OASIS ebXML Messaging Services
Version 3.0
Working Draft 01, 5 May 2004

Document identifier:
wd-ebms-3_0-01

Location:
http://www.oasis-open.org/committees/ebxml-msg

Editor:
Matthew MacKenzie, Adobe Systems Incorporated <mattm@adobe.com>

Contributors:
Jeff Turpin, Cyclone Commerce <jturpin@cyclonecommerce.com>
Ian Jones, Individual Member <ian.c.jones@bt.com>
Doug Bunting, Sun Microsystems <doug.bunting@sun.com>
Dale Moberg, Cyclone Commerce <dmoberg@cyclonecommerce.com>
Jacques Durand, Fujitsu Software <jdurand@us.fujitsu.com>

Abstract:
This abstract needs to be written.

Status:
This is a Working Draft.

Committee members should submit comments to the ebxml-msg@lists.oasis-open.org list. Others should submit comments by filling out the form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=ebxml-msg

The errata page for this specification is at [todo] [http://www.oasis-open.org/committees/ebxml-msg/errata/3_0].

Copyright © 2004 OASIS Open, Inc. All Rights Reserved.

Table of Contents
1. Introduction ................................................................. 3
2. Terminology ............................................................... 3
3. Audience ................................................................. 3
4. Caveats & Assumptions ..................................................... 3
5. Concept of Operation ..................................................... 3
  5.1. Scope .................................................................. 3

# OASIS ebXML Messaging Services

## Appendixes

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. The ebXML SOAP Extension Elements Schema</td>
<td>12</td>
</tr>
<tr>
<td>B. Communications Protocol Bindings</td>
<td>12</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>13</td>
</tr>
<tr>
<td>2. HTTP</td>
<td>13</td>
</tr>
<tr>
<td>2.1. Minimum Level of HTTP Protocol</td>
<td>13</td>
</tr>
<tr>
<td>2.2. Sending ebXML Service Messages over HTTP</td>
<td>13</td>
</tr>
<tr>
<td>2.3. HTTP Response Codes</td>
<td>13</td>
</tr>
<tr>
<td>2.4. SOAP Error Conditions and Synchronous Exchanges</td>
<td>13</td>
</tr>
<tr>
<td>2.5. Synchronous vs. Asynchronous</td>
<td>13</td>
</tr>
<tr>
<td>2.6. Access Control</td>
<td>13</td>
</tr>
<tr>
<td>2.7. Confidentiality and Transport Protocol Level Security</td>
<td>13</td>
</tr>
<tr>
<td>3. SMTP</td>
<td>13</td>
</tr>
<tr>
<td>3.1. Minimum Level of Supported Protocols</td>
<td>13</td>
</tr>
<tr>
<td>3.2. Sending ebXML Service Messages over SMTP</td>
<td>13</td>
</tr>
<tr>
<td>3.3. Response Messages</td>
<td>13</td>
</tr>
<tr>
<td>3.4. Access Control</td>
<td>13</td>
</tr>
<tr>
<td>3.5. Confidentiality and Transport Protocol Level Security</td>
<td>13</td>
</tr>
<tr>
<td>3.6. SMTP Model</td>
<td>13</td>
</tr>
<tr>
<td>3.7. Communication Errors during Reliable Messaging</td>
<td>13</td>
</tr>
<tr>
<td>C. Supported Security Services</td>
<td>13</td>
</tr>
<tr>
<td>D. Relationship to WSDL</td>
<td>13</td>
</tr>
<tr>
<td>E. WS-I Compliance</td>
<td>13</td>
</tr>
<tr>
<td>F. Notices</td>
<td>14</td>
</tr>
<tr>
<td>G. Revision History</td>
<td>14</td>
</tr>
</tbody>
</table>
1. Introduction

This specification is one of a series of specifications realizing the vision of creating a single global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based messages. The set of specifications enable a modular, yet complete electronic business framework.

This specification focuses on defining a communications-protocol neutral method for exchanging electronic business messages. It defines specific enveloping constructs supporting reliable, secure delivery of business information. Furthermore, the specification defines a flexible enveloping technique, permitting messages to contain payloads of any format type. This versatility ensures legacy electronic business systems employing traditional syntaxes (i.e. UN/EDIFACT, ASC X12, or HL7) can leverage the advantages of the ebXML infrastructure along with users of emerging technologies.

