



POLARIS U.K. LIMITED

An Approach To XML Message Routing & Packaging Standards

Author: C Sparkes

Version: 1. 0

Date: October 2001

D R A F T

(First Published: 19/10/01)

Document Change Control

Reference:	DO#9457
Version:	1.00
Status:	DRAFT FOR REVIEW
Author(s):	C Sparkes (Polaris)

Change History

Version	Release Date	Purpose
1.00	October 2001	First Issue – Draft for external review.

References

Ref	Title	Version
[EBTRP]	Message Service Specification ebXML Transport, Routing & Packaging http://www.ebxml.org/specs/ebMS.pdf	1.0
[PXRIG]	XML-RTE Integrators Guide http://www.polaris-uk.co.uk/	3.0
[RFC2045, 2046 and related]	IETF RFC 2045 Multipurpose Internet Mail Extensions (MIME) and various related RFCs.	N/A
[SOAPSPEC]	Simple Object Access Protocol http://www.w3.org/TR/SOAP	1.1
[SOAPATTACH]	SOAP Messages with Attachments http://www.w3.org/TR/SOAP-attachments	11.12.00

Contents

1	INTRODUCTION.....	5
1.1	REQUIREMENTS.....	5
1.2	EBXML USE	6
1.3	LOGICAL MESSAGE STRUCTURE	6
2	MESSAGE STRUCTURES	8
2.1	GENERAL	8
2.1.1	Message Package	8
2.1.2	MessageHeader Element.....	8
2.2	REQUEST QUOTE.....	8
2.2.1	ebXML SOAP Message.....	9
2.2.2	Schemes Payload.....	9
2.2.3	PolData payload	9
2.3	QUOTE	10
2.3.1	ebXML SOAP Message.....	10
2.3.2	SchemeResult.....	11
2.4	DYNAMIC PRICING REQUEST	11
2.4.1	ebXML SOAP Message.....	12
2.4.2	DP Manifest	12
2.5	DYNAMIC PRICING QUOTE.....	13
2.5.1	DPErrror	13
2.6	OTHER MESSAGES.....	14
	APPENDIX A - DYNAMIC PRICING.....	15
	APPENDIX B – MESSAGE EXAMPLES.....	17
	B1 - REQUEST QUOTE.....	17
	B2 - DYNAMIC PRICING REQUEST	18
	B3 - DYNAMIC PRICING QUOTE.....	19
	APPENDIX C – POLARIS XML RTE IMPLICATIONS.....	22

1 Introduction

This document is not a normative reference for Polaris XML messaging standards. It is a draft of the proposed approach to meeting the community's messaging requirements and has been provided to enable interested parties to participate in the process of developing the eventual standards.

This document describes the proposed message wrapper structure for domestic UK general insurance messages based upon XML. It is the intention that this structure be utilised primarily for remote message exchanges between trading partners.

It assumes the adoption of the ebXML Transport Routing and Packaging specification [EBTRP] which itself is a utilisation of existing prevalent protocols such as SOAP (Simple Object Access Protocol) [SOAPSPEC]. Although the ebXML Transport Routing and Packaging specification appears to provide a sound basis for our messaging standards, it is not clear at this time whether a complete implementation of the ebXML architecture is readily achievable by our members and has therefore not been assumed. Polaris will continue to review developments in this area and advise our members accordingly.

1.1 Requirements

The requirements of any structure either developed by or as in this case adopted by Polaris on behalf of the industry were initially identified as follows:

1. The wrapper must be based upon non-proprietary technology standards.
2. The wrapper must be light-weight. That is to say, that there should be no requirement upon the message user to implement any particular system architecture or infrastructure.
3. The wrapper must be generic i.e. it should be possible to use the same basic wrapper for all business and technical exchanges - not just business processes like "request a quote" or "claim enquiry", but technical functions such as "query service status".
4. The wrapper must be independent of the transmission 'protocol' e.g. HTTP, FTP, SMTP.
5. The wrapper must support notionally synchronous and asynchronous exchanges. Some exchanges for example a "request quote" may receive a synchronous "quote" response, a "quote referral" might get a response over night and others such as "request policy documents" might have no electronic business response at all.
6. The wrapper must allow for the inclusion in the message of multiple data formats, e.g. text, binary, image etc. This will be particularly important if the messages are used for larger risks, for claims processing or to carry encrypted payload components see 9.
7. There must be a generic way of identifying message level errors, e.g. "service unavailable", "message not understood", "payload invalid".
8. The wrapper must provide elements to carry control, delivery and receipt information so that message users can implement audit and non-repudiation mechanisms.
9. The wrapper must support digital signatures and encryption. Whilst it is expected that particular implementations will utilize session level security, it is quite conceivable that

