ebXML Transport Definitions

The following are a list of definitions of the terms associated with the transport of messages over the Internet. They are derived initially from work being done within the IETF.

It is split into two sections:

- Documents, Parties, Messages and Document Exchanges, and
- Services and Transactions

Words or phrases that defined elsewhere are highlighted in *italics*.

COMMENTS WILL BE APPRECIATED.

David Burdett

1 Documents, Parties, Messages and Document Exchanges

1.1 Overview

This section describes how *Parties*, such as buyers and suppliers, customers and merchants, can ~~exchange~~ transmit *Documents* contained in *Messages* in order to request execution~~e~~ of *Services*.

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

A *Message* can optionally include a *Digital Signature* so that:

1) the authenticity of the message might be determined
2) any changes to the message can be identified.

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

3) processes the *Request Message* by carrying out a *Service* and
4) optionally generates a *Response Message* to indicate the result.

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

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

More detail is provided below.

1.2 A Document

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

Examples of *Documents* include:

1) a set of XML Elements
2) an XML Document
3) an HTML Document
4) a word processing file
5) a Adobe Acrobat PDF file
6) a binary file.

1.2.1 Party

A *Party* is a company, organization or individual or other entity that can generate or receive *Documents*.

Examples of a *Party* include:

1) a Merchant
2) a Customer
3) a Lawyer
4) a Bank

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

~~1.1.2~~1.2.2 Message

A *Message* is data that is sent from one *Party* to another. A *Message* consists of information such as:

1) a *Message Envelope* that indicates who sent, who should receive and the reason for sending the message
2) *Message Routing Information*, that indicates how the message was delivered
3) zero or more *Digital Signatures* to bind the data in the message, or elsewhere, together, and
4) zero or more *Documents* which is the business data that actually needs to be sent

All the data in a *Message* is contained within a *Message Wrapper*.

Examples of a *Message* include:

1) a Purchase Order that is sent by a buyer to a supplier
2) an Invoice that is sent by the supplier back to the buyer
3) a request to make a payment of $50 sent to a Credit Card acquirer
4) the authorization received from a Credit Card acquirer as a result of making a payment
5) Status Data indicating the success or failure of a Service

1.2.3 Message Envelope

A *Message Envelope* 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*. It can contain information such as:

1) Transaction Identity data to identify the set of *Messages* that are related to one another through one or more *Document Exchanges*
2) Message Identity data to enable the *Message* to be identified and referenced within the *Transaction*
3) a Message Manifest to identify the documents, other than the Message Envelope, that are contained within the same *Message Wrapper*
4) Action Data to indicate the *Service* that is being sent the message and the reason for sending
5) Organization Data that describes one or more of:
    a) the Sender organization that sent the *Message*
    b) the Recipient organization(s) that ought to receive the *Message*
    c) the Authorizing organization(s) that provide evidence that a requested *Service* should be carried out.
6) Status Data that describes the results of carrying out a *Service*.

1.2.4 Message Routing Information

Message Routing Information contains data that indicates the path taken by a *Message* in reaching its ultimate destination.

~~1.1.5~~1.2.5 Digital Signature

A *Digital Signature* is a cryptographic signature over data contained in a *Message*, or elsewhere that are addressable via [URI]s, that permits the authenticity of the data being signed to be determined, and helps detect if the data in the *Message* has changed.

~~1.1.6~~1.2.6 Message Wrapper

A *Message Wrapper* is the outermost container for a Message. It can be such things as:

1) an XML Document, or
2) a multi-part MIME message

1.2.7 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*.

~~1.1.8~~1.2.8 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*.

~~1.1.9~~1.2.9 Document Exchange

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

~~1.1.10~~1.2.10 Simple Document Exchange

A Simple Document Exchange consists of:

1) a *Request Message* sent from one *Party* to a second *Party*, and
2) the *Response Message* that is returned as a result.

Examples of instances of a *Simple Document Exchange* include:

1) a Purchase Order sent by a buyer to a seller and the acknowledgement from the seller of its receipt
2) a Purchase Order sent by a buyer to a seller and the Invoice that is sent back as a result of fulfilling the order
3) sending a document for review by a lawyer followed by the legal opinion that is sent back as a result

1.2.11 Multiple Round Trip Document Exchange

A *Multiple Round Trip Document Exchange* consists of:

1) a *Request Message* sent from one *Party* to a second *Party*, followed by
2) a series of *Exchange Messages* that are exchanged between the two *Parties* until finally
3) the second *Party* generates and sends a *Response Message* back to the first *Party*.

Examples of *Multiple Round Trip Document Exchanges* include:

1) the exchange of messages required to make a payment using payment method protocols such as [SET] or [Mondex]
2) the exchange of messages required to negotiate an agreement on terms and conditions.

1.2.12 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*.

Examples of *Exchange Messages* include:

1) intermediate messages that are part of a Payment Protocol
2) a counter offer to an offer made as part of a negotiation.

1.2.13 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.

Examples of an *Error Message* include:

1) an *Error Message* reporting that an XML document was invalid or did not conform to its XML schema
2) 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.

1.3 Services and Transactions

1.3.1 Overview

A *Service Definition* consists of either a *Document Exchange* or a set of *Sub-Services*. Each *Sub-Service* is a *Service* in its own right. So, at the lowest level, all *Service Definitions* must consist of *Document Exchanges*.

The dependencies between the *Sub-Services* in a *Service* is described in a *Sub-Service Choreography*.

An instance of the execution of a *Service Definition* is called a *Transaction*.

More detail is provided below.

~~1.1.2~~1.3.2 Service Definition

A *Service Definition* describes a process that can be carried out by a *Party* as a result of receiving a *Request Message* that requests the execution of that *Service*.

A *Service Definition* can consist of either:

1) a Document Exchange, or
2) a set of Sub-Services

Examples of *Service Definitions* include descriptions of:

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

1.3.3 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:

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

1.3.4 Sub-Service Choreography

A *Sub-Service Choreography* is a description of the dependencies that control the sequence and choices that determine which *Sub-Services* are executed when carrying out a *Transaction*.

The *Sub-Services* in a *Service* will have dependencies between them. Dependencies can be:

1) Serial. One *Sub-Service* must start only after the completion of another *Sub-Service*
2) Alternative. One *Sub-Service* may be executed as an alternative to another
3) Iterative Loop. A *Sub-Service* may be repeated a variable number of times
4) 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.
5) Parallel. A *Sub-Service* may execute in Parallel with another *Service*
6) Concurrent. A *Sub-Service* must Execute at the same time as another *Sub-Service*.

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

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

1.3.5 Transaction

A *Transaction* is an instance of the execution of a *Service*.

Examples of a *Transaction* include:

1) a Purchase Transaction that buys an Company Report for $20. It consists of three Sub-Service instances:
    a) an Offer Service instance to buy the Company Report for $20
    b) a Payment Service instance that accepts a Payment for $20 using a credit card, and finally
    c) a Delivery Service instance that delivers the Company Report as an HTML web page.
2) a Buying Service that consists of the following Sub-Services:
    a) three Price Negotiation Service instances that negotiate the price of a Photocopier
    b) a Purchase Order Service instance that places the order for the Photocopier.