2. 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 [RFC 2119].

Some specifications may choose to capitalize these words, or use other typographic conventions to make them stand out and to separate their use in a normative sense from their use in a natural-language sense. This document does not.

3. Audience

The target audience for this specification is the community of software developers who will implement the ebXML Message Service.

4. Caveats & Assumptions

It is assumed the reader has an understanding of communications protocols, MIME, XML, SOAP, SOAP Messages with Attachments and security technologies.

All examples are to be considered non-normative. If inconsistencies exist between the specification and the examples, the specification supersedes the examples.

It is strongly RECOMMENDED implementors read and understand the Collaboration Protocol Profile/ Agreement [ebCPPA] specification and its implications prior to implementation.

5. Concept of Operation

5.1. Scope

The ebXML Message Service (ebMS) defines the message enveloping and header document schema used to transfer ebXML messages over a communications protocol such as HTTP or SMTP and the behavior of software sending and receiving ebXML messages. The ebMS is defined as a set of layered extensions to the base Simple Object Access Protocol [SOAP] and SOAP Messages with Attachments [SOAPAttach] specifications. This document provides security and reliability features necessary to support international electronic business. These security and reliability features are not provided in the SOAP or SOAP with Attachments specifications.
The ebXML infrastructure is composed of several independent, but related, components. Specifications for the individual components are fashioned as stand-alone documents. The specifications are totally self-contained; nevertheless, design decisions within one document can and do impact the other documents. Considering this, the ebMS is a closely coordinated definition for an ebXML message service handler (MSH).

The ebMS provides the message packaging, routing and transport facilities for the ebXML infrastructure. The ebMS is not defined as a physical component, but rather as an abstraction of a process. An implementation of this specification could be delivered as a wholly independent software application or an integrated component of some larger business process.

## 5.2. Background and Objectives

Traditional business information exchanges have conformed to a variety of standards-based syntaxes. These exchanges were largely based on electronic data interchange (EDI) standards born out of mainframe and batch processing. Some of the standards defined bindings to specific communications protocols. These EDI techniques worked well; however, they were difficult and expensive to implement. Therefore, use of these systems was normally limited to large enterprises possessing mature information technology capabilities.

The proliferation of XML-based business interchanges served as the catalyst for defining a new global paradigm that ensured all business activities, regardless of size, could engage in electronic business activities. The prime objective of ebMS is to facilitate the exchange of electronic business messages within an XML framework. Business messages, identified as the ‘payloads’ of the ebXML messages, are not necessarily expressed in XML. XML-based messages, as well as traditional EDI formats, are transported by the ebMS. Actually, the ebMS payload can take any digital form—XML, ASC X12, HL7, AIAG E5, database tables, binary image files, etc.

The ebXML architecture requires that the ebXML Message Service protocol be capable of being carried over any available communications protocol. Therefore, this document does not mandate use of a specific communications protocol. This version of the specification provides bindings to HTTP and SMTP, but other protocols can, and reasonably will, be used.

The ebXML Requirements Specification [ebREQ] mandates the need for secure, reliable communications. The ebXML work focuses on leveraging existing and emerging technology—attempts to create new protocols are discouraged. Therefore, this document defines security within the context of existing security standards and protocols. Those requirements satisfied with existing standards are specified in the ebMS, others must be deferred until new technologies or standards are available, for example encryption of individual message header elements.

Reliability requirements defined in the ebREQ relate to delivery of ebXML messages over the communications channels. The ebMS provides mechanisms to satisfy the ebREQ requirements. The reliable messaging elements of the ebMS supply reliability to the communications layer; they are not intended as business-level acknowledgments to the applications supported by the ebMS. This is an important distinction. Business processes often anticipate responses to messages they generate. The responses may take the form of a simple acknowledgment of message receipt by the application receiving the message or a companion message reflecting action on the original message. Those messages are outside of the MSH scope. The acknowledgment defined in this specification does not indicate the payload of the ebXML message was syntactically correct. It does not acknowledge the accuracy of the payload information. It does not indicate business acceptance of the information or agreement with the content of the payload. The ebMS is designed to provide the sender with the confidence the receiving MSH has received the message securely and intact.

