order Response
                   Message)
                - Signature5, signs
                    Signature4 (on Purchase Order Response Message)
                    Payment Instrument
                    Payment Request
              o Documents:
                - Payment Request (amount to pay, which Brand)
                - Brand List
                - Payment Instrument

 6.           The Payment Handler Checks the Payment Request including
              the Authorization Data to check that they know the
              Supplier, and on the Payment Instrument data to check that
              the Buyer wants to make the payment. Accepts the Payment,
              generates a Payment Receipt, digitally signs the
              information and sends it back to the Buyer.
```



```
     B <-- P Response Message (Payment Response):
              o Message Header
                - Message Type (Response)
                - Transaction Identity Data
                - Message Identity Data
                - From (Payment Handler)
                - To (Buyer)
                - Service Type (Payment)
                - Intent (Make Payment)
                - Message Manifest
                - Status Data (Payment Response)
              o Message Routing Info
              o Signatures
                - Signature6, signs
                    Message Header
                    Payment Receipt
                    Signature4 (on Purchase Order Response Message)
                    Signature5 (on Payment Request Message)
              o Document:
                - Payment Receipt

 7.           The Buyer checks the Payment Response Message and finds it
              OK. As a result generates a Delivery Request Document and
              sends it to the Supplier to request delivery of the goods.

     B --> S Request Message (Delivery Request):
              o Message Header
                - Message Type (Request)
                - Transaction Identity Data
                - Message Identity Data
                - From (Buyer)
                - To (Supplier)
                - Service Type (Supplier Order Processing)
```

```
                - Intent (Make Delivery)
                - Message Manifest
                - Authorization Data
                    Payment Handler Data
                    ref to Signature6
                    ref to Payment Receipt
              o Message Routing Info
              o Signatures
                - Signature6 (copied from Payment Response Message)
              o Documents
                - Payment Receipt

 8.           The Supplier checks the Delivery Request including the
              Authorization Data to check that payment has been made.
              Checks that delivery is possible and therefore generates a
              Delivery Response that confirms how delivery will occur,
              digitally signs the information and sends the message back
              to the Buyer.

     B <-- S Response Message (Delivery Response):
              o Message Header
```



```
                - Message Type (Response)
                - Transaction Identity Data
                - Message Identity Data
                - From (Supplier)
                - To (Buyer)
                - Service Type (Supplier Order Processing)
                - Intent (Make Delivery)
                - Message Manifest
                - Status Data (Delivery Response)
              o Message Routing Info
              o Signatures
                - Signature7, signs:
                    Message Header
                    Delivery Response
                    Signatures6 (on the Delivery Request Message)
              o Documents
                - Delivery Response

 9.           The Buyer checks the Delivery Response Message and finds
              it OK then logs the information and waits for the physical
              Delivery of the goods.

              Figure 5 Multiple Document Exchange Transaction


 [Note]     In the example above, signatures are used, to help provide
            the audit trail and so that one Party, e.g. the Payment
            Handler, can check the validity of the Payment Request
            Message. Alternative approaches that avoid the need for
            signatures are possible. For example:
            o the Buyer and Seller use a secure Transport Mechanism
              such as [HTTPS] to set up a secure channel between them
            o the Seller authenticates the Buyer (perhaps using a
              User/Server Authentication Transaction)
            o the first two Document Exchanges occur over the secure
              channel.
 [Note End]
```

7.2.5 Relating Two Transactions

XML Messaging allows two or more transactions to be linked together. This example shows how a purchase transaction can be linked to an earlier contract negotiation transaction.

The first example is the contract negotiation transaction. Note that this also illustrates a Multiple Round Trip Document Exchange.

```
 Buyer
     |  Supplier
STEP |     |
 1.          Buyer generates a draft contract places it in an XML
             Message together with a Message Header and sends it to the
             Supplier.
```



```
     B --> S Request Message:
             o Message Header
               - Message Type (Request)
               - Transaction Identity Data
               - Message Identity Data
               - From (Buyer)
               - To (Supplier)
               - Service Type (Contract Processing)
               - Intent (New Draft Contract)
               - Message Manifest
             o Message Routing Info
             o Document
               - Draft Contract

 2.          The Supplier reviews the Contract, makes revisions and
             places the revised contract in an Exchange Message and
             sends it back to the Buyer

     B <-- S Exchange Message:
             o Message Header:
               - Message Type (Exchange)
               - Transaction Identity Data
               - Message Identity Data
               - From (Supplier)
               - To (Buyer)
               - Service Type (Contract Processing)
               - Intent (Contract Amendment)
               - Message Manifest
             o Message Routing Info
             o Document
               - Revised Contract

 3.          The Buyer reviews the revised contract, amends it and
             places it in an XML Message together with a Message Header
             and sends it to the Supplier.

     B --> S Exchange Message:
             o Message Header
               - Message Type (Exchange)
               - Transaction Identity Data
               - Message Identity Data
               - From (Buyer)
```

```
                - To (Supplier)
                - Service Type (Contract Processing)
                - Intent (Contract Amendment)
                - Message Manifest
              o Message Routing Info
              o Document
                - Revised Contract

 N-1.         The Supplier and Buyer keep on swapping drafts of the
              contract in Exchange Messages until eventually they agree
```



```
              and the Supplier sends the final version of the contract,
              digitally signed, back to the Buyer in a Response Message.

