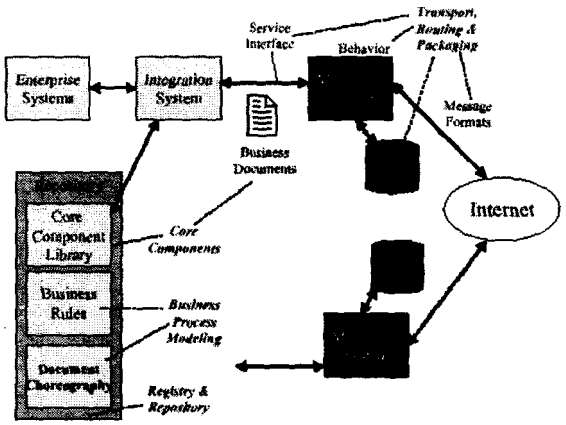
ebXML Transport, Routing & Packaging Specification

소프트웨어기술연구소
[illegible]연구부 전자거래연구팀
박찬규
ETRI
한국전자통신연구원

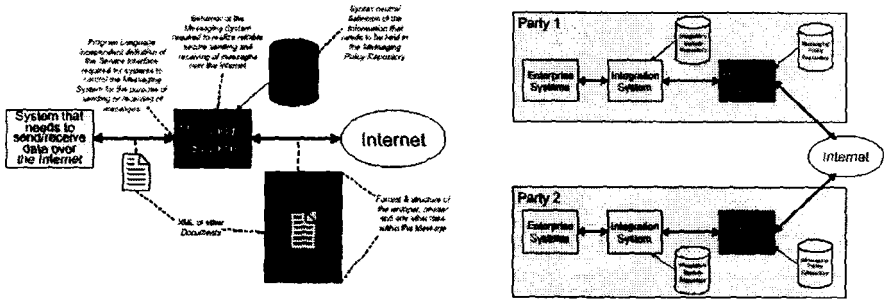
ebXML TR&P Related Specification

- ebXML *Message Services Requirements Specification*
  - Working Draft 26-May-2000, v0.96
- ebXML *Message Services Specification*
  - Working Draft 13-Sep-2000, v0.21
- ebXML *Collaboration Protocol Profile and Agreement Specification*
  - Working Draft 24-Jan-2001, v0.29
- ebXML *Technical Architecture Security Specification*
  - Working Draft 5-Feb-2001, v1.02(Under development)
- ebXML *Message Service Interface Specification*
  - Under development

Relationships with other ebXML activities



Scope of Transport, Routing and Packaging Activities



Message Service Specification
ebXML Transport, Routing & Packaging

01/02/2001 Version 0.92

ebXML TR&P Message Service Specification

What ?

- Defines the ebXML Message Service protocol that enables the secure and reliable exchange of messages between two parties.
- Defines robust, yet basic, functionality to transfer messages using various existing communication protocols.
- One of the three "infrastructure" components of ebXML that includes: Registry/Repository, Collaboration Protocol Profile/Agreement and the ebXML Message Service

ebXML TR&P Message Service Specification

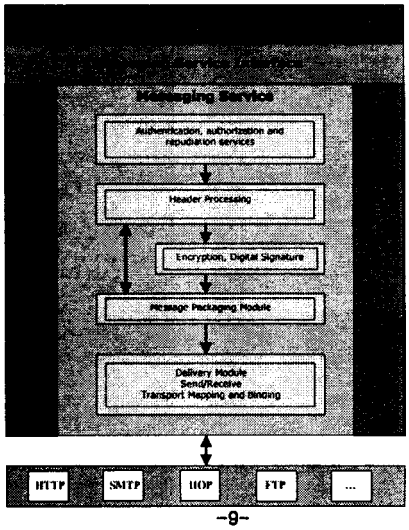
This spec is organized around the following topics.

