



Creating A Single Global Electronic Market

# ebXML Transport, Routing & Packaging Reliable Messaging Specification

## 1 Working Draft 2-October-2000

### 2 This version:

3 ebXML Reliable Messaging Specification v0-080

### 4 Latest version:

5 N/A

### 6 Previous version:

7 v0-078

### 8 Editor:

9 Jim Hughes <[jfh@fs.fujitsu.com](mailto:jfh@fs.fujitsu.com)>

### 10 Authors:

11 Masayoshi Shimamura <[shima@rp.open.cs.fujitsu.co.jp](mailto:shima@rp.open.cs.fujitsu.co.jp)>

### 12 Contributors:

13 See Acknowledgements

---

## 14 Abstract

15 This document defines the structures and processes used to provide Reliable Messaging within  
16 the ebXML Transport, Routing and Packaging architecture.

## 17 Status of this Document

18 This document represents work in progress and no reliance should be made on its content.

19 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",  
20 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be  
21 interpreted as described in IETF RFC 2119.

### 22 Editor Note 1: This version:

- 23 - incorporates (as appropriate) email comments through 28 September
- 24 - incorporates the decisions made during the Dallas F2F
- 25 - splits the prior version into a part for incorporation into MS Ver0.21c
- 26 - identifies the remaining parts as Phase 2 or Phase 3 work, or for inclusion in a non-  
27 normative Implementer's Guide (to be developed)

28



## 28 Table of Contents

29	1	Introduction .....	3
30	1.1	Purpose and Scope .....	3
31	2	Sections to Add to MS Ver0.21C.....	3
32	2.1	Definition and Scope [MS section 7].....	3
33	2.1.1	EbXML Message Structure [MS section 7.1.1] .....	3
34	2.1.2	EbXML Header Container Example [MS section 7.3.4] .....	3
35	2.1.3	Reliable Messaging Flow [MS section 7.5, added] .....	3
36	2.1.4	Reliable Messaging Recovery Procedures [MS section 7.6, added] .....	5
37	2.2	ebXML Header Document [MS section 8] .....	7
38	2.2.1	ReliableMessagingInfo [MS section 8.4.4] .....	7
39	2.2.2	Routing Header Document [MS section 8.5, added] .....	8
40	2.3	Schema and DTD Definitions [MS Appendix A].....	9
41	2.4	Examples [MS Appendix B] .....	9
42	2.5	Communication Protocol Interface [MS Appendix E].....	9
43	2.5.1	HTTP .....	10
44	2.5.2	SMTP .....	11
45	2.5.3	FTP .....	12
46	2.5.4	Communication Protocol Errors during Reliable Messaging .....	12
47	3	Modifications to ebXML Glossary .....	13
48	3.1	Reliable Messaging Terms .....	13
49	3.1.1	Once And Only Once .....	13
50	3.1.2	At Most Once .....	13
51	3.1.3	Best Effort .....	14
52	4	Phase 2/Phase 3 Activities .....	14
53	4.1	Reliable Routing .....	14
54	4.1.1	Store and Forward Semantics.....	14
55	4.1.2	Routing Information.....	15
56	4.1.3	Error Handling in Routing .....	15
57	4.2	Transient Errors .....	15
58	5	Non-normative Material .....	16
59	5.1	Basic Concepts .....	16
60	5.2	Detection of Repeated Messages by the Receiver .....	18
61	6	Acknowledgements.....	18
62	7	Author's Address .....	18
63			
64			
65			



## 65 **1 Introduction**

### 66 **1.1 Purpose and Scope**

67 This specification defines the Reliable Messaging function used between ebXML Messaging  
68 Services. It responds to the requirements for Reliable Messaging found in section 4.2(1) of  
69 Reference [1]. Material from this draft document will be incorporated into the Messaging Services  
70 Specification at this time, and at a future date after further definitions are completed and there is  
71 experience from the Proof of Concept activities.

72 Where appropriate, MS-Editor notes are provided to show where items might appear in version  
73 0.21C of the Messaging Services specification.

## 74 **2 Sections to Add to MS Ver0.21C**

75 The material in this section is suitable for inclusion in Phase 1 specifications.

### 76 **2.1 Definition and Scope [MS section 7]**

#### 77 **2.1.1 EbXML Message Structure [MS section 7.1.1]**

78 *MS Editor – insert Routing Header box into the ebXML Header Document in Figure 7-1.*

#### 79 **2.1.2 EbXML Header Container Example [MS section 7.3.4]**

80 *MS Editor – add <Routing Header> ... </Routing Header> into example.*

#### 81 **2.1.3 Reliable Messaging Flow [MS section 7.5, added]**

82 The Reliable Messaging function defines an interoperable protocol such that any two Messaging  
83 Service implementations can “reliably” exchange messages that are sent using “reliable  
84 messaging” semantics.

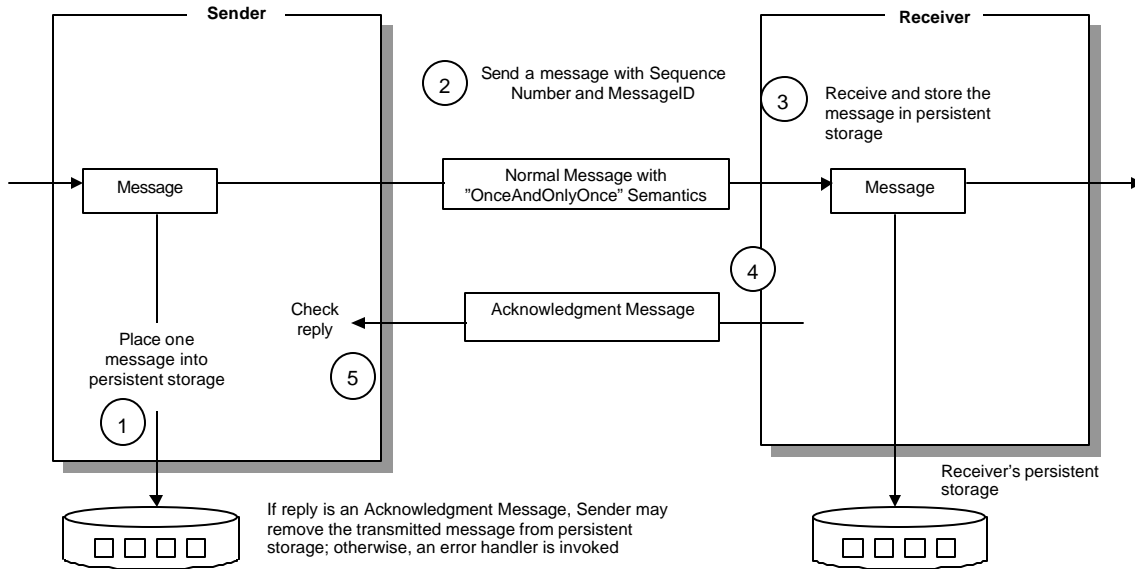
85 Reliably exchanging messages means that, with respect to Sending and Receiving Message  
86 Service implementations:

- 87 • For any given message provided to the Sending Messaging Service, the Receiving  
88 Messaging Service will deliver at most one copy of the message to the Receiver.
- 89 • A positive acknowledgement will be sent from the Receiving Messaging Service to the  
90 Sending Messaging Service to indicate receipt and storage in persistent storage, and if this  
91 acknowledgement is not received the Sending Messaging Service will notify the original  
92 Sending Party
- 93 • Both the Sending and Receiving Messaging Services will use persistent storage for recovery

94 Reliable Messaging is defined only for direct connections between Messaging Service  
95 implementations. At a later time, networks consisting of intermediate Messaging Service  
96 implementations will be supported.

97 All ebXML Messaging Service implementations SHALL support the Reliable Messaging function.  
98 With respect to a particular Sender and Receiver pair, transmission of one reliable message  
99 SHALL be completed before another reliable message may be sent.

100 The following figure shows the reliable messaging flow:



101  
102

**Figure 7-2: Reliable Message Transfer Sequence**

103 Reliable Messaging processing is shown in the following sequence:

104 (1) Message preparation

105 Sender initially stores messages passed from the ebXML "From-Party" in persistent storage,  
106 and then prepares the stored message for message transfer.

107 (2) Sending message

108 A Reliable Message has DeliverySemantics = "OnceAndOnlyOnce", and receipt of a  
109 message with this value notifies the Receiver of Reliable Messaging semantics.

110 (3) Receiving, checking and storing message

111 The Receiver receives the reliable message and, if the message is not a duplicate message,  
112 stores the message in persistent storage and processes the message appropriately.

113 (4) Acknowledgment by Receiver

114 The Receiver returns an Acknowledgment Message to the Sender for every received reliable  
115 message, even if it is a duplicate message.

116 (5) Sender checks the acknowledgement and removes transferred message

117 Sender checks the Acknowledgement Message from the Receiver. If the reply is an  
118 appropriate Acknowledgement Message for the transferred message, Sender may remove  
119 the transferred message from Sender's persistent storage if the message is no longer  
120 needed for some other Messaging Service function or later failure recovery.

121 The Receiver's Messaging Service sends an Acknowledgement Message to the Sender's  
122 Messaging Service for every Normal Reliable Messaging message received. There is no reply to  
123 the Acknowledgement message from the Sender's Messaging Service and the Sender's  
124 Messaging Service will not initiate a new message transmission until a current message  
125 transmission has been correctly received. In the Acknowledgement Message:

- 126 • The **MessageType** SHALL be "Acknowledgement".
- 127 • There is no Payload and no business level response information.

128 **2.1.4 Reliable Messaging Recovery Procedures [MS section 7.6, added]**

129 **2.1.4.1 Messaging Service Parameters**

130 In Reliable Messaging, the Sending Messaging Service uses the following Messaging Service  
 131 parameters during recovery procedures.

132 This information may be determined in a number of ways, such as the TPA or some other method.

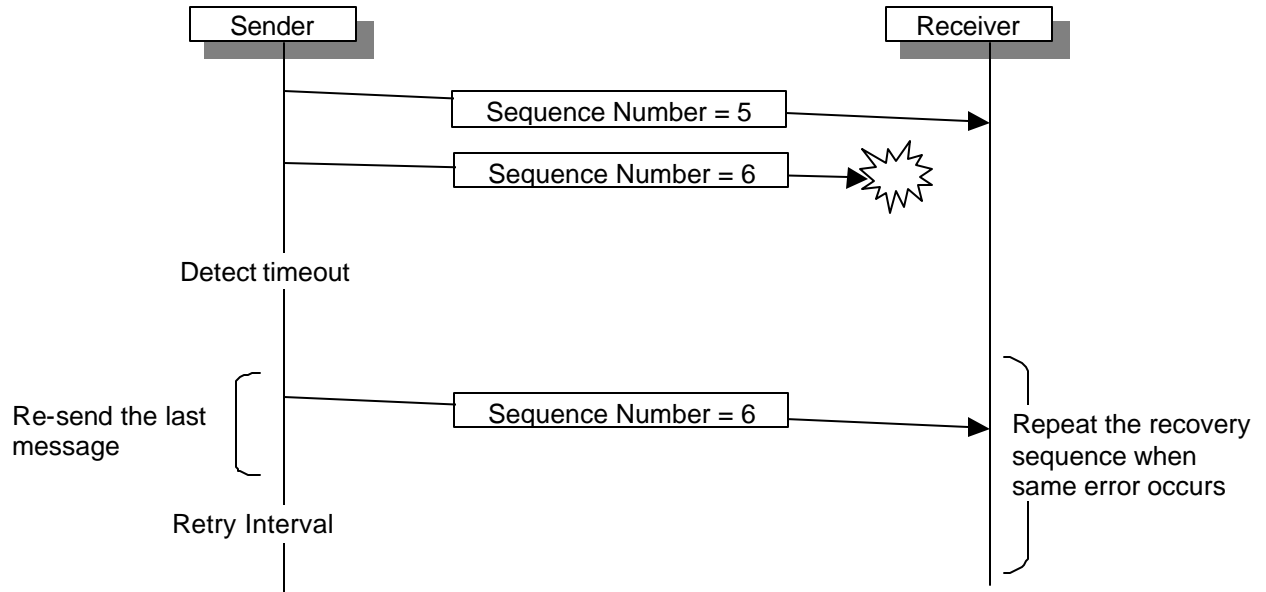
133 **Table 2-1 Messaging Service Parameters used in Recovery**

<b>Argument</b>	<b>Outline Description</b>
<b>Timeout</b>	Wait time for any response from the Receiver. <ul style="list-style-type: none"> <li>• Integer value specifying a number of seconds</li> <li>• After sending a Normal Message, the Sender SHALL wait for any response (MS Acknowledgement or Error Message) for the specified time before start of retry</li> </ul>
<b>Retries</b>	Maximum number of retries. <ul style="list-style-type: none"> <li>• Integer value specifying the number of retries</li> <li>• The Sender SHALL repeat retries the specified number of times until the Sender receives an MS Acknowledgement Message</li> <li>• If the Sender does not receive an MS Acknowledgement Message after the maximum number of retries, the Sender SHALL notify the incident to the higher level (application and/or system admin)</li> </ul>
<b>RetryInterval</b>	Wait time between retries, if an Acknowledgement Message is not received <ul style="list-style-type: none"> <li>• Integer value specifying a number of seconds</li> <li>• After a retry, the Sender SHALL wait for a response (MS Acknowledgement or Error Message) for specified time before start of the next retry</li> </ul>