certain parts of the payload will need to be separately signed and/or encrypted for example if they originate from different sources.

10. The wrapper must identify the purpose of the payload so that it is not necessary for it to be understood/parsed prior to its delivery to the appropriate remote procedure.
11. The message structure as a whole (perhaps at a wrapper level) will need to support the aggregation of information from more than one service into a single message. For example an internet host could direct a "request for quote" to various different web sites and then combine the output from each of the respondents into a single message to the client. It is not clear whether this is something to be treated as a multiple payload scenario or whether the aggregated data will be manifested as a single new payload. Another example where this feature may be required would be to accommodate supplementary call outs to services such as CUE. The business requirements in this area are not clear, but the options need to be considered.

Adoption of the ebXML proposals for Transport, Routing and Packaging will provide for these requirements.

1.2 ebXML Use

The remainder of this document describes how Polaris messages could utilize ebXML. Readers would benefit from an understanding of the principles of SOAP 1.1 [SOAPSPEC], MIME [RFC2045 and related], the method of using SOAP with MIME [SOAPATTACH] and will have to understand the ebXML Transport Routing and Packaging specification [EBTRP].

Note that only the proposed Polaris use of the ebXML structure is described herein. Unless specifically instructed by this document, implementers **MUST** follow [EBTRP]. Some areas of functionality are not mandated by ebXML and although implementers of ebXML Message Service Handlers (MSHs) should support such optional features, their use in Polaris messages may not be required.

1.3 Logical Message Structure

The approach taken by the proposal is to utilize separate MIME parts within the messages for logically separate data sets. The same essential utilization of the ebXML structure has been adopted for all messages without the need for extension to the ebXML elements.

The standard ebXML Manifest points to a MIME part as though it were a parameter passed to the Service Action.

The decision as to what constitutes a separate MIME part has been based upon an expectation of what data needs to be exposed to what process. One of the key aspects of the message structure was to be able to support the aggregation of requests via a central processor such as an internet portal. This has added a portal specific MIME part to enable the portal to construct individual insurer messages to forward on to insurer located services (See Appendix A Dynamic Pricing). The utilization of separate MIME parts means that the need for an internet portal to understand the risk data payload has been removed.

Further, the approach adopted will permit separate signing of payloads and the end-to-end encryption of individual MIME parts (using S/MIME). For multi-recipient messages where end-to-end encryption is required, it is expected that users will wish to asymmetrically encrypt the symmetric key for each recipient. Polaris has not looked at this area in detail, but such an

approach removes the need to expose unencrypted data for the purpose routing etc., and is (we believe) commonly understood.

Appropriate Polaris payload structures have been identified for the business processes discussed to date. Risk Data structures have been based upon (but also modify see Appendix C) the structures described in the Polaris XML Integrators Guide [PXRIG]. Once agreement has been reached on the message structures, it is expected that [PXRIG] will be updated accordingly.

Note that the messages described herein are general purpose and do not mandate the use of Polaris software.

2 Message Structures

This section describes the messages identified to date. The wrapper structure is intended to work for all messages and this information is included in the first instance in lieu of a formal approach to the publication of message payload constructs. URIs for schemas etc., although proposed within the structure **do not** yet map to physical locations and will only be implemented once broad agreement has been reached as to the applicability of the structures.

2.1 General

The following general rules will apply to all messages:

2.1.1 Message Package

The message package will be implemented as per [EBTRP]. The message **MUST** be encoded in UTF-8 and the charset attribute in the MIME content-type header **MUST** be set to “UTF-8”.