1. Packaging Specification
   - How to package an ebXML Message and its associated parts into a form that can be placed into the body of a communications protocol such as HTTP or SMTP
2. Message Headers
   - The Structure & composition of the information necessary for an ebXML Message Service to successfully generate or process an ebXML compliant message.
3. Message Service Handler Services
   - A description of two services that enable one service to discover the status of another Message Service Handler or an individual message

ebXML TR&P Message Service Specification

4. Reliable Messaging
   - Defines an interoperable protocol such that any two Message Service implementation can "reliably" exchange messages that are sent using "reliable messaging" semantics
5. Error Handling
   - Describes how one ebXML Message Service reports errors it detects to another ebXML Message Service Handler
6. Security
   - Provides a complete spec of the security requirements for ebXML Messages

System Overview of Message Service

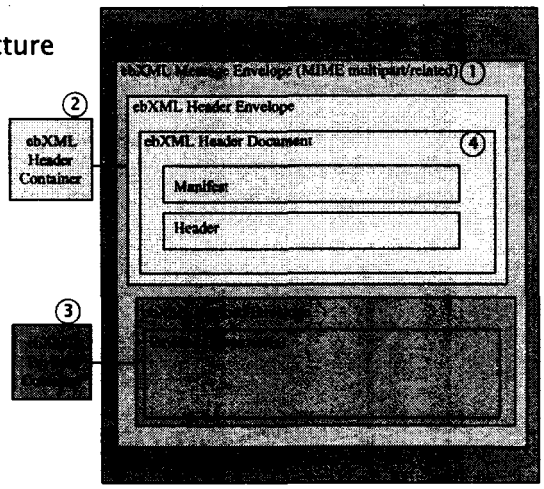


1. Packaging Specification

- An ebXML Message consists of:
  - An outer communication Protocol Envelope, such as HTTP or SMTP
  - An inner communication "protocol independent" ebXML Message Envelope, specified using MIME multipart/related, that contains the two main parts of the Message:
    - An ebXML Header Container that is used to envelop one ebXML Header Document, and
    - An optional, single ebXML Payload Container that MUST be used to envelop the actual payload(transferred data) of the message

2. Message Header

Message Structure

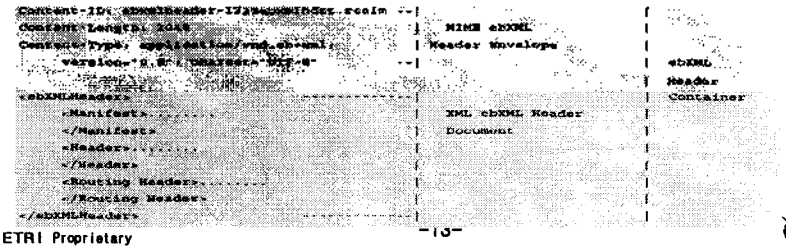


ebXML Message Envelope - ①

- Used to identify the message as an ebXML compliant structure and encapsulates the header and payload body parts.
- Must conform to [RFC2045] and must contain two MIME headers.
  - Content-Type
  - Content-Length
- example

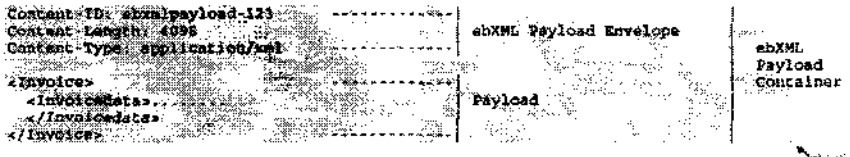
ebXML Header Container - ②

- MIME body part that must consist of
  - One ebXML Header Envelope
    - Conforms to [RFC2045] and must consist of three MIME headers
      - Content-ID, Content-Length, Content-Type
    - Must be the first MIME body part in the ebXML Message
  - One ebXML Header Document
- example



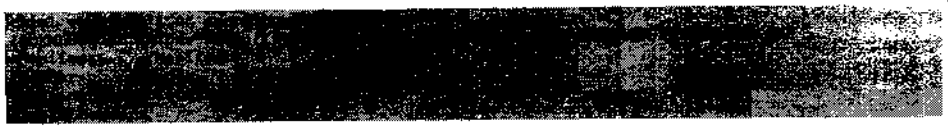
ebXML Payload Container - ③

- If the ebXML Message contains a payload, then a single ebXML Payload Container MUST be used to envelop it
- If an ebXML Payload Container is present then it MUST conform to [RFC2045] and MUST consist of
  - A MIME header portion - the ebXML Payload Envelope
  - A Content portion - the payload itself