The underlying architecture of the MSH assumes messages are exchanged between two ebMS-compliant MSH nodes. This pair of MSH nodes provides a hop-to-hop model extended as required to support a multi-hop environment. The multi-hop environment allows the next destination of the message to be an intermediary MSH other than the ‘receiving MSH’ identified by the original sending MSH. The ebMS architecture assumes the sender of the message MAY be unaware of the specific path used to deliver a message. However, it MUST be assumed the original sender has knowledge of the final recipient of the message and the first of one or more intermediary hops.

The MSH supports the concept of ‘quality of service.’ The degree of service quality is controlled by an agreement existing between the parties directly involved in the message exchange. In practice, multiple agreements may be re-
quired between the two parties. The agreements might be tailored to the particular needs of the business exchanges. For instance, business partners may have a contract defining the message exchanges related to buying products from a domestic facility and another defining the message exchanges for buying from an overseas facility. Alternatively, the partners might agree to follow the agreements developed by their trade association. Multiple agreements may also exist between the various parties handling the message from the original sender to the final recipient. These agreements could include:

1. an agreement between the MSH at the message origination site and the MSH at the final destination; and
2. agreement between the MSH at the message origination site and the MSH acting as an intermediary; and
3. an agreement between the MSH at the final destination and the MSH acting as an intermediary. There would, of course, be agreements between any additional intermediaries; however, the originating site MSH and final destination MSH MAY have no knowledge of these agreements.

An ebMS-compliant MSH shall respect the in-force agreements between itself and any other ebMS-compliant MSH with which it communicates. In broad terms, these agreements are expressed as Collaboration Protocol Agreements (CPA). This specification identifies the information that must be agreed. It does not specify the method or form used to create and maintain these agreements. It is assumed, in practice, the actual content of the contracts may be contained in initialization/configuration files, databases, or XML documents complying with the ebXML Collaboration Protocol Profile and Agreement Specification [ebCPP].

5.3. Operational Policies and Constraints
5.4. Modes of Operation
6. Conformance
6.1. Minimum Requirements for Conformance
7. Message Package Specification
7.1. SOAP Structural Conformance
7.2. Message Package
7.3. Header Container
7.4. Payload Container
7.4.1. Embedded Payload Container
7.4.2. Attachment Payload Containers
7.4.3. MIME Considerations
7.4.3.1. Additional MIME Parameters
7.4.3.2. Reporting MIME Errors

7.4.4. XML Prolog

7.4.4.1. XML Declaration

7.4.4.2. Encoding Declaration

7.4.5. ebXML SOAP Envelope extensions

7.4.5.1. Namespace pseudo attribute

7.4.5.2. xsi:schemaLocation attribute

7.4.5.3. SOAP Header Element

7.4.5.4. SOAP Body Element

7.4.6. ebXML SOAP Extensions

7.4.6.1. SOAP Header Extensions

7.4.6.2. SOAP Body Extensions

7.4.6.3. Core ebXML Modules

7.4.7. #wildcard Element Content

7.4.8. id Attribute

7.4.9. version Attribute

7.4.10. SOAP mustUnderstand Attribute

7.4.11. ebXML "Next MSH" actor URI

7.4.12. ebXML "To Party MSH" actor URI

8. Core Extension Elements

8.1. MessageHeader Element

8.1.1. From and To Elements

8.1.1.1. PartyId Element

8.1.1.2. Role Element

8.1.2. CPAId Element
8.1.3. ConversationId Element  
8.1.4. Service Element  
8.1.4.1. type Attribute  
8.1.5. Action Element  
8.1.6. MessageData Element  
8.1.6.1. MessageId Element  
8.1.6.2. Timestamp Element  
8.1.6.3. RefToMessageId Element  
8.1.6.4. TimeToLive Element  
8.1.7. DuplicateElimination Element  
8.1.8. Description Element  
8.1.9. MessageHeader Sample  

8.2. Manifest Element  
8.2.1. Reference Element  
8.2.1.1. Schema Element  
8.2.1.2. Description Element  
8.2.2. Manifest Validation  
8.2.3. Manifest Sample  

9. Core Modules  
9.1. Security Module  
9.1.1. Signature Element  
9.1.2. Security and Management  
9.1.2.1. Collaboration Protocol Agreement  
9.1.3. Signature Generation
9.1.4. Countermeasure Technologies
9.1.4.1. Persistent Digital Signature
9.1.4.2. Persistent Signed Receipt
9.1.4.3. Non-persistent Authentication
9.1.4.4. Non-persistent Integrity
9.1.4.5. Persistent Confidentiality
9.1.4.6. Non-persistent Confidentiality
9.1.4.7. Persistent Authorization
9.1.4.8. Non-persistent Authorization
9.1.4.9. Trusted Timestamp