134 **2.1.4.2 Recovery Sequence for Lost Messages**

135 When the Sender detects a timeout while waiting for an Acknowledgement Message from the last  
 136 sent message, the appropriate recovery handler in the Sender executes a Messaging Service  
 137 recovery sequence. The timeout value is defined as **Timeout**. This recovery sequence SHALL re-  
 138 send the final message to the Receiver and SHALL use a retry interval the **retry** number of times.  
 139 The content of the re-sent message is exactly the same as the original message. In the recovery  
 140 sequence or after the recovery sequence,

- 141 • If the Sender does not receive any error message or Acknowledgment Message in the  
 142 retry interval, the recovery handler repeats the recovery sequence the **retry** number of  
 143 times.
- 144 • If the Sender detects or receives another Error Message, the recovery handler executes  
 145 the appropriate recovery sequence for the error.
- 146 • If the Sender receives an Acknowledgment Message in the recovery sequence, the  
 147 message transmission is completed.



148  
149

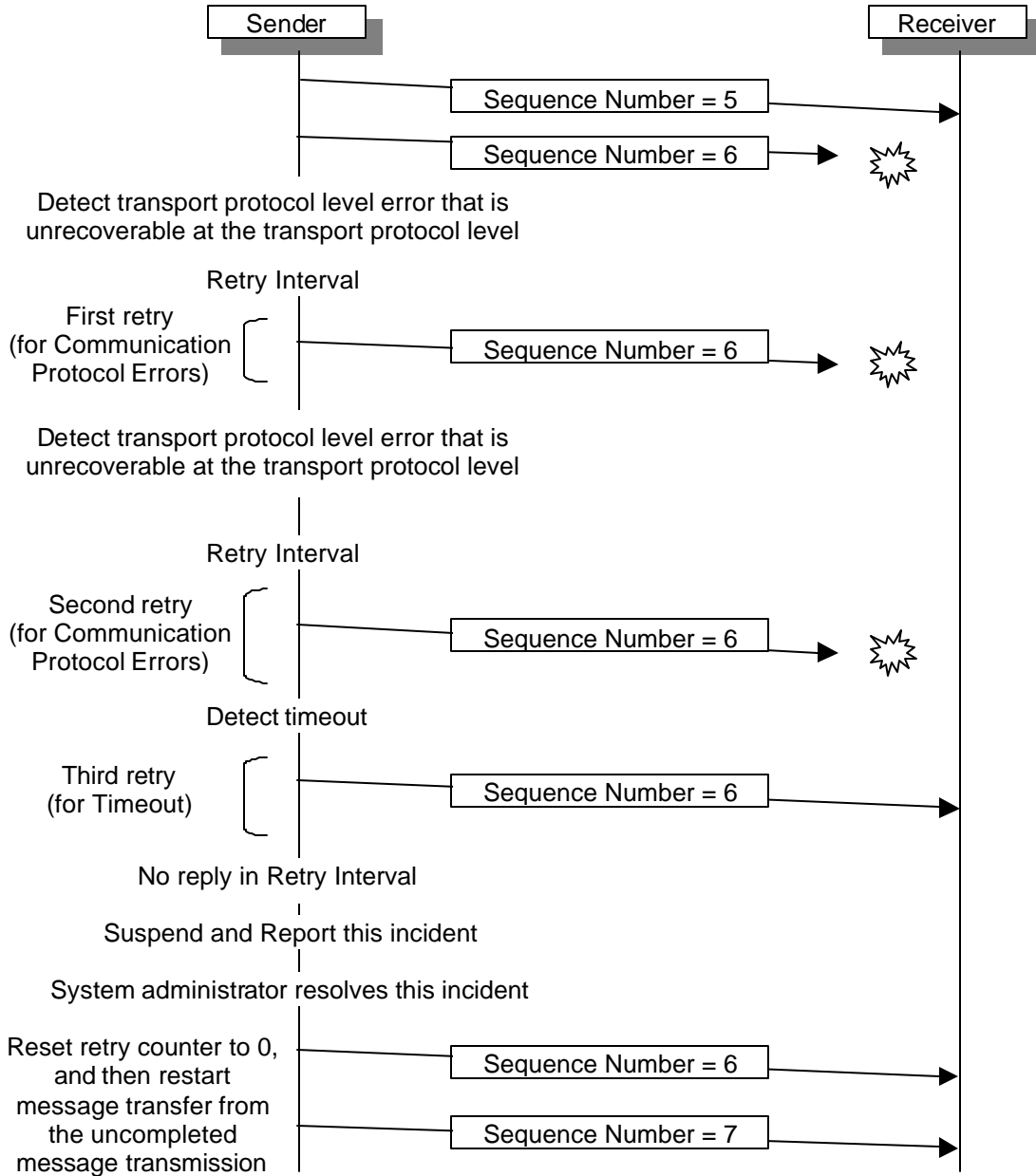
**Figure 2-5: Recovery Sequence for Timeout**

150 **2.1.4.3 Maximum Number of Retries and Retry Interval**

151 The retry interval is defined as **RetryInterval**. When the total number of retries in a reliable  
152 message transmission reaches a maximum number, defined as **Retries**, and the last error is still  
153 not resolved, the recovery handler will:

- 154 (1) Suspend sending messages to the Receiver  
155 (2) Report this incident to a higher-level so that a system administrator can resolve this incident

156 When the system administrator resolves the incident, the recovery handler will reset the retry  
157 counter to zero and then re-start message transfer sequence from the uncompleted reliable  
158 message transmission.



159

160 **Figure 2-7: Repeat of Recovery Sequence (specified maximum number of retries is 3)**

161 **2.2 ebXML Header Document [MS section 8]**

162 **2.2.1 ReliableMessagingInfo [MS section 8.4.4]**

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



- 167 • "OnceAndOnlyOnce" – reliable messaging semantics: the receiving Service Interface handler
- 168 will receive a given message no more than once, the sending Messaging Service will execute
- 169 retry procedures in the event of failure and the sending Service Interface handler will be
- 170 notified in the event of failure.
  