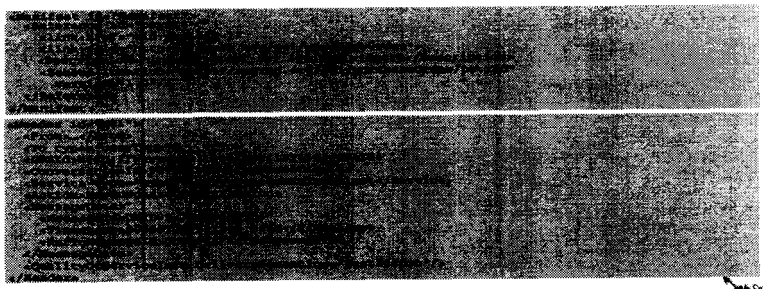
ebXML Header Document - ④

- Single XML document
- Root Element
  - ebXMLHeader attributes -Namespace, Version
  - ebXMLHeader sub elements
    - Manifest, Header, RoutingHeaderList, ApplicationHeaders, StatusData, ErrorList, Acknowledgement, Signature, #wildcard



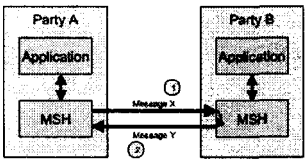
ebXML Header Document - ④

- Manifest
  - Contains a list of references to other parts the Message
  - Includes references to Payload Message
- Header
  - Contains the information required by the recipient to process the message.

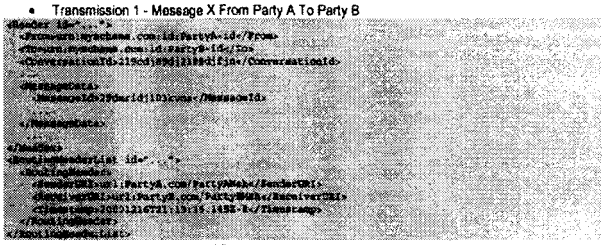


ebXML Header Document - ④

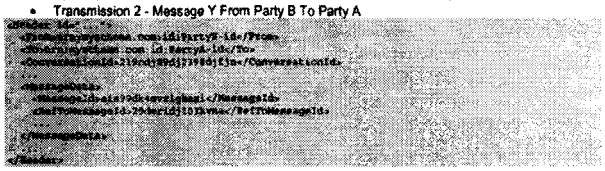
- RoutingHeaderList - Single Hop Routing Header Sample



- Transmission 1 - Message X From Party A To Party B

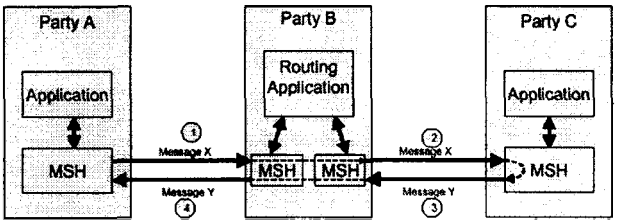


- Transmission 2 - Message Y From Party B To Party A



ebXML Header Document - ④

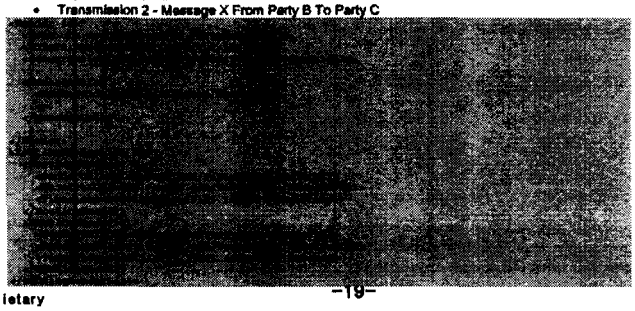
- RoutingHeaderList - Single Hop Routing Header Sample



ebXML Header Document - ④

- Transmission 1 - Message X From Party A To Party B

- Transmission 2 - Message X From Party B To Party C



3.Message Service Handler Services

To help provide smooth operation of a message Handling Service implementation

- Message Status Request
  - Sending a Message Status Request to a MSH about a message previously sent
  - The MSH that receives the request sending a Message Status Response message in return.
- Message Service Handler Ping
  - Enables one MSH to determine if another MSH is operating
  - Sending a Message Service Handler Ping message to a MSH
  - The MSH that receives the Ping responding with a Message Service Handler Ping message