9.1.5. Security Considerations

9.2. Error Handling Module
9.2.1. Definitions
9.2.2. Types of Errors
9.2.3. ErrorList Element
9.2.3.1. highestSeverity attribute
9.2.3.2. Error Element
9.2.3.3. ErrorList Element
9.2.3.4. errorCode values
9.2.3.5. ErrorList Sample
9.2.3.6. Implementing Error Reporting and Handling
9.2.3.6.1. When to Generate Error Messages
9.2.3.6.2. Identifying the Error Reporting Location
9.2.3.6.3. Service and Action Element Values

9.3. SyncReply Module
9.3.1. SyncReply Element
10. Combining ebXML SOAP Extension Elements

10.1. MessageHeader Element Interaction
10.2. Manifest Element Interaction
10.3. Signature Element Interaction
10.4. Errorlist Element Interaction
10.5. SyncReply Element Interaction

11. Additional (optional) Features

11.1. Reliable Messaging Module

THIS SECTION WILL BE REPLACED WITH APPROVED PROPOSAL FOR INTEGRATION OF WS-Reliability.

11.2. Payload Services

11.2.1. Introduction

Payload services refers to functionality implemented by the messaging server to automatically perform some manipulation on payload content either before envelope digital signing (if non repudiation is being used) and message transmission, or after digital signature verification (if non repudiation is being used) and before passing the payload(s) in question to the application.

Payload services are not meant to be used as application level message handlers, rather, they should be treated as "filters".

11.2.2. Example Use Cases for Payload Services

11.2.2.1. Transparently converting XML content to ASN.1 and vice-versa

There are cases where it is desirable to use a more compact format, such as Abstract Syntax Notation to transmit data between partners, while still maintaining the easy-to-process and display qualities of XML. By using ASN.1's XML Encoding Rules (X ER), it is possible to convert between ASN.1 and XML.

Payload services could be used to automatically create ASN.1 representations of XML payloads prior to the message being transmitted. On the receiver's side, the same payload service could be used to convert back to XML. The net result being that the application developers only see XML on both sides.

11.2.2.2. Compression

Sometimes, using a specialized compression algorithm can yield impressive reductions in a server's network utilization.

Payload services could be used to automatically compress and decompress payload content.
11.2.2.3. Encryption

Users often apply encryption at the payload level to ensure confidentiality of their payloads. Often, all that is needed to ensure confidentiality is to encrypt a single payload, as opposed to heavy weight approaches such as using S/MIME to encrypt an entire message.

Payload services could be used to encrypt and decrypt payloads.

11.2.2.4. XSL Transforms

As XML vocabularies evolve, business processes making use of XML messages will need to evolve with changes to their XML vocabularies. In cases where it would be too costly to modify the application(s) producing and consuming the XML, XSL can be used to perform structural transformations.

Payload services could be used to automatically execute an XSL stylesheet prior to transmission, or before being delivered to the application.

11.2.3. Payload Service Invocation

Payload service invocation can be requested using two methods: SOAP Header Extensions, or CPA entries.

In cases where invocation requests are specified in both the CPA and the SOAP header, the CPA takes precedence. If the CPA entry explicitly forbids the use of payload services, then the Payload Services SOAP Headers MUST be ignored.

Payload services MAY NOT be invoked on the 0th attachment -- the SOAP envelope.

11.2.3.1. Invocation by SOAP Header Extension

Payload services may be invoked upon payloads by inserting a block of XML into the SOAP Header. Below is an example of the XML format for invoking payload services.

**Figure 1. Sample XML**

```
<PayloadServices>
  <Service name="urn:foo:ps:Normalizer" payloadRef="cid:att1" type="onSend" order="0" />
  <Service name="urn:foo:ps:CompressionSvc" payloadRef="cid:att1 type="onSend" order="1"
    <Parameter name="command" value="compress" />
    <Parameter name="algorithm" value="gzip" />
    <Parameter name="gzip-level" value="9" />
  /></Service>
  <Service name="urn:foo:ps:CompressionSvc" payloadRef="cid:att1 type="onReceive" order="0"
    <Parameter name="command" value="uncompress" />
    <Parameter name="algorithm" value="gzip" />
  /></Service>
</PayloadServices>
```

As the example above illustrates, it is possible to chain payload services together using the *order* attribute. When interpreting the *order* attribute, lower numbers have higher priorities, and must be executed first. The value of the *order* attribute does not have to start at 0, although that convention is recommended for the sake of simplicity.

