INTRODUCTION

Editor's Note. Editor's notes like this provide additional explanation. They are non-normative and will eventually need to be removed.

Editor's note. This section is a re-work of section 7.9 in version 0.21d of the ebXML Messaging Services specification. If we agree, then it could replace that section in its entirety. The changes that have been made (and their reasons) are as follows:

- ?? *Rework TPAInfo to additionally support the use case when a message is sent without a previously agreed "TPA". The main result of this is that ...*
- ?? *Replace TPA Id by either: a CPA Reference (equivalent to the TPA ID but reflects the terminology in the TP spec), a Party Profile Reference or neither. The idea of having neither is that if two parties always communicate in the same way (e.g. the Gas Industry) then specifying the TPAId (or CpaReference) is superfluous.*
- ?? *Change Service Interface to Service. This will avoid confusion with the definition of Service Interface in the TP spec and is the same as the name in the TRP Overview and Requirements spec*
- ?? *Move Service and Action up a level to the header element – the rationale is a) the other "TPA Info" like information becomes optional and b) it places some of the main information that is used for routing at a higher level.*
- ?? *Allow the From and the To parties to each have their own values for Conversation Id and CPA Reference. This as Marty Sachs has said, and I agree, makes it easier to tie a message back to a system running in the background*
- ?? *Make the definitions of From and To contain multiple elements so that all the elements that contain values specific to a party are in one place. This means that From (or To) now contains: PartyId, ConversationId, CpaReference and/or PartyProfileReferences. There is an alternative to this structure that keeps the information all in on place which I think is harder to understand. Both versions of the DTDs are included at the end of this paper.*
- ?? *PartyId has been re-worked as a URI by default so that it agrees with the approach I think we all agreed to on the list*

... now for the replacement text ....

David Burdett, Commerce One, October 23, 2000.

7.9 XML Header

The ***Header*** element immediately follows the ***Manifest*** element. It is required in all ***ebXMLHeader*** documents. The ***Header*** element is a composite element comprised of the following required subordinate elements:

- ?? ***From***
- ?? ***To***
- ?? ***Service***
- ?? ***Action***
- ?? ***MessageData***
- ?? ***ReliableMessagingInfo***

7.9.1 From and To

The ***From*** element contains information about the *Party* which originated the message. The ***To*** element contains information for the *Party* that is the intended recipient of the message. Both the ***From*** and the ***To*** have similar structures and contain the following elements:

- ?? ***PartyId*** - required

- ?? ***ConversationId*** – required on ***From***, optional on ***To***
- ?? one or neither of:
  - ***CpaReferences***, or
  - ***PartyProfileReference***

7.9.1.1 Party Id

PartyId is a logical identifier that identifies the *From Party* or the *To Party*. By default it takes the form of a URI. It is a REQUIRED attribute of both the ***From*** and the ***To*** elements.

The ***PartyId*** has a single attribute: ***type***. It is optional. If it is not present, the content of the ***PartyId*** element MUST contain a [URI]. If it is present then it MUST contain the value `UserDefined` and the identification of the *Party* represented by the ***PartyId*** MUST be understood by the recipient of a message using a method that is outside the scope of this specification.

The following fragments contain examples of the ***PartyId*** element using a URI.

```
<PartyId>MAILTO:joe@example.com</PartyId>
<PartyId>URN:duns.com:3210987654321</PartyId>
```

The following fragment demonstrates usage using the ***type*** attribute.

```
<PartyId type="UserDefined">AH-2745920</PartyId>
<PartyId type="UserDefined">18291-129012-HD</PartyId>
```

7.9.1.2 Conversation Id

The ***ConversationId*** identifies a set of related *Messages* exchanged between two or more *Parties*. It is called a *conversation* as the exchange of messages is similar to a conversation that two or more individuals may have when talking with one another. It SHOULD be used by the recipient of a *Message* to identify data previously stored about the *conversation* by the recipient. It is RECOMMENDED that the ***ConversationId*** is unique within the ***PartyId*** of the *Party* that first created it.

The ***ConversationId*** within the ***From*** element is required on every *Message*. It is set by the *From Party*. It is the identifier by which the *From Party* identifies a *conversation*.

The ***ConversationId*** within the ***To*** element is required on every *Message* apart from the first *message* sent within a *conversation*. It is the identifier by which the *To Party* identifies a *conversation*. If a *Message* is being sent to a *Party*, as a result of receiving an earlier *Message* from that *Party*, then the value of the ***ConversationId*** in the *To Party* MUST be set to the value of the *ConversationId* in the ***From*** element of the *message* received earlier.

The following fragment contains examples of the ***ConversationId*** element.

```
<ConversationId>47923-12310-23423-12312</ConversationId>
```

7.9.1.3 CPA Reference

The ***CpaReference*** element identifies a *Collaborative Processing Agreement* (see the ebXML Trading Partner Specification [ebXMLTP]). A Collaborative Processing Agreement (CPA) is an agreement between two (or more) *Parties* that describes how those *parties* will exchange messages in order to carry out a business process or service.