4. Reliable Messaging

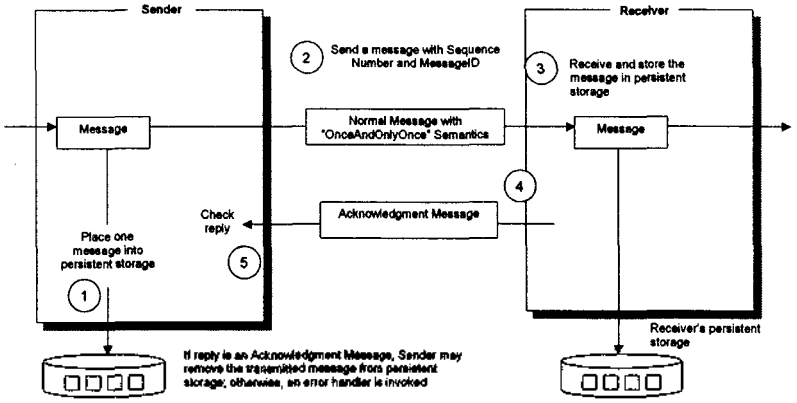
- Defines an interoperable protocol such that two MSH operated by a *From Party* and a *To Party* can "reliably" exchange messages that are sent using "reliable messaging" semantices.
- "Reliably" means that the *From Party* can be highly certain that the message sent will be delivered to the *To Party*. If there is a problem in sending a message then the sender resends the message until either the message is delivered, or the sender gives up.
- If the message can't be delivered, for example because of a catastrophic failure of the *To Party's* system, then the *From Party* is informed.
- A MSH that supports Reliable Messaging MUST keep messages that are sent or received reliably in *persistent storage*(a method of storing data that does not lose information after a system failure or interruption).

Reliable Messaging Flow

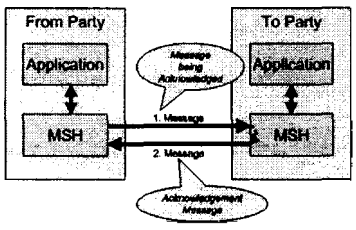
Sending Message Behavior

1. Create message from components received from the application that includes
   - *deliverySemantics* set to *OnceAndOnlyOnce*, and
   - A *RoutingHeader* element : sender and receiver URIs
2. Save the message in *persistent storage*
3. Send the message to the *Receiver* MSH
4. Wait for the *Receiver* MSH to return an *ack message* and, if it doesn't, then resend the identical message
5. If the message doesn't need to be sent reliably, then *deliverySemantics* MUST be set to *BestEffort*(the default)

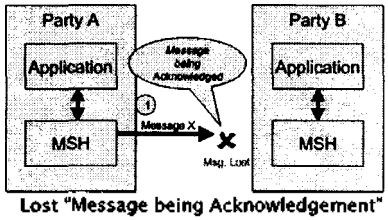
Reliable Message Transfer Sequence



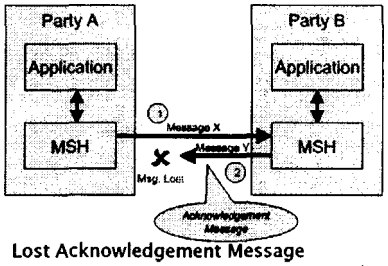
Single-hop Reliable Messaging



Normal Case

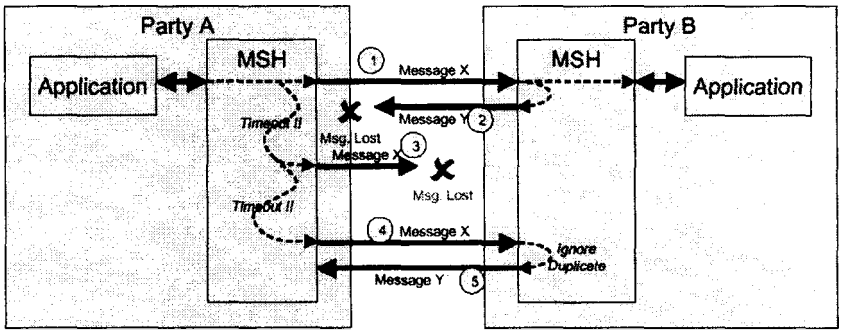


Lost "Message being Acknowledgement"