- 171 • "BestEffort" – reliable delivery semantics are not specified: no Acknowledgement Message is
- 172 returned to the Sender, duplicate messages might be delivered and persistent storages are
- 173 not required.

```
174 <ReliableMessagingInfo>
175   <DeliverySemantics>OnceAndOnlyOnce</DeliverySemantics>
176 </ReliableMessagingInfo>
```

177 **2.2.2 Routing Header Document [MS section 8.5, added]**

178 One **RoutingHeader** element immediately follows the **Header** element. It is required in all

179 **ebXMLHeader** documents. The **RoutingHeader** element is a composite element comprised of at

180 least the following 4 required subordinate elements:

- 181 • **SenderURI** – the Sender’s Messaging Service Handler URI.
- 182 • **ReceiverURI** – the Receiver’s Messaging Service Handler URI.
- 183 • **ErrorURI** – URI designated by the Sender for reporting errors.
- 184 • **Timestamp** – timestamp of the **RoutingHeader** creation, in the same format used for
- 185 **Timestamp** in the **XML Header MessageData** element.

186 When the **RoutingHeader** is used for a message sent with Reliable Messaging functions

187 (**DeliverySemantics** is set to “OnceAndOnlyOnce” in the **XML Header ReliableMessagingInfo**

188 element), the Sender SHALL add one additional **RoutingHeader** element to the **RoutingHeader**:

- 189 • **SequenceNumber** – Integer value that is incremented (e.g. 1, 2, 3, 4..) for each Sender-
- 190 prepared message sent to the Receiver. The Sequence Number consists of ASCII numerals
- 191 in the range 1-999,999,999. In following cases, the Sequence Number takes the value “1”:
  
- 192 a) First message from the Sender to a particular Receiver
- 193 b) First message after wraparound (next value after 999,999,999)
- 194 c) First message after removing Sequence Number information in the Sender (Sender
- 195 MAY remove Sequence Number information when it has no messages which were sent
- 196 to the Receiver for long time).

197 *Editor Note 2: “a long time” needs more precision. Should there be a way for the Sender to*

198 *notify the Receiver that the Sequence Number count has been reset?*

199 The following fragment demonstrates the structure of the **RoutingHeader** element of the

200 **ebXMLHeader** document when Reliable Messaging is used:

```
201 <RoutingHeader>
202   <SenderURI>...</SenderURI>
203   <ReceiverURI>...</ReceiverURI>
204   <ErrorURI>...</ErrorURI>
205   <Timestamp>...</Timestamp>
206   <SequenceNumber>...</SequenceNumber>
207 </RoutingHeader>
```

208 The **Header** structures in an Acknowledgement Message SHALL have at least the following

209 element values, which are obtained from the message being acknowledged:

- 210 • **From = ReceiverURI** as shown in the Routing Header Document



- 211 • **To** = **SenderURI** as shown in the Routing Header Document
- 212 • **TPAId** and **ConversationID** as shown in the Header Document
- 213 • **ServiceInterface** and **Action** are empty
- 214 • **RefToMessageId** = **MessageId** of the reliable message
- 215 • **DeliverySemantics** = “BestEffort”

## 216 2.3 Schema and DTD Definitions [MS Appendix A]

217 *MS Editor – need to add appropriately...*

218

219

DTD:

```
220 <!ELEMENT RoutingHeader (SenderURI , ReceiverURI , ErrorURI, Timestamp,  
221     SequenceNumber )>  
222 <!ELEMENT SenderURI (#PCDATA )>  
223 <!ATTLIST SenderURI e-dtype NMTOKEN #FIXED 'uri' >  
224 <!ELEMENT ReceiverURI (#PCDATA )>  
225 <!ATTLIST ReceiverURI e-dtype NMTOKEN #FIXED 'uri' >  
226 <!ELEMENT ErrorURI (#PCDATA )>  
227 <!ATTLIST ErrorURI e-dtype NMTOKEN #FIXED 'uri' >  
228 <!ELEMENT TimeStamp (#PCDATA )>  
229 <!ATTLIST TimeStamp e-dtype NMTOKEN #FIXED 'dateTime' >  
230 <!ELEMENT SequenceNumber (#PCDATA )>
```

## 231 2.4 Examples [MS Appendix B]

232 *MS Editor – need to add appropriately...*

233

234

Sample:

```
235 <RoutingHeader>  
236 <SenderURI>  
237     http://www.sender_company.com/ebxmlhandler/  
238 </SenderURI>  
239 <ReceiverURI>  
240     http://www.receiver_company.com/ebxmlhandler/  
241 </ReceiverURI>  
242 <ErrorURI>  
243     http://www.sender_company.com/ebxmlerrorhandler/  
244 </ErrorURI>  
245 <Timestamp>  
246     19991110T102344.000Z  
247 </Timestamp>  
248 <SequenceNumber>  
249     000000023  
250 </SequenceNumber>  
251 </RoutingHeader>
```

252

## 253 2.5 Communication Protocol Interface [MS Appendix E]

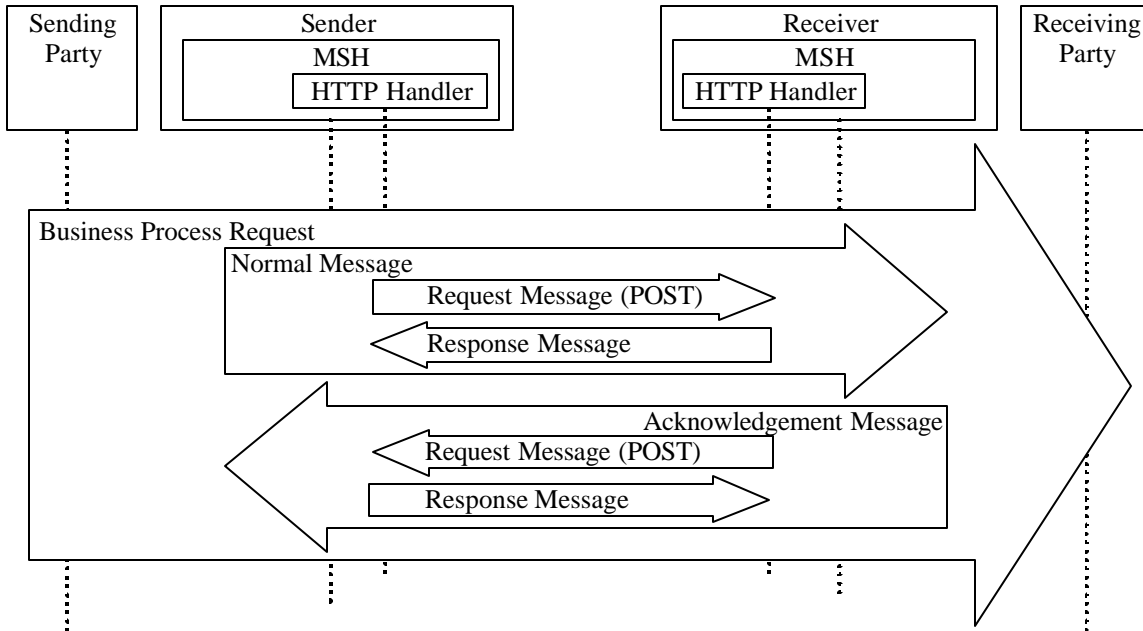
254 The ebXML Messaging Service messages are carried by Transport Protocols as shown in the  
255 following sections.