11.2.3.2. Error Handling
Error handling for payload services is out of the scope of this specification. It is assumed that problems encountered in the processing of payload services will be reported back to the application, or otherwise handled by the MSH.

11.2.3.3. Invocation by Collaborative Protocol Agreement

TBD

**TODO: discuss with CPPA TC.**

11.2.4. Required Services

This section defines services that every ebMS 3.0 compliant message handler must implement.

11.2.4.1. Encryption Service

This service, named `urn:oasis-open:committees:ebxml-msg:ps:encrypt` provides a way of encrypting payloads using a variety of algorithms. This specification does not define which algorithms should be implemented, but W3C XML Encryption is recommended.

**Example 1. Sample SOAP Header Extension for XML Encryption**

```xml
<PayloadServices>
  <Service name="urn:oasis-open:committees:ebxml-msg:ps:encrypt" payloadRef="cid:att1 type="onSend" order="0">
    <Parameter name="command" value="encrypt" />
    <Parameter name="xpath" value="/Invoice" />
    <Parameter name="key" value="accountspayable" />
  </Service>
  <Service name="urn:oasis-open:committees:ebxml-msg:ps:encrypt" payloadRef="cid:att1 type="onReceive" order="0">
    <Parameter name="command" value="decrypt" />
    <Parameter name="xpath" value="/Invoice" />
    <Parameter name="key" value="accountspayable" />
  </Service>
</PayloadServices>
```

**Example 2. Sample CPA entry**

**TODO**

11.3. Message Status Service

11.3.1. Message Status Messages

11.3.1.1. Message Status Request Message

11.3.1.2. Message Status Response Message
11.3.1.3. Security Considerations
11.3.2. StatusRequest Element
11.3.2.1. RefToMessageId Element
11.3.2.2. StatusRequest Sample
11.3.2.3. StatusRequest Element Interaction
11.3.3. StatusResponse Element
11.3.3.1. RefToMessageId Element
11.3.3.2. Timestamp Element
11.3.3.3. messageStatus attribute
11.3.3.4. StatusResponse Sample
11.3.3.5. StatusResponse Element Interaction
11.4. Message Service Handler Ping Service
11.4.1. Message Service Handler Ping Message
11.4.2. Message Service Handler Pong Message
11.4.3. Security Considerations
11.5. MessageOrder Module
11.5.1. MessageOrder Element
11.5.1.1. SequenceNumber Element
11.5.1.2. MessageOrder Sample
11.5.2. MessageOrder Element Interaction
11.6. Multi-Hop Module

The ebXML SOAP Extension Elements Schema
Communications Protocol Bindings
1. Introduction

2. HTTP
   2.1. Minimum Level of HTTP Protocol
   2.2. Sending ebXML Service Messages over HTTP
   2.3. HTTP Response Codes
   2.4. SOAP Error Conditions and Synchronous Exchanges
   2.5. Synchronous vs. Asynchronous
   2.6. Access Control
       2.6.1. Basic & Digest Authentication
       2.6.2. SSL Client (Digital Certificate) Authentication
   2.7. Confidentiality and Transport Protocol Level Security

3. SMTP
   3.1. Minimum Level of Supported Protocols
   3.2. Sending ebXML Service Messages over SMTP
   3.3. Response Messages
   3.4. Access Control
   3.5. Confidentiality and Transport Protocol Level Security
   3.6. SMTP Model
   3.7. Communication Errors during Reliable Messaging

Supported Security Services

Relationship to WSDL

WS-I Compliance
Notices

Copyright © 2002, 2003, 2004 OASIS Open, Inc. All Rights Reserved.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS’s procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS has been notified of intellectual property rights claimed in regard to some or all of the contents of this specification. For more information consult the online list of claimed rights.

Revision History

Revision 01 5 May 2004 mm
Create outline, document structure, added payload services

References

Normative

Normative