2.1.2 MessageHeader Element

The PartyId element within both the From and To elements **MUST** have one instance with a valid value from InStep code list 1. The PartyId type attribute **MUST** be set to <http://insteponline.co.uk/stds/codes/1>. Other PartyId elements **MAY** be present within the From and To elements, but do not need to be understood by Polaris message applications.

The CPAId element **MUST** be set to “<http://polaris-uk.co.uk/ebxmlcpa>”.

The Service element **MUST** be set to the URN of a service identified at “<http://polaris-uk.co.uk/ebxmlcpa>” and the Action element **MUST** be set to a valid action for that Service.

(It has currently been assumed that Polaris will develop an industry Collaboration Protocol Agreement. More work is required to understand the ebXML proposals in this area, but in any case we expect some form of CPA to be published centrally and that it will include definition of services and actions).

The MessageHeader Service element Type attribute will identify the line of insurance business and **MUST** be set to a valid value from Instep code list 7 at <http://insteponline.co.uk/stds/codes/7>.

2.2 Request Quote

The Request Quote Message is used to invoke the risk assessment and quotation processing provided by a single quotation service. It contains the following MIME parts:

- a REQUIRED ebXML SOAP Message,
- a REQUIRED Schemes payload,
- a REQUIRED PolData payload,
- any number of non-mandated supporting documents.

Note that supporting documents could be anything from a plan/report to a structured message such as the result of a CUE enquiry.

The receipt of valid but **non-understood** payload components should lead to one of two outcomes. Either the application should reject the Request Quote and respond with an error or it

should forward the message (and its attachments) for human intervention and if necessary, downgrade any business response to asynchronous whilst acknowledging message receipt synchronously. The decision as to whether to reject or refer will depend upon the MIME type of the non-understood payload and the insurers ability/desire to service it. For example, it may be common in some lines of business to submit supporting image documents that would ordinarily need to be reviewed by an underwriter.

2.2.1 ebXML SOAP Message

The following components will be set as described below:

1. The MessageHeader From element will identify the party requesting the quote. This will be the immediate message sender. In the case where a broker/intermediary sends a message direct to an insurer service, this will identify the broker/intermediary. In the scenario where the message is sent via a portal or some other intermediate service provider, this will identify that service. Where the identification of the true trading partner is required for business purposes, this will be handled as part of the business payload(s).
2. The MessageHeader To element will identify the insurer from whom the quote is requested.
3. If spawned as part of a Dynamic Pricing Request, the MessageHeader ConversationId element will be set to that of the Dynamic Pricing Request message.
4. The MessageHeader Service element will be set to “urn:polaris-uk.co.uk/ebxmlcpa/newbusiness”.
5. The MessageHeader Action element will be set to “requestquote”.

2.2.2 Schemes Payload

The Schemes payload will conform to the XML fragment for the Schemes element in [PXRIG].

Manifest elements referencing this MIME part MUST have:

1. their Reference xlink:role attribute set to “http://polaris-uk.co.uk/xmlstds/schemes/role.htm”
2. their Schema element Location attribute set to “http://polaris-uk.co.uk/xmlstds/schemes/v1.xsd” and
3. their Schema element Version attribute set to “1.0”.

2.2.3 PolData payload

The PolData payload will conform to the PolData XML fragment described in [PXRIG].

More particularly:

A REQUIRED PolData element with its Type attribute set to “Input” will describe the risk for which the quotation is required and will conform to a Polaris dictionary or published data capture subset of a dictionary.

An OPTIONAL PolData element with its Type attribute set to “Request” may be included and permits the specification of the data returned in the Quotation. Insurers are NOT obliged to support this feature.

The dictionary data will be present in the “Object” and “Property” tags.

None of the other PolData element's attributes or attribute values are supported in this scenario.

Manifest elements referencing this MIME part MUST have:

1. their Reference xlink:role attribute set to "http://polaris-uk.co.uk/xmlstds/podata/role.htm"
2. their Schema element Location attribute set to "http://polaris-uk.co.uk/xmlstds/podata/" + SchemaName + "/" + LongDictionaryVersion + ".xsd" and
3. their Schema element Version attribute set to DictionaryVersion.