256 **2.5.1 HTTP**

257 All ebXML Messaging Service messages are carried by an HTTP Request Message (POST  
 258 method). The HTTP Response Message to an HTTP Request Message has no entity body.

259 The following Figure x.x shows how a Normal Message and its corresponding Acknowledgement  
 260 Message (when Reliable Messaging is used) are carried using HTTP:

261 **Figure x.x HTTP Flow**



262

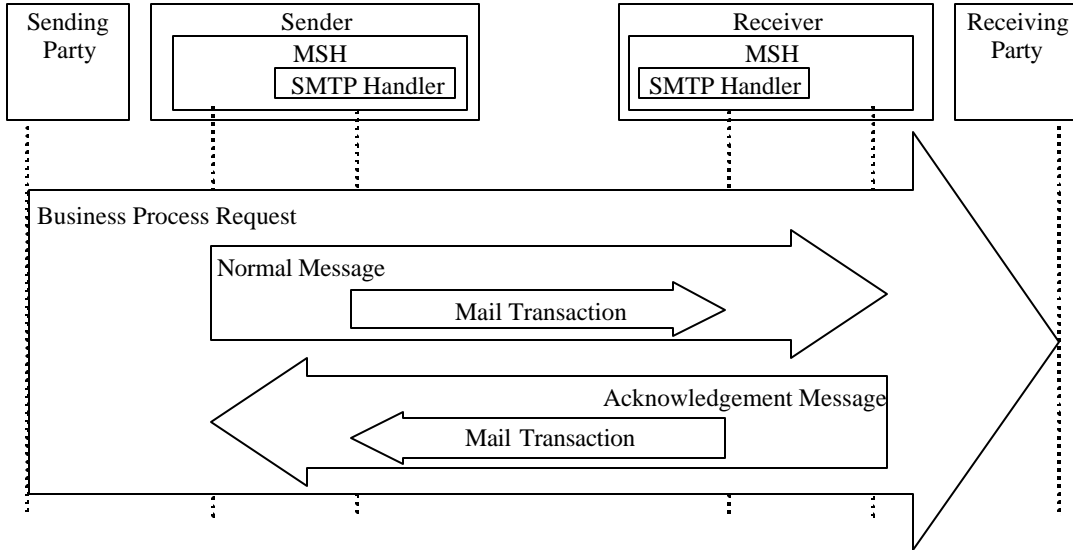
263 **Table x-x Relationship with HTTP**

<b>ebXML Messaging Service message</b>	<b>HTTP</b>
<b>Normal Message</b>	<ul style="list-style-type: none"> <li>Request Message (POST method) from Sender to Receiver</li> <li>Response Message to the Request Message has no entity body</li> </ul>
<b>Acknowledgement Message</b>	<ul style="list-style-type: none"> <li>Request Message (POST method) from Receiver to Sender</li> <li>Response Message to the Request Message has no entity body</li> </ul>
<b>Error Message</b>	<ul style="list-style-type: none"> <li>Request Message (POST method) from Receiver to Sender</li> <li>Response Message to the Request Message has no entity body</li> </ul>

264 **2.5.2 SMTP**

265 All ebXML Messaging Service messages are carried as mail in an SMTP Mail Transaction as  
 266 shown in the following Figures.

267 **Figure x.x SMTP Flow**



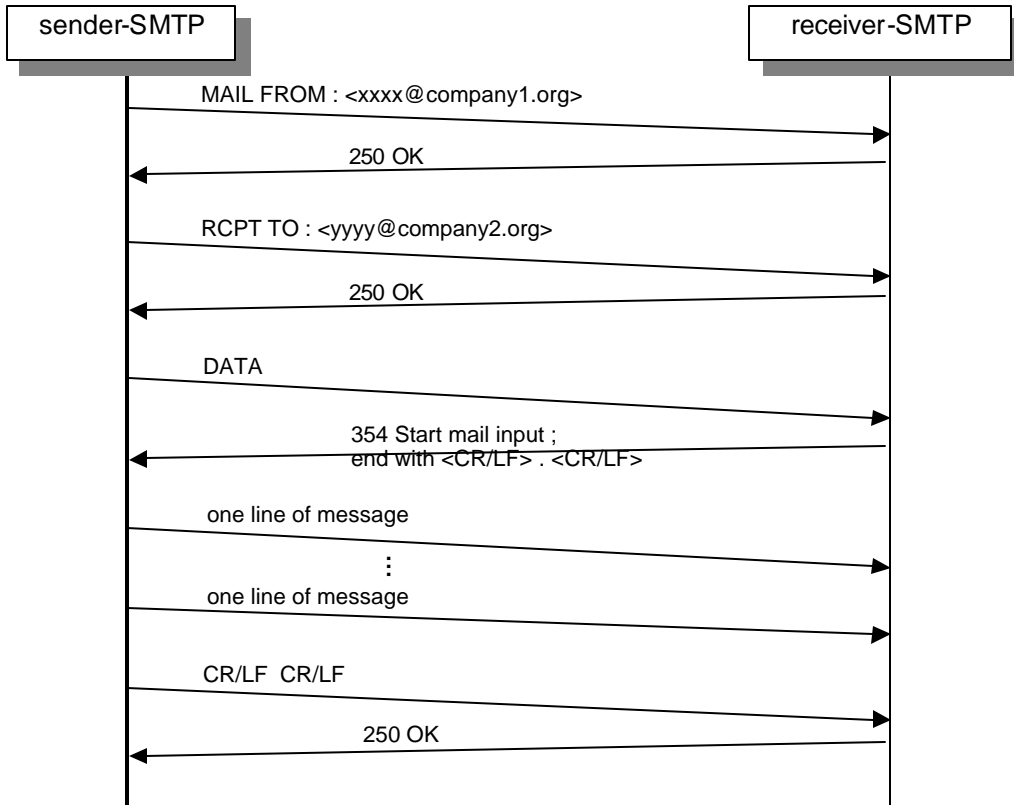
268

269

270 The Mail Transaction follows RFC 821, "SIMPLE MAIL TRANSFER PROTOCOL", as shown in  
 271 the following Figure:

272

Figure x.x SMTP Sequence



273

274

Table 2-2 Relationship with SMTP

<i>ebXML Messaging Service message</i>	<i>SMTP</i>
<b>Normal Message</b>	Mail Transaction from Sender to Receiver
<b>Acknowledgement Message</b>	Mail Transaction from Receiver to Sender
<b>Error Message</b>	Mail Transaction from Receiver to Sender

