ebXML Encapsulation

Proposal to the DGI Interoperability Test Team

v4

11/27/01

David Fischer
DRUMMOND GROUP, Inc.

Table of Contents

1 Encapsulation

Some implementations may require that ebXML messages be Encapsulated into the payload of another ebXML message in a recursive fashion. This might be needed for encrypting a message where the headers need to be included in the encryption with the payloads. This might be useful when an intermediary node needs to add a payload to an existing message without disturbing a signature. This might also be used to break very large messages, or messages with large numbers of payloads, into smaller pieces for transmission (how this is done is outside the scope of this specification). These examples highlight some possible uses for Encapsulation but do not encompass all possible uses.

When the Encapsulation process is applied, the MSH shall put the original message with the normal Multipart/Related content-type, as the payload of another ebXML message. If required, this Multipart/Related MIME structure may be encrypted using an approved encryption process (XMLEncryption, S/MIME, etc.). A minimal set of headers shall be constructed, or copied from the original message, for this message with:

- a ***Service*** element set to ***urn:oasis:names:tc:ebxml-msg:service***
- an ***Action*** element set to ***Encapsulate***

These settings are NOT REQUIRED if the *Receiving MSH* can understand other settings and correctly process the Encapsulated message.

When the *Receiving MSH* parses this message, the Encapsulated payload should be reprocessed as a new message.

2 General Rules

Encapsulation can be used in a variety of situations, but there are some basic processes, which are common to most situations. These may be modified as necessary, particularly in cases where Encapsulation is used to provide more than one service, such as Encryption with an Intermediate ***TimeStamp***.

2.1 Creating an Encapsulated Message

An Encapsulated Message can be constructed by preceding the original Multipart/Related MIME header (the first MIME header after the transport headers but before the SOAP headers) with a new Multipart/Related MIME header and a new set of SOAP Headers as described below. This new Multipart/Related will require a new "boundary" value placed immediately after the MIME headers, immediately before the original Multipart/Related and at the end of the message.

2.1.1 SOAP Extension Headers

The new SOAP headers will be similar to the original message headers.

2.1.1.1 MessageHeader

In most cases, the ***MessageHeader To***/***From***, ***CPAId*** and ***ConversationId*** headers will not change, this would not be true in the case of Forwarding (see section 3.1). The ***Service***/***Action*** will be as specified in section 1, or as agreed between the parties. ***MessageData*** SHOULD contain a new ***MessageId*** and ***Timestamp***. In some cases, such as Encryption, it may be possible to use the original ***MessageId*** and ***Timestamp***; however, this has a potential for conflict and is not recommended.