SchemaName MUST be either a valid DictionaryName i.e. "personalmotor", "household", "commercial" or some published Polaris data capture subset of the same in the format dictionary + "/" + SubsetName e.g. "commercial/shop".

LongDictionaryVersion will be a 4 character string derived from DictionaryVersion, three characters right aligned and zero padded for the integer part plus one character for the decimal part. So version "42.1" will become "0421".

DictionaryVersion is the version number of the Polaris dictionary upon which the schema is based, e.g. "42.1".

2.3 Quote

The Quote Message is used to provide a quotation for the related Request Quote message. It contains the following MIME parts.

a REQUIRED ebXML SOAP Message,
a CONDITIONAL SchemeResult payload for each scheme identified in the Schemes payload of the Request Quote, and
any number of non-mandated supporting documents.

2.3.1 ebXML SOAP Message

The following components will be set as described below:

1. The MessageHeader From element will identify the insurer providing the quote.
2. The MessageHeader To element will identify the immediate recipient of the quote

i.e. the From and To elements will be the reverse of those on the Request Quote.
3. The MessageHeader ConversationId element will be set to that of the Request Quote message.
4. The MessageHeader Service element will be set to "urn:polaris-uk.co.uk/ebxmlcpa/newbusiness".
5. The MessageHeader Action element will be set to "quote".
6. The MessageHeader MessageData RefToMessageId will be set to the MessageHeader MessageData MessageId of the Request Quote message.

2.3.2 SchemeResult

The SchemeResult payload will conform to and extend the XML fragment for the SchemeResult element in [PXRIG].

More particularly:

Beneath the REQUIRED SchemeResult element will be either:

A PolData element with its Type attribute set to “Output” which will provide details of the Quote, or

One or more ErrorDetail elements plus a PolData element with its Type attribute set to “Error” and which contains the risk data at the time of the error.

An OPTIONAL repeating element of SupplementRef will extend the structure, will take the same form as the Manifest Reference element and will bind any supplementary payloads items to the particular SchemeResult.

Manifest elements referencing this MIME part MUST have:

1. their Reference xlink:role attribute set to “http://polaris-uk.co.uk/xmlstds/schemeresult/role.htm”
2. their Schema element Location attribute set to “http://polaris-uk.co.uk/xmlstds/schemeresult/”+ SchemaName + “/” LongDictionaryVersion + “.xsd” and
3. their Schema element Version attribute set to DictionaryVersion.

SchemaName, LongDictionaryVersion, DictionaryVersion as defined in 2.3.1.

2.4 Dynamic Pricing Request

The Request Dynamic Pricing message is used to invoke the risk assessment and quotation processing provided by multiple quotation services aggregated at a third party portal. It contains the following MIME parts:

a REQUIRED ebXML SOAP Message,
a REQUIRED DP Manifest payload,
a REQUIRED PolData payload,
any number of non-mandated supporting documents (see 2.2),

and for discussion:

any number of OPTIONAL Portal Ancillary Service Supplements (hereinafter called PASS) payloads.

(Discussions have been held as to the value of the portal providing certain validation services such as CUE or Credit Checking before passing the data on to the insurers for quoting – avoiding lots of insurers having to do the same checks. The business requirements have not been established at this point and indeed there are conflicting requirements in this area. For example, these service providers will not wish to receive PolData type structures, but will have messages formats of their own. How does the portal find the appropriate data from within

PolData without understanding the payloads? Further, if the portal stores or manipulates non-encrypted data “plaintext” this creates a significant security risk. One way of enabling these sorts of ancillary functions would be to have the party that requests the Dynamic Pricing format up the PASS payloads and identify which of the insurers is looking to receive them. The information they need to know to achieve this could be distributed along with insurer products and Dynamic Pricing Filters.)

2.4.1 ebXML SOAP Message

The following components will be set as described below:

1. The MessageHeader From element will identify the party requesting the quote. This will be the broker/intermediary.
2. The MessageHeader To element will identify the portal.
3. The MessageHeader Service element will be