275

**2.5.3 FTP**

276

[TBD]

277

**2.5.4 Communication Protocol Errors during Reliable Messaging**

278

When the Sender or the Receiver detects a transport protocol level error (such as an HTTP,

279

SMTP or FTP error), the appropriate transport recovery handler will execute a recovery

280

sequence. No Reliable Messaging functions are involved in this recovery sequence, since it

281

happens at a lower level.

282

However, if the Sender detects a transport protocol level error that is unrecoverable at the

283

transport protocol level, the appropriate recovery handler in the Sender will execute a Messaging

284

Service recovery sequence. This recovery sequence SHALL use a retry interval and SHALL re-

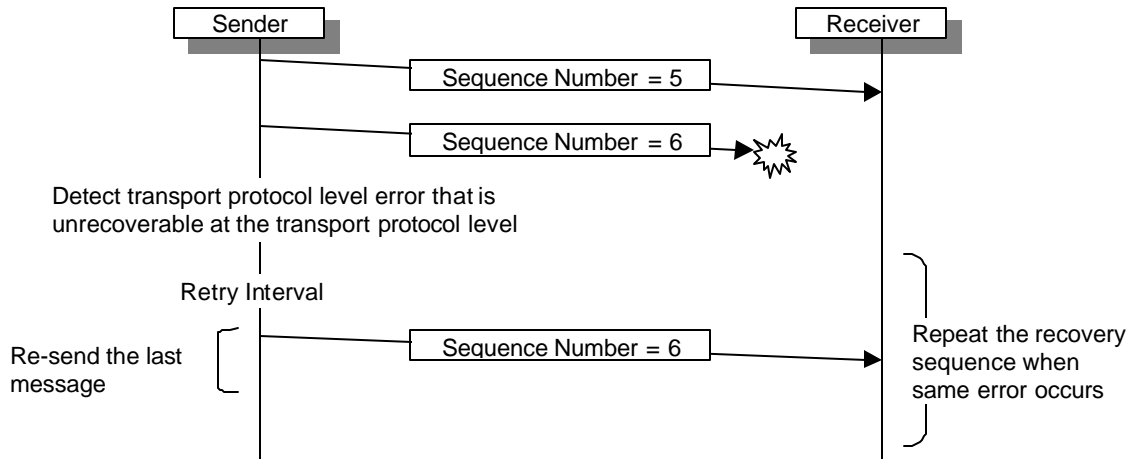
285

send the last message to the Receiver. The format of the re-sent message is exactly the same as

286

the original message. In the recovery sequence or after the recovery sequence:

- 287
- 288
- 289
- If the Sender detects a transport protocol level error again, which is unrecoverable at the transport protocol level, the recovery handler repeats the recovery sequence for an implementation-defined number of times.
- 290
- If the Sender detects or receives another error, the recovery handler executes an appropriate recovery sequence for the error.
- 291
- If the Sender receives an Acknowledgment Message, the message transmission is completed.
- 292
- 293



294

295

**Figure x.x: Recovery Sequence for Communication Protocol Errors**

### 296 3 Modifications to ebXML Glossary

297 The following items should be placed in the approved ebXML Glossary.

#### 298 3.1 Reliable Messaging Terms

##### 299 3.1.1 Once And Only Once

300 A message delivery semantic that means:

- 301
- Message delivery is guaranteed under most circumstances, and the Sending Party will be notified if there is no delivery.
- 302
- A message will always reach the Receiving Party no more than once.
- 303
- If a message does not reach the Receiving Party, the Sending Party does not need to execute retry procedures (retry is automatically executed by the messaging service).
- 304
- 305

##### 306 3.1.2 At Most Once

307 A message delivery semantic that means:

- 308
- Message delivery is not guaranteed
- 309
- A message will always reach the Receiving Party no more than once.
- 310
- If a message is not delivered, the Sending Party can detect the incident, and if the Sending Party wants to guarantee message delivery, the Sending Party must execute retry procedures
- 311
- 312

313 **3.1.3 Best Effort**

314 A message delivery semantic that means:

- 315 • Message delivery is not guaranteed
- 316 • A message will always reach the Receiving Party no more than once.
- 317 • If a message does not reach the Receiving Party, the Sending Party can not detect the
- 318 incident

319

320 **4 Phase 2/Phase 3 Activities**

321 Material in this section will be discussed in future TRP meetings as a likely base for further  
322 additions to the Messaging Service specification.

323 **4.1 Reliable Routing**

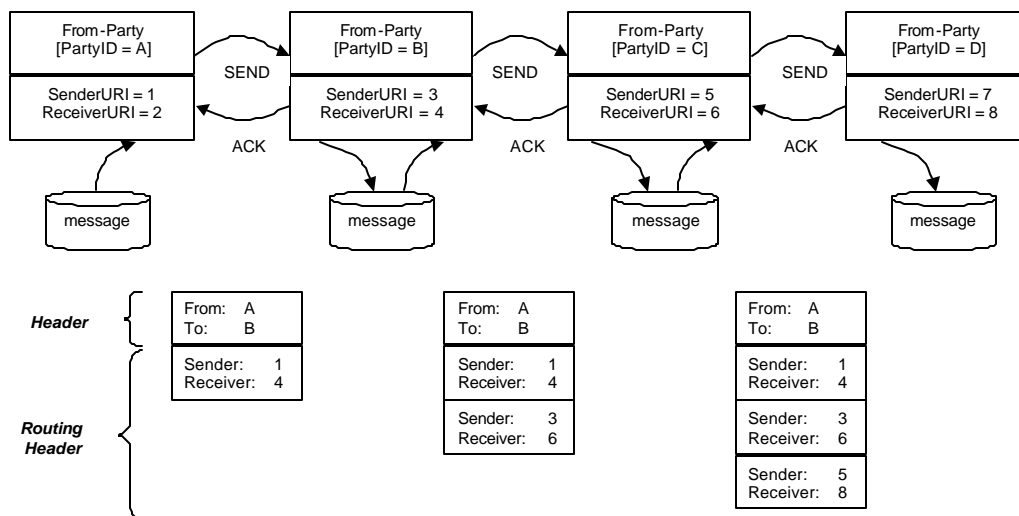
324 **4.1.1 Store and Forward Semantics**