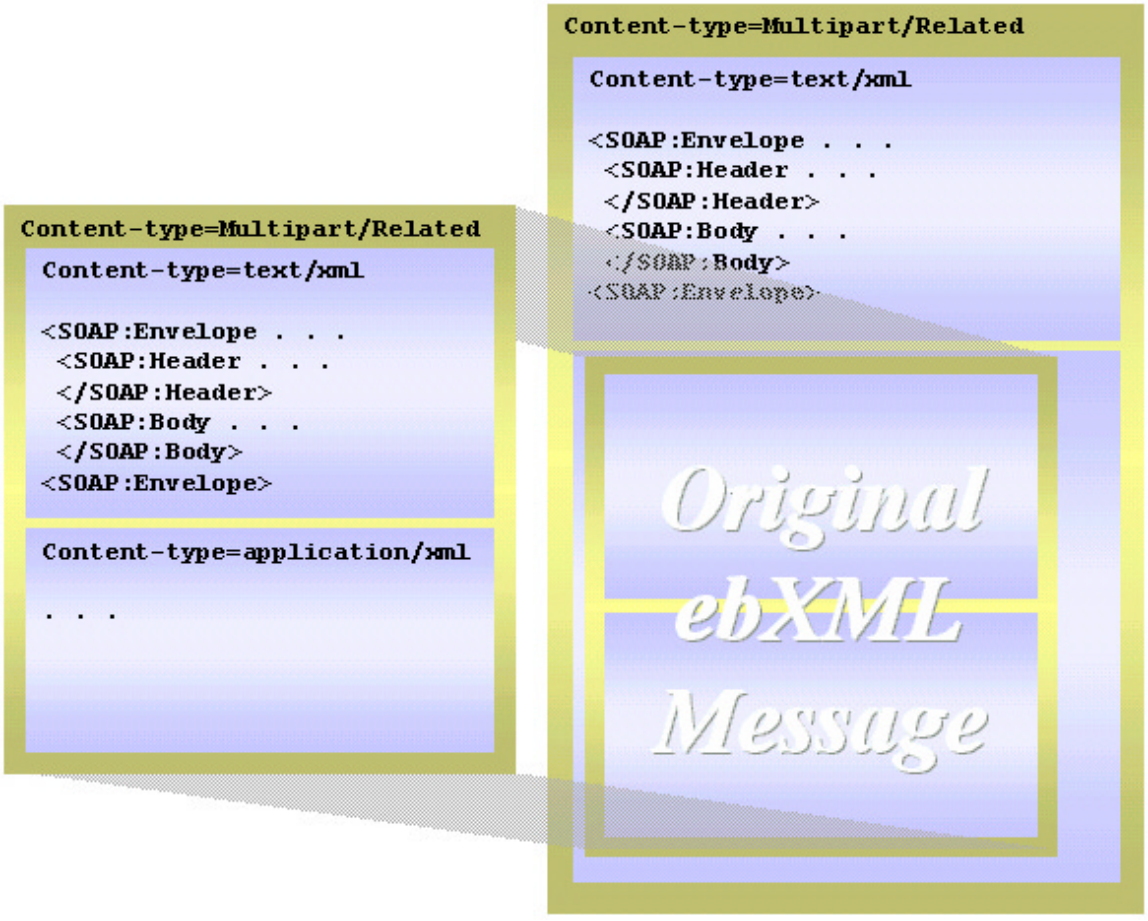

Figure 1 The original ebXML Message becomes the payload of another ebXML Message.

2.1.1.2 Manifest

The ***Manifest*** header will contain a single ***Reference*** element pointing to the Multipart/Related of the original message, or to the header where the original message is contained, as in the case of Encryption.

2.1.1.3 Other SOAP Extension Headers

Since this is a standard ebXML/SOAP message, any other ebXML headers could also be included, such as ***AckRequested***, ***ds***:***Signature***, etc. These should be processed as usual by the Receiving MSH prior to un-encapsulating the payload.

3 Profiles

The following examples provide some typical uses for Encapsulation although they do not represent all possible applications for Encapsulation.

3.1 Profile 1 – Forwarding

There are uses for Intermediary Nodes in the ebXML Message chain. In most cases, Forwarding may be accomplished by creating a message with the end recipient's ID or URI in the ***MessageHeader*** + ***To*** field and introducing the Intermediary's address into the Transport Header fields. This assumes that the Intermediary can correctly parse and forward based upon the MessageHeader fields. If this is not possible, or if the Intermediary cannot, or will not, accept and process a message intended for another recipient, such as may be the case if the Intermediary wishes to validate the signature, then Forwarding might be accomplished using the Encapsulation process. Using Encapsulation, a ***Signature*** element can be applied to the outside headers destined for the Intermediary using a signature key for which the Intermediary has the appropriate public-key.

3.1.1 Forwarding Rules

An Encapsulated message sent to an Intermediary, should parse the message, removing the encapsulation headers, creating and sending any requested MSH signals, then recreate a new set of ebXML message headers consistent with the requirements of the next hop. For the next hop, the sender MUST NOT leave the previous encapsulation headers on the message. The Sender and the Receiver/Intermediary should agree on a ***Service***/***Action*** pair consistent with the next MSH's environment, such as:

- a ***Service*** element set to ***urn:oasis:names:tc:ebxml-msg:service***
- an ***Action*** element set to ***Encapsulate-Forward***