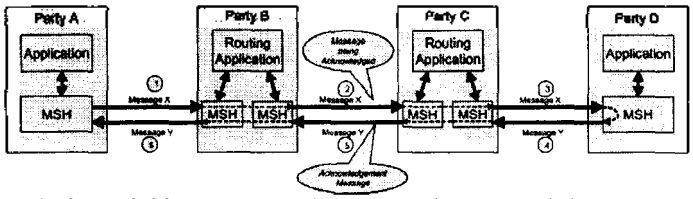


Lost Acknowledgement Message

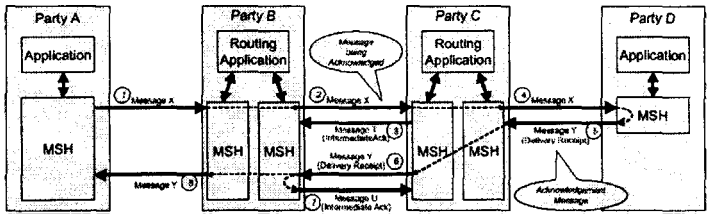
Resending Lost Messages



Multi-hop Reliable Messaging

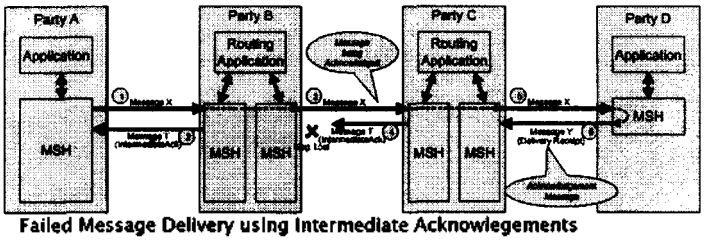


Multi-hop Reliable Messaging without Intermediate Acknowledgments

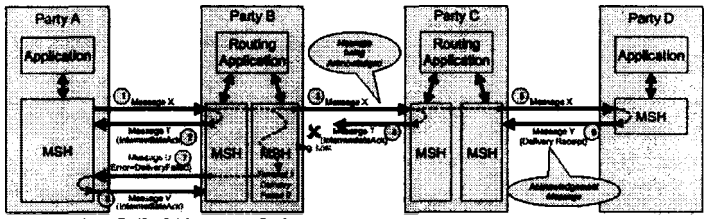


Multi-hop Reliable Messaging with Intermediate Acknowledgments

Failed Message Delivery : Multi-hop



Failed Message Delivery using Intermediate Acknowlegements



Reporting Failed Message Delivery

Message Service Parameters

Specified By	Parameter	CPA/ CPP	Message Header	Routing Header
From Party	deliverySemantics	Yes	Yes	N/A
From Party	deliveryReceiptRequested	Yes	Yes	N/A
From Party	syncReplyMode	Yes	Yes	N/A
From Party	timeToLive	Yes	Yes	N/A
To Party	deliveryReceiptProvided	Yes	No	No
Sending MSH	reliableMessagingMethod	No	N/A	Yes
Sending MSH	intermediateAckRequested	No	N/A	Yes
Sending MSH	timeout	Yes	No	No
Sending MSH	retries	Yes	No	No
Sending MSH	retryInterval	Yes	No	No
Receiving MSH	reliableMessagingSupported	Yes	No	No
Receiving MSH	intermediateAckSupported	Yes	No	No
Receiving MSH	persistDuration	Yes	No	No
Receiving MSH	mshTimeAccuracy	Yes	No	No

5. Error Handling

- *message in error*
  - A message that contains or causes an error of some kind
- *message reporting the error*
  - A message that contains an ebXML ErrorList element that describes the error(s) found in a *message in error*
- Types of Errors
  - The structure or content of the Message Envelope(e.g.MIME)
  - The ebXML Message Header document
  - Reliable messaging failures, or
  - Security

Error Code

Error Code	Short Description	Long Description
UnableToParse	XML not well formed or invalid	The XML document is not well formed or not valid and cannot be successfully parsed. See [XML] for the meaning of "well formed" and "not valid".
ValueNotRecognized	Element content or attribute value not recognized.	Although the document is well formed and valid, the element/attribute contains a value that could not recognized and therefore could not be used by the ebXML Message Service
NotSupported	Element or attribute not supported	Although the document is well formed and valid, an element or attribute is present that • is consistent with the rules and constraints contained in this specification, but • is not supported by the ebXML Message Service that is processing the instance.
Inconsistent	Element content or attribute value inconsistent with other elements or attributes	Although the document is well formed and valid, according to the rules and constraints contained in this specification the content of an element or attribute is inconsistent with the content of other elements or their attributes.
OtherKind	Other error in an element content or attribute value.	Although the document is well formed and valid, the element content or attribute value contains values that do not conform to the rules and constraints contained in this specification and is not covered by other error codes. The *errorMessage* attribute should be used to indicate the nature of the problem.