The ***CpaReference*** SHOULD be unique within the combination of the *From* ***PartyId*** and the *To* ***PartyId***.

The ***CpaReference*** in the ***From*** element contains an identifier or reference by which the sender of a *message* identifies the CPA.

The ***CpaReference*** in the *To* element contains an identifier or reference by which the recipient of a *message* identifies the CPA.

The ***CpaReference*** in the ***From*** and the ***To*** elements may be the same, for example if both the sender and receiver of a *message* have agreed on the same reference number.

There is an error if the recipient of a *message* does not:

1) recognize the ***CpaReference*** in the ***To*** element, or

2) recognizes the ***CpaReference*** in the ***To*** element but it does not identify a CPA that the *From Party* and *To Party* have agreed to.

If an error occurs then the recipient of the message must report the error by sending an error message with a ***Severity*** of `Error` back to the sender of the message.

The following fragment contains an example of ***CpaReference***.

```
<CpaReference>as93je0p9-asio32jsd-r3osej-asd3</CpaReference>
```

Editor's Note. We might pre-define a number of CpaReferences that identify "standard' CPAs that every MSH MUST implement for boot-strapping purposes. For example we could define: "ebXML:CPA1", "ebXML:CPA2" etc. If we do, then we should reserve CpaReference values that start with ebXML. Thoughts?

7.9.1.4 Party Profile Reference

The ***PartyProfileReference*** element contains a reference to a *Party Profile* document (see the ebXML Trading Partner Specification [ebXMLTP]). A *Party Profile* is a document that describes how a *party* "can" send or receive *messages* in order to carry out a business process or service.

The ***PartyProfileReference*** SHOULD be a [URI]. A ***PartyProfileReference*** SHOULD identify a document that can be retrieved by the recipient of a message. For example it may be:

- ?? retrievable by using an HTTP "get"
- ?? contained in a MIME part that is a payload to the message.

There is an error if the recipient of a *message*:

1) does not recognize the ***PartyProfileReference*** in the ***To*** element as referencing a *Party Profile* that was created by the party that received the message, or

2) cannot retrieve the *Party Profile* referenced by the ***PartyProfileReference*** in the ***From*** element, or

3) having retrieved the *Party Profiles* referenced by the ***From*** and ***To*** elements, decides that they do not want to continue exchanging *messages*.

If an error occurs then the recipient of the *message* must report the error by sending an error message with a ***Severity*** of `Error` back to the sender of the message.

The following fragments contain examples of ***PartyProfileReferences***.

```
<PartyProfileReference>http://pp.example.com/POProfile1</PartyProfileReference>
<PartyProfileReference>cid:12309dsa012309----do93k</PartyProfileReference>
```

Editor's Note. ISSUE. It is possible that a Party Profile may contain more than one way by which messages can be exchanged by a party. For example, a Party Profile may allow sending of messages using either SMTP or HTTP. In this case, it may not be possible to determine what will actually be used from the two Party Profiles alone. Therefore some additional mechanism is required to specify exactly what will occur. I don't think we can force a Party Profile to contain only one set of options as then there may be far too many Profiles to cover all the possible permutations.

7.9.1.5 CPA References and Party Profiles – Valid Combinations

A ***From*** or a ***To*** element MAY contain:

- a ***CpaReference***,
- a ***PartyProfileReference***, or
- neither

There is an error if:

- a ***From*** element contains a ***CpaReference*** and the ***To*** element does not, and vice versa, or
- a ***From*** element contains a ***PartyProfileReference*** and the ***To*** element does not, and vice versa, or
- a ***From*** element contains neither a ***CpaReference*** nor a ***PartyProfileReference*** and the ***To*** element does contain one or the other, and vice versa.

If neither a ***CpaReference*** nor a ***PartyProfileReference*** is present then the recipient of a message MUST be able to determine the rules to be used when exchanging messages from other data contained in the *Message*, for example by using: the *From* ***PartyId***, the ***Service*** or the ***Action***. If this cannot be done then there is an Error.

If an error occurs then the recipient of the *message* must report the error by sending an error message with a ***Severity*** of `Error` back to the sender of the message.

7.9.1.6 From and To Party Examples

The following shows the overall structure of a ***From*** or ***To*** element.

```
  <From>...<From>
    <PartyId>...</PartyId>
    <ConversationId>...</ConversationId>
    ... one or none of ...
      <CpaReference>...</CpaReference>
    ... or ...
      <PartyProfileReference>...</PartyProfileReference>
  </From>
  <To>
    <PartyId>...</PartyId>
    <ConversationId>...</ConversationId>
                           ... on all but first message in a conversation...
    ... one or none of ...
      <CpaReference>...</CpaReference>
    ... or ...
      <PartyProfileReference>...</PartyProfileReference>
  </To>
```

The following is a more specific example. Note that there is no ***ConversationId*** in the ***To*** element indicating this is the first message in a conversation.

```
<From>
  <PartyId>MAILTO:joe@example.com</PartyId>
  <ConversationId>47923-12310-23423-12312</ConversationId>
  <CpaReference>as93je0p9-asio32jsd-r3osej-asd3</CpaReference>
</From>
<To>
  <PartyId>https://bigcompany.com/pomgmt</PartyId>
  <CpaReference>ahw23890asd-asdoi12#08ednl</CpaReference>
</To>
```