3.1.2 Forwarding Example

```
. . . Transport Headers
SOAPAction: "ebXML"
Content-Type: Multipart/Related; boundary="Encapsulation-boundary1"

--Encapsulation-boundary1
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xlink="http://www.w3.org/1999/xlink"
               xmlns:eb="http://oasis-open.org/committees/ebxml-msg/schemas/msg-header-2_0.xsd">
  <SOAP:Header>
    <eb:MessageHeader eb:version="2.0" SOAP:mustUnderstand="1">
      <eb:From>
        <eb:PartyId>http://17.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">CompanyA</eb:PartyId>
      </eb:From>
      <eb:To>
```

```
        <eb:PartyId>http://18.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">CompanyB</eb:PartyId>
      </eb:To>
      <eb:CPAid>CompanyA-CompanyB</eb:CPAid>
      <eb:ConversationId>20011001-160101-00321</eb:ConversationId>
      <eb:Service>urn:oasis:names:tc:ebxml-msg:service</eb:Service>
      <eb:Action>Encapsulation-Forward</eb:Action>
      <eb:MessageData>
        <eb:MessageId>20011001-160101-003479@companyA.com</eb:MessageId>
        <eb:TimeStamp>2001-10-01T16:01:02</eb:TimeStamp>
      </eb:MessageData>
    </eb:MessageHeader>
  </SOAP:Header>
  <SOAP:Body>
    <eb:Manifest eb:version="2.0">
      <eb:Reference xlink:href="cid:Encapsulated-Payload"/>
    </eb:Manifest>
  </SOAP:Body>
</SOAP:Envelope>

--Encapsulation-boundary1
Content-ID: <Encapsulated-Payload>
Content-Type: Multipart/Related; boundary="ebXML-boundary1"

--ebXML-boundary1
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xlink="http://www.w3.org/1999/xlink"
               xmlns:eb="http://oasis-open.org/committees/ebxml-msg/schemas/msg-header-2_0.xsd">
  <SOAP:Header>
    <eb:MessageHeader eb:version="2.0" SOAP:mustUnderstand="1">
      <eb:From>
        <eb:PartyId>http://17.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">CompanyA</eb:PartyId>
      </eb:From>
      <eb:To>
        <eb:PartyId>http://19.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">CompanyC</eb:PartyId>
      </eb:To>
      <eb:CPAid>CompanyA-CompanyC</eb:CPAid>
      <eb:ConversationId>20011001-160101-00321</eb:ConversationId>
      <eb:Service type="Test">FileTransfer</eb:Service>
      <eb:Action>Receive</eb:Action>
      <eb:MessageData>
        <eb:MessageId>20011001-160101-003478@companyA.com</eb:MessageId>
        <eb:TimeStamp>2001-10-01T16:01:01</eb:TimeStamp>
      </eb:MessageData>
    </eb:MessageHeader>
  </SOAP:Header>
  <SOAP:Body>
    <eb:Manifest eb:version="2.0">
      <eb:Reference xlink:href="cid:Payload-1" xlink:type="simple">
        <eb:Description xml:lang="en-US">Test Data - XML File</eb:Description>
      </eb:Reference>
    </eb:Manifest>
  </SOAP:Body>
</SOAP:Envelope>

--ebXML-boundary1
Content-ID: "Payload-1"
Content-Type: text/xml

    <<Data omitted>>
--ebXML-boundary1--

--Encapsulation-boundary1--
```

3.2 Profile 2 – Signature/Payload Addition

There are some cases, which require an Intermediary to change the content of an ebXML message en route, such as the addition of a payload or the application of a ***TimeStamp***/***Signature***. These specific requirements can be accomplished by creating an Encapsulated message as previously described. The MSH/Application will create new Encapsulation headers, placing the original message in a payload as required. The sender should utilize appropriate ***Service***/***Action*** values, such as:

a ***Service*** element set to ***urn:oasis:names:tc:ebxml-msg:service***

- an ***Action*** element set to ***Encapsulate-TimeStamp***

3.3 Profile 3 – Encryption

It may be necessary in some instances to protect the SOAP Headers during transmission. Encapsulating an Encrypted message also solves problems with security order (sign before encrypting) and confusion with potentially application-encrypted payloads (how does the MSH know whether to decrypt a PKCS7 encrypted payload or to pass the payload to an application – encrypted payloads of a message with the Encapsulation ***Service***/***Action*** is always encrypted by the MSH).