Error Code	Short Description	Long Description
MessageTooLarge	Message too large	The message is too large to be processed by this ebXML Message Service.
MimeProblem	A MIME error has occurred	An error has been detected in the structure or format of a MIME part of the message. For example: • Missing MIME Part. Although the MIME message is correctly structured, a MIME part is missing that should have been present if the rules and constraints contained in this specification are followed • Unexpected MIME Part. Unexpected MIME part. Although the MIME message is correctly structured, a MIME part is present that is not expected in the particular context according to the rules and constraints contained in this specification
DeliveryFailure	Message Delivery Failure	A message has been received that either probably or definitely could not be sent to its next destination. Note that if *severity* is set to *Warning* then there is a small probability that the message was delivered.
TimeToLiveExpired	Message Time To Live Expired	A message has been received that arrived after the time specified in the *TimeToLive* element of the *MessageHeader* element
SecurityFailure	Message Security Checks Failed	Validation of signatures or checks on the authenticity or authority of the sender of the message have failed.
Unknown	Unknown Error	Indicates that an error has occurred that is not covered explicitly by any of the other errors. The *errorMessage* attribute should be used to indicate the nature of the problem.

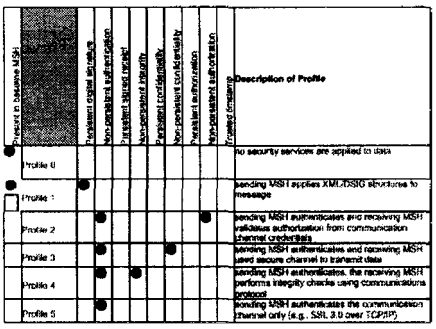
6. Security

- A Message Service may be at risk by means of
  - Unauthorized access
  - Data integrity and/or confidentiality attacks
  - Denial-of-Service, spoofing, bombing attacks
- Security and Management
  - No technology, regardless of how advanced it might be, is an adequate substitute to the effective application of security management policies and practices
- Collaboration Protocol Agreement
  - The configuration of Security for MSHs is specified in the CPA
  - Three areas of the CPA
    - Document Exchange section, Message section, Transport section

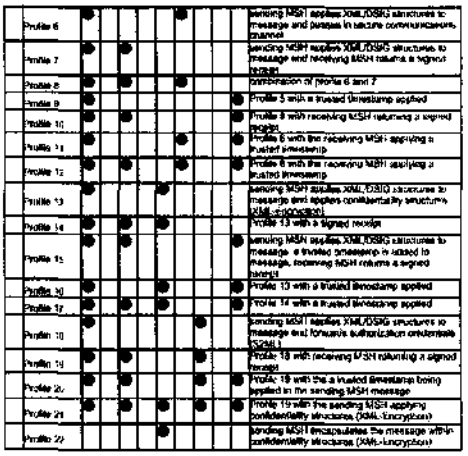
6. Security

- Countermeasure Technologies
  - Persistent Digital Signature - XML Signature
  - Persistent Signed Receipt - XML Signature
  - Non-persistent Authentication - TLS(SSL3.0), IPSEC
  - Non-persistent Integrity - TLS, SSL3.0(CRC check)
  - Persistent Confidentiality - XMLEncryption, S/MIME
  - Non-persistent Confidentiality - TLS, IPSEC
  - Persistent Authorization - OASIS Security Services TC
  - Non-persistent Authorization - TLS, IPSEC(certificate)
  - Trusted Timestamp