325 Reliable Routing consists of a series of individual simple Reliable Messaging transmissions  
326 between a Sender and a Receiver. These *Store and Forward* semantics consist of the following  
327 sequence:

- 328 (1) Sender A transfers a message to Receiver B using Reliable Messaging.
- 329 (2) After completion of the reliable messaging transmission between Sender A and Receiver B,  
330 Sender B transfers the received message to Receiver C using Reliable Messaging.
- 331 (3) After completion of the reliable messaging transmission between Sender B and Receiver C,  
332 Sender C transfers the received message to Receiver D using Reliable Messaging.
- 333 (4) [Repeat until end of routing]

334

**Figure 4-1 Reliable Routing**



335

336



337 **4.1.2 Routing Information**

338 The first Sender (From-Party's Sender) specifies the From/To elements in the Header, and the  
339 SenderURI/ReceiverURI elements in the Routing Header for first message transferred to the  
340 Router.

341 When the message is forwarded between Routers, the Router adds a new **RoutingHeader** to the  
342 message with new SenderURI/ReceiverURI elements for message forwarding to the next Router.

343 **4.1.3 Error Handling in Routing**

344 When message forwarding to a subsequent Router is not available or fails from a particular  
345 Router, and if that Router received the message from previous Sender, the Router's Receiver  
346 returns an ErrorMessage (Transient Error) to the Sender instead of an Acknowledgement  
347 Message. By this rule, the Sending Party will not receive a Messaging Service acknowledgement  
348 of successful transmission until the Receiving Party's Message Service Handler has actually  
349 received and stored the message.

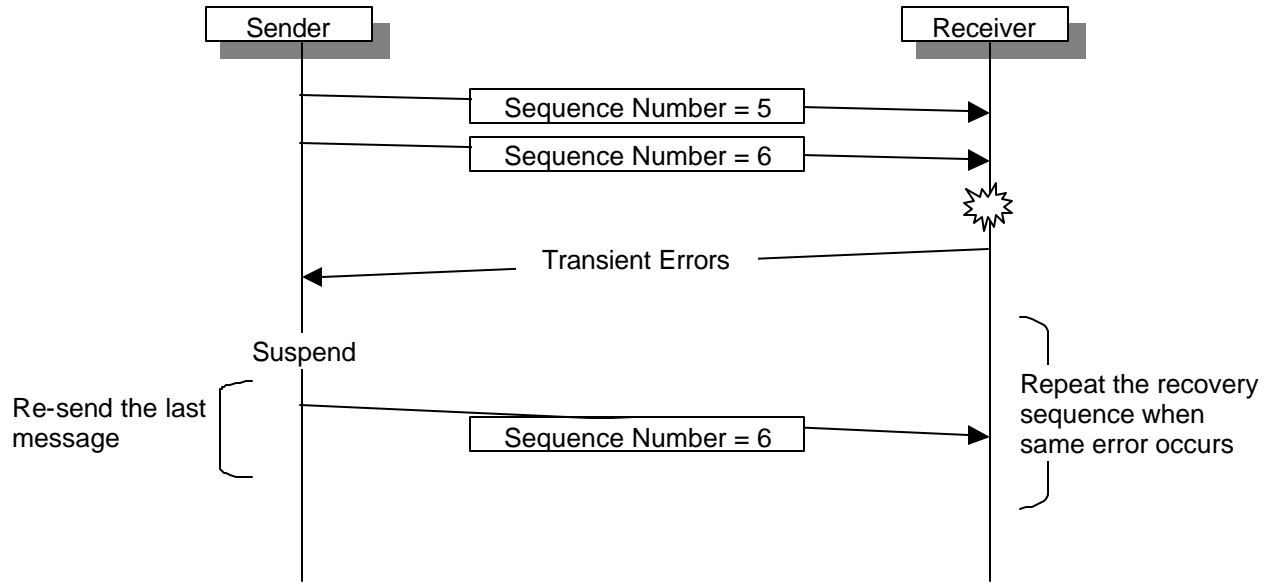
350 **4.2 Transient Errors**

351 When the Sender receives the error message "Transient Error", the appropriate recovery handler  
352 in the Sender executes a Messaging Service recovery sequence. The recovery sequence SHALL  
353 suspend sending of further messages to the Receiver for the period specified in the  
354 **MinRetrySecs** field in the error message. If the **MinRetrySecs** field does not exist in the error  
355 message, **RetryInterval** specified in TPA or elsewhere is used as the suspending time.

356 After the suspension, the Sender's recovery handler SHALL re-send the last sent message to the  
357 Receiver. The format of the re-sent message is exactly the same as the original message. In the  
358 recovery sequence or after the recovery sequence,

- 359 • If the Sender receives the error message "Transient Errors" again, the recovery handler  
360 repeats the recovery sequence.
- 361 • If the Sender detects or receives another error, the recovery handler executes the  
362 appropriate recovery sequence for the error.
- 363 • If the Sender receives an Acknowledgment Message, the message transmission is  
364 completed.

365



366  
367 **Figure 4-6: Recovery Sequence for Transient Errors**

## 368 **5 Non-normative Material**

369 The majority of the material in this section should remain in a non-normative Implementer's Guide.

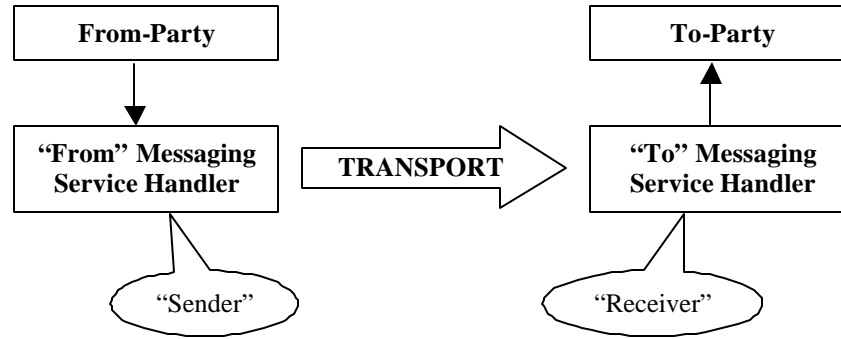
370 However, since the three recovery parameters (*Timeout*, *Retries* and *RetryInterval*) used by the  
371 Sender for Reliable Messaging Recovery have not yet been formally defined in Phase 1, the  
372 portions of this section 5 relating to recovery remain in this non-normative portion of the  
373 specification. In the next phase of specification development, these parameters will be defined  
374 formally and it is expected that the recovery process will be moved into the normative portion of  
375 the Messaging Services specification.