      B <-- S Response Message:
              o Message Header:
                - Message Type (Response)
                - Transaction Identity Data
                - Message Identity Data
                - From (Supplier)
                - To (Buyer)
                - Service Type (Contract Processing)
                - Intent (Final Contract)
                - Message Manifest
                - Status Data
              o Message Routing Info
              o Signatures
                - Signature1, signs:
                    Message Header
                    Final Contract
              o Document
                - Final Contract

 N.           The Buyer carries out a final check on the contract and
              stores it for later use.


 Some time later, the Buyer wants to make a purchase under the terms
 of the agreed contract. Now suppose that the contract negotiation
 transaction had an identifier of "ACV-CN-1999/2456@example.com". Then
 the Purchase Transaction could refer to the Contract Negotiation by
 referring to the identifier in the Transaction Identity Data of the
 Purchase Transaction. For example:

 Buyer
     |  Supplier
STEP |     |
 1.           Buyer generates a Purchase Order places it in an XML
              Message. The Transaction Identity Data refers to the
              earlier Contract Negotiation transaction.

      B --> S Request Message (Purchase Order):
              o Message Header
                - Message Type (Request)
                - Transaction Identity Data
                - Message Identity Data
                - From (Buyer)
                - To (Supplier)
                - Service Type (Supplier Order Processing)
```

```
                - Intent (New Purchase Order)
                - Message Manifest
                - Related Transaction Data (refers to
                   "ACV-CN-1999/2456@example.com")
              o Message Routing Info
              o Signatures
```



```
                - Signature1, signs
                    Message Header
                    Purchase Order
              o Document
                - Purchase Order

 2.           Purchase continues ...


 [Note]     Note that the digital signature is being used by the Buyer
            to bind the Purchase Order to the earlier contract
            negotiation.
 [Note End]
```

8. References

This section contains references to related documents identified in this specification.

[Base64] Base64 Content-Transfer-Encoding. A method of transporting binary data defined by MIME. See: RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. N. Freed & N. Borenstein. November 1996.

[DOM-HASH] A method for generating hashes of all or part of an XML tree based on the DOM of that tree. See http://www.ietf.org/internet-drafts/draft-ietf-trade-hiroshi-dom-hash-*.txt.

[DUNS] The Data Universal Numbering System, the D&B D-U-N-S Number, is a unique nine-digit code that helps identify and link more than 57 million companies worldwide. See http://www.dnb.com/dunsno/list.htm.

[HTTP] Hyper Text Transfer Protocol versions 1.0 and 1.1. See RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0. T. Berners-Lee, R. Fielding & H. Frystyk. May 1996. and RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. January 1997.

[HTTPS] Hyper Text Transfer Protocol (secure) transported using The TLS Protocol (see RFC 2246, T. Dierks & C Allen. January 1999.

[IOTP] The Internet Open Trading Protocol, David Burdett et al. See RFCxxx. This document is currently in the RFC editor queue. Also see http://www.ietf.org/internet-drafts/draft-ietf-trade-iotp-v1.0-protocol-07.txt.

[MIME] Multipurpose Internet Mail Extensions. See RFC822, RFC2045, RFC2046, RFC2047, RFC2048 and RFC2049.

[MONDEX] A proprietary electronic cash product where cash is stored on a smart card. See http:/www.mondex.com/

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

[SET] Secure Electronic Transaction. See http://www.setco.org/

[SMTP] Simple Mail Transfer Protocol, RFC 821, J. Postel, August 1982

[SSL/TLS] SSL is a standard developed by Netscape for encrypting

data over IP networks. See http://home.netscape.com/eng/ssl3/index.html. TLS is the likely successor to SSL being developed by the IETF. See http://www.ietf.org/internet-drafts/draft-ietf-tls-protocol-05.txt

[URI] Uniform Resource Identifiers (URI): Generic Syntax. T Berners-Lee, R. Fielding, L. Masinter. IETF RFC 2396.

[URL] Universal Resource Locator (URL). R Fielding, IETF RFC 1808

[XDSL] The W3C Schema language specification. See http://www.w3.org/TR/xmlschema-1/

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

[XML.org] A repository designed to hold definitions of XML Documents. See http://www.xml.org./

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

[XMLDSIG] XML Signature Core Syntax and Processing. Specifies XML digital signature processing rules and syntax. A W3C Working Draft. See http://www.w3.org/TR/xmldsig-core/

[XMLMSG-CDE] Common Document Exchanges. Defines a number of Common Document Exchanges that are generally applicable to many situations.

[XMLMSG-CME] XML Messaging - Common XML Message Elements. Contains definitions of elements and attributes used to construct messages that conform to XML Messaging. To be completed.

[XMLMSG-DCD] Document Choreography Definitions. Describes how the sequence in which documents are exchanged may be defined, agreed between the parties involved and then used dynamically to determine the way in which a particular type of business service or process is carried out.

[XMLMSG-DET] Document Exchanges and Transactions. Defines standard templates for exchanging documents between parties that can be used to implement transactions that support different types of services and processes. To be completed.

[XMLMSG-RM] Reliable Messaging. Defines how to exchange messages in a way that is reliable, robust and resilient and

results in "guaranteed once-only" message delivery.

[XMLMSG-SM] Secure Messaging. Describes how digital signatures and other methods such as [SSL] may be used to ensure the tamper resistance and authenticated exchange of messages.

9. Author's Address

The author of this document is:

David Burdett
Commerce One
1600 Riviera Ave, Suite 200
Walnut Creek
California 94596
USA

Tel: +1 (925) 941 4422 or +1 (650) 623 2888

Email: david.burdett@commerceone.com

The author would also like to acknowledge everyone who worked on the development and implementation of [IOTP] on which many of the ideas in this specification are based and the numerous individuals who have reviewed earlier versions of this document and provided their most helpful comments.

File Name: draft-ietf-trade-xmlmsg-requirements-00.txt