Security Profile



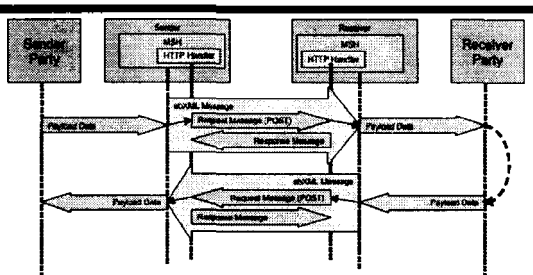
Present in baseline MSH	Persistent digital signature	Non-persistent authentication	Persistent signed receipt	Non-persistent integrity	Persistent confidentiality	Non-persistent confidentiality	Persistent authorization	Non-persistent authorization	Trusted timestamp	Description of Profile
●										No security services are applied to data
Profile 0										
●	●									sending MSH applies XML/DSIG structures to message
Profile 1										
	●							●		sending MSH authenticates and receiving MSH validates authorization from communication channel credentials
Profile 2										
	●				●					receiving MSH authenticates and receiving MSH used secure channel to transmit data
Profile 3										
	●		●							sending MSH authenticates, the receiving MSH performs integrity checks using communications protocol
Profile 4										
	●									sending MSH authenticates the communication channel only (e.g., SSL 3.0 over TCP/IP)
Profile 5										



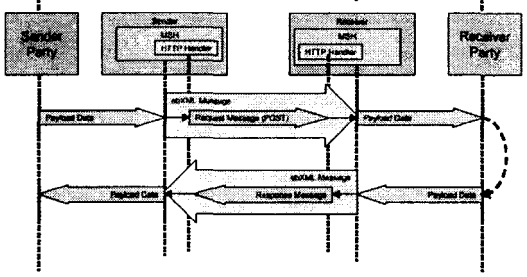
Profile 6	●			●						sending MSH applies XML/DSIG structures to message and passes in secure communications channel
Profile 7	●		●						●	sending MSH applies XML/DSIG structures to message and receiving MSH returns a signed receipt
Profile 8	●		●		●					combination of profile 6 and 7
Profile 9	●								●	Profile 5 with a trusted timestamp applied
Profile 10	●		●						●	Profile 9 with receiving MSH returning a signed receipt
Profile 11	●				●				●	Profile 8 with the receiving MSH applying a trusted timestamp
Profile 12	●		●		●				●	Profile 8 with the receiving MSH applying a trusted timestamp
Profile 13	●			●						sending MSH applies XML/DSIG structures to message and applies confidentiality structures (XML-Encryption)
Profile 14	●		●	●						Profile 13 with a signed receipt
Profile 15	●		●						●	sending MSH applies XML/DSIG structures to message, a trusted timestamp is added to message, receiving MSH returns a signed receipt
Profile 16	●			●					●	Profile 13 with a trusted timestamp applied
Profile 17	●		●	●					●	Profile 14 with a trusted timestamp applied
Profile 18	●					●				sending MSH applies XML/DSIG structures to message and forwards authorization credentials (SAML)
Profile 19	●		●			●				Profile 18 with receiving MSH returning a signed receipt
Profile 20	●				●	●			●	Profile 18 with the a trusted timestamp being applied in the sending MSH message
Profile 21	●		●		●	●			●	Profile 19 with the sending MSH applying confidentiality structures (XML-Encryption)
Profile 22				●						sending MSH encapsulates the message with confidentiality structures (XML-Encryption)

Appendix. Communication Protocol Interface: HTTP

Asynchronous HTTP Message Flow

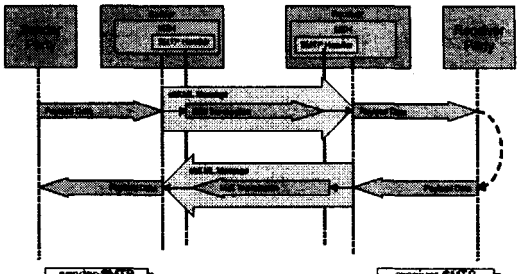


Synchronous HTTP Message Flow

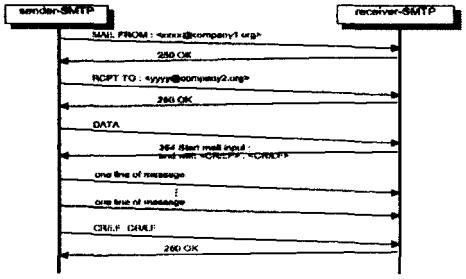


Appendix. Communication Protocol Interface: SMTP

SMTP Message Flow



SMTP Sequence



ETRI Proprietary

Reference

- http://www.ebxml.org/project_teams/technical_arch/private/
- http://www.ebxml.org/project_teams/transport/private/
- http://www.ebxml.org/project_teams/trade_partner/private/
- http://www.ebxml.org/mail-archives.htm