3.3.1 Example

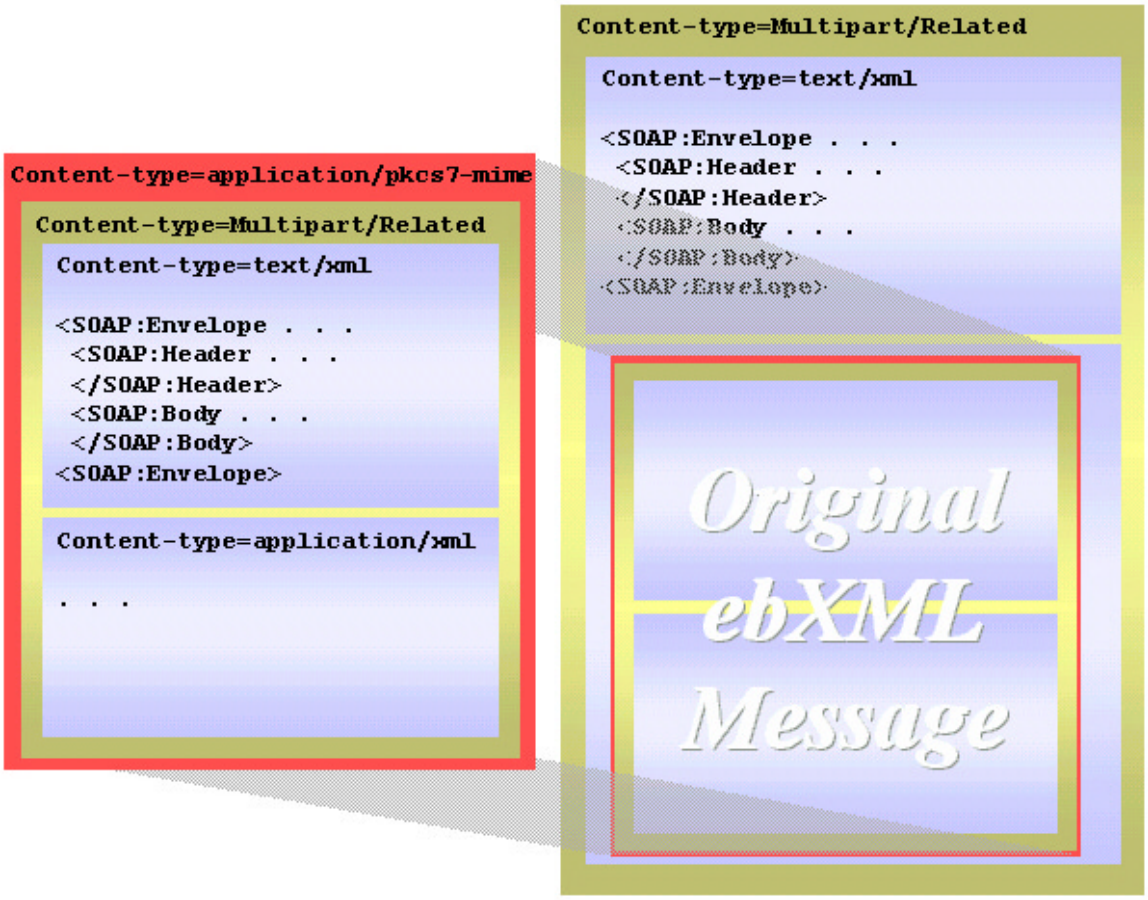

Figure 2 Encrypt before Encapsulate

3.3.2 Encryption Rules

Encryption via Encapsulation is accomplished by encrypting the entire ebXML message and encapsulating the result as the payload of another ebXML message. This allows the original payload and headers to be protected with only minimal routing headers visible. This requires an extra step in the encapsulating process to create a PKCS7-MIME object.