A *Message* that was returned as a result of receiving the *message* above could have the following as ***From*** and ***To*** element. Note that the ***ConversationId*** and the ***CpaReference*** that were present in the ***From*** element are now present in the ***To*** element.

```
<From>
  <PartyId>https://bigcompany.com/pomgmt</PartyId>
  <ConversationId>2asd0913uj-12309diw3jm</ConversationId>
  <CpaReference>ahw23890asd-asdoi12#08ednl</CpaReference>
</From>
<To>
  <PartyId>MAILTO:joe@example.com</PartyId>
  <ConversationId>47923-12310-23423-12312</ConversationId>
  <CpaReference>as93je0p9-asio32jsd-r3osej-asd3</CpaReference>
</To>
```

7.9.2 Service

The ***Service*** element identifies the Service that SHOULD act on the payload in the message. It SHOULD be unique within the domain of the ***Party*** to which the *message* is being sent. A URN MAY be considered suitable for the element content.

An example of a ***Service*** element follows.

```
<Service>urn:pip3a4<Service>
```

7.9.3 Action

The ***Action*** element identifies a process within a ***Service***, which processes the payload in the *Message*. ***Action*** MUST be unique within the ***Service*** in which it is defined.

An example of an ***Action*** element follows.

```
<Action>NewPurchaseOrder<Action>
```

7.9.4 MessageData

Editor's Note. This section has not changed

7.9.5 ReliableMessagingInfo

Editor's Note. This section has not changed.

7.9.6 XML Header sample

The following fragment demonstrates the structure of the ***Header*** element of the ***ebXMLHeader*** document:

```
<Header>
  <From>...</From>
  <To>...</To>
  <Service>...</Service>
  <Action>...</Action>
  <MessageData>...</MessageData>
  <ReliableMessagingInfo>...</ReliableMessagingInfo>
</Header>
```

7.10 DTDs

Editor's Note. The following is the revised DTD for the XML Header element. Note that MessageData and ReliableMessagingInfo have not been specified as they have not changed.

```
<!ELEMENT Header  (From, To, Service, Action, MessageData ,
                                                ReliableMessagingInfo )>
<!ELEMENT From (PartyId, ConversationId, (CpaRef | PartyProfile)? )>
<!ELEMENT To (PartyId, ConversationId?, (CpaRef | PartyProfile)? )>

<!ELEMENT PartyId (#PCDATA )>
<!ATTLIST PartyId
    type CDATA ('UserDefined') #IMPLIED
    e-dtype NMTOKEN #FIXED 'uri' >

<!ELEMENT ConversationId (#PCDATA) >
<!ATTLIST ConversationId
    e-dtype NMTOKEN #FIXED 'string' >

<!ELEMENT CpaRef (#PCDATA) >
<!ATTLIST CpaRef
    e-dtype NMTOKEN #FIXED 'string' >

<!ELEMENT PartyProfile (#PCDATA) >
<!ATTLIST PartyProfile
    e-dtype NMTOKEN #FIXED 'href' >
```

Editor's Note. The following is the alternative version of the revised DTD for the XML Header element. It contains the same data in a different structure. I don't like this as much – what do others think.

```
<!ELEMENT Header  (From, To, Service, Action, ConversationIds,
                                                PartyParameters?,
                                                MessageData,
                                                ReliableMessagingInfo )>

<!ELEMENT From (PartyId )>
<!ELEMENT To (PartyId )>

<!ELEMENT PartyId (#PCDATA )>
<!ATTLIST PartyId
    type CDATA ('UserDefined') #IMPLIED
    e-dtype NMTOKEN #FIXED 'uri' >

<!ELEMENT ConversationIds (FromConversationId, ToConversationId?)
<!ELEMENT FromConversationId (ConversationId) >
<!ELEMENT ToConversationId (ConversationId) >
<!ELEMENT ConversationId (#PCDATA) >
<!ATTLIST ConversationId
    e-dtype NMTOKEN #FIXED 'string' >

<!ELEMENT PartyParameters (CpaReferences | PartyProfiles )>
<!ELEMENT CpaReferencess (FromCpaReference, ToCpaReference) >
<!ELEMENT FromCpaReference (CpaReference) >
```

```
<!ELEMENT ToCpaReference (CpaReference) >
<!ELEMENT CpaReference (#PCDATA) >
<!ATTLIST CpaReference
    e-dtype NMTOKEN #FIXED 'string' >

<!ELEMENT PartyProfiles (FromPartyProfile, ToPartyProfile) ) >
<!ELEMENT FromPartyProfile (PartyProfile) >
<!ELEMENT ToPartyProfile (PartyProfile) >
<!ELEMENT PartyProfile (#PCDATA) >
<!ATTLIST PartyProfile
    e-dtype NMTOKEN #FIXED 'href' >
```