### 376 **5.1 Basic Concepts**

377 To achieve reliable messaging between Parties, this specification defines a process that enables  
378 the Parties' ebXML Messaging Services to communicate with each other using "Once and Only  
379 Once" semantics, coupled with a timeout to determine lost messages.

380 For the purposes of this document, the term "*Sender*" means the Sending Party's Messaging  
381 Service that sends the message on the underlying message transport, and "*Receiver*" means the  
382 Messaging Service used by the Receiving Party. The term "*From-Party*" means the party that  
383 originally prepared the message and provided the message to its Messaging Service, and the  
384 term "*To-Party*" means the party that was identified by the From-Party as the final recipient of the  
385 message.

386 For example, a simple message transmission using two Message Service Handlers and one  
387 transport is shown in Figure 2-1.



388

389

**Figure 2-1: Simple Message Transmission**

390 Reliable Messaging consists of the following basic concepts:

- 391 1) Messages are sent and received through Messaging Service Handlers (MSH), which function  
 392 on behalf of their respective Parties (and Business Processes). With respect to a particular  
 393 underlying transport, each MSH can be identified as a “Sender” or a “Receiver”.
- 394 2) A message is identified by its **MessageId** field, which is contained in the Message Header’s  
 395 **MessageData** element created by the Sender.
- 396 3) When the From-Party requests Reliable Messaging semantics for the message, the Sender  
 397 sets the **DeliverySemantics** field in the **ReliableMessagingInfo** element of the Message  
 398 Header to “OnceAndOnlyOnce”.
- 399 4) Reliable Messaging processing requires no changes to the Message Header during  
 400 transmission, once the Message Header is prepared.
- 401 5) Reliable Messaging uses a “Routing Header” contained in the Message Envelope.
- 402 6) A Reliable message indicated by setting the **DeliverySemantics** field to **OnceAndOnlyOnce**.
- 403 7) For each reliable message, the Sender generates a **Sequence Number** that is unique to the  
 404 MSH Sender-Receiver pair. For subsequent reliable messages, the Sender increments the  
 405 Sequence Number placed in that message. The **Sequence Number** is contained in the  
 406 Routing Header Data Element.
- 407 8) A Messaging Service level Acknowledgement is sent from the Receiver to the Sender for  
 408 every received message with a message type of Normal after persisting the message.
- 409 9) Within a reliable message transmission, the Receiver must determine whether a received  
 410 message is a duplicate message. Two possible approaches are through using the  
 411 **MessageId** and/or the Sender-Receiver unique **Sequence Number**. If the received message  
 412 is a duplicate, the Receiver discards the message after sending the acknowledgement. If the  
 413 message is not a duplicate, the Receiver stores the message in its persistent storage, sends  
 414 an acknowledgement and delivers the message to a higher processing level.
- 415 10) Because every message received with Reliable Messaging semantics will cause the sending  
 416 of a related Acknowledgement Message, the Sender must be prepared to discard duplicate  
 417 Acknowledgement Messages if multiple copies of the original message are sent.
- 418 11) To detect loss of a reliable message, the Sender sets a timeout, retry interval and number of  
 419 retries for that message. If the transmitted reliable message is lost due to system or  
 420 communication failure, the Sender will re-send this message using these parameters before  
 421 reporting failure to the From-Party. These values might be specified in the Trading Partner  
 422 Agreement (TPA) or some other fashion.



## 423 5.2 Detection of Repeated Messages by the Receiver

424 Detection of repeated messages in the Receiver using **Message Identifiers** and/or **Sequence**  
425 **Numbers** is implementation dependent.

426 Comparison of Message Identifiers could be used to detect duplicated messages. Another  
427 effective detection logic can be suggested which uses **Sequence Numbers**, which are unique to  
428 a particular Sender-Receiver pair.

429 The Receiver receives the reliable message and then compares the received reliable message's  
430 **Sequence Number** with the immediately previous reliable message's **Sequence Number**:

431 (1) Received message's **Sequence Number** == Previous message's **Sequence Number** + 1

432 The message is not a repeated message. The Receiver stores the message into persistent  
433 storage, processes the message appropriately and returns an Acknowledgement Message.

434 (2) Received message's **Sequence Number** == Previous message's **Sequence Number**

435 The message is a repeated message. The Receiver discards the message and returns  
436 Acknowledgement Message.

437 When both the received message's **Sequence Number** and the previous message's  
438 **Sequence Number** are 1, the Receiver will have to use the **MessageId** instead of a  
439 Sequence Number to check for a repeated message. This situation happens only when  
440 following sequence occurs (this is a very unusual situation):

441 a) The Sender sends only one message to the Receiver, and it's the first message for the  
442 Sender-Receiver pair. There is no subsequent message for long time.

443 b) Since there is no other message that should be sent to the Receiver for long time, the  
444 Sender might remove its **Sequence Number** information for the Receiver. However, the  
445 Receiver does not remove its **Sequence Number** information for the Sender.

446 After that, the Sender is given a message from the sending Party that should be sent to the  
447 Receiver. Since the Sender's **Sequence Number** information was previously removed, the  
448 Sender sends the message to the Receiver using **Sequence Number** 1.

449 (3) Received message's **Sequence Number** == 1 && not case (2)

450 The message is not a repeated message. The Receiver stores the message into persistent  
451 storage, processes the message appropriately and returns an Acknowledgement Message.

452 (4) Any other case

453 It means that the Sequence Number value is invalid. The Receiver discards the message and  
454 returns an error message.

## 455 6 Acknowledgements

456 The author wishes to acknowledge the members of the ebXML TR&P who commented on  
457 Fujitsu's proposal in the face-to-face meetings and in e-mail.

## 458 7 Author's Address

459 Masayoshi Shimamura  
460 Fujitsu Limited  
461 Shinyokohama Nikko Bldg., 15-16, Shinyokohama 2-chome  
462 Kohoku-ku, Yokohama 222-0033, Japan  
463 Telephone: +81-45-476-4590  
464 E-mail: shima@rp.open.cs.fujitsu.co.jp