Once the message is encrypted, it becomes a single bodypart of a new ebXML message. The headers of this new message would be similar to the original message, with the same ***To***/***From*** and the same ***ConversationId*** and ***CPAId***. The message SHOULD have a new ***MessageId***, different from the original message, and it SHOULD have a ***Service*** and ***Action*** set as:

- a ***Service*** element set to ***urn:oasis:names:tc:ebxml-msg:service***
- an ***Action*** element set to ***Encapsulate-Encrypted***

The new headers will include a new ***Manifest*** with a single entry pointing to the encrypted payload.

3.3.3 Example Code

```
. . . Transport Headers
SOAPAction: "ebXML"
Content-Type: Multipart/Related; boundary="Encapsulation-boundary1"

--Encapsulation-boundary1
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xlink="http://www.w3.org/1999/xlink"
               xmlns:eb=" http://oasis-open.org/committees/ebxml-msg/schemas/msg-header-2_0.xsd">
  <SOAP:Header>
    <eb:MessageHeader eb:version="2.0" SOAP:mustUnderstand="1">
      <eb:From>
        <eb:PartyId>http://17.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">ZZCompanyA</eb:PartyId>
      </eb:From>
      <eb:To>
        <eb:PartyId>http://18.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">ZZCompanyB</eb:PartyId>
      </eb:To>
      <eb:CPAid>CompanyA-CompanyB</eb:CPAid>
      <eb:ConversationId>20011001-160101-00321</eb:ConversationId>
      <eb:Service>urn:oasis:names:tc:ebxml-msg:service</eb:Service>
      <eb:Action>Encapsulation-Encrypted</eb:Action>
      <eb:MessageData>
        <eb:MessageId>20011001-160101-003479@companyA.com</eb:MessageId>
        <eb:TimeStamp>2001-10-01T16:01:02</eb:TimeStamp>
      </eb:MessageData>
    </eb:MessageHeader>
  </SOAP:Header>
  <SOAP:Body>
    <eb:Manifest eb:version="2.0">
      <eb:Reference xlink:href="cid:Encapsulated-Payload"/>
    </eb:Manifest>
  </SOAP:Body>
</SOAP:Envelope>

--Encapsulation-boundary1
Content-ID: <Encapsulated-Payload>
Content-Type: application/pkcs7-mime

    <<Encrypted portion>>
```

```
Content-Type: Multipart/Related; boundary="ebXML-boundary1"

--ebXML-boundary1
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xlink="http://www.w3.org/1999/xlink"
               xmlns:eb="http://www.ebxml.org/namespaces/messageheader">
  <SOAP:Header>
    <eb:MessageHeader eb:version="1.0" SOAP:mustUnderstand="1">
      <eb:From>
        <eb:PartyId>http://17.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">ZZCompanyA</eb:PartyId>
      </eb:From>
      <eb:To>
        <eb:PartyId>http://18.3.67.128/ebXML/receive.dll:8080</eb:PartyId>
        <eb:PartyId type="Name">ZZCompanyB</eb:PartyId>
      </eb:To>
      <eb:CPAid>CompanyA-CompanyB</eb:CPAid>
      <eb:ConversationId>20011001-160101-00321</eb:ConversationId>
      <eb:Service type="Test">FileTransfer</eb:Service>
      <eb:Action>Receive</eb:Action>
      <eb:MessageData>
        <eb:MessageId>20011001-160101-003478@companyA.com</eb:MessageId>
        <eb:TimeStamp>2001-10-01T16:01:01</eb:TimeStamp>
      </eb:MessageData>
    </eb:MessageHeader>
  </SOAP:Header>
  <SOAP:Body>
    <eb:Manifest eb:version="2.0">
      <eb:Reference xlink:href="cid:Payload-1" xlink:type="simple">
        <eb:Description xml:lang="en-US">Test Data - XML File</eb:Description>
      </eb:Reference>
    </eb:Manifest>
  </SOAP:Body>
</SOAP:Envelope>

--ebXML-boundary1
Content-ID: "Payload-1"
Content-Type: text/xml

    <<Data omitted>>

--ebXML-boundary1--
    <<Encrypted portion>>
--Encapsulation-boundary1--
```

Note: *Code lines in* ***bold*** *are readable. Code lines not in bold are encrypted and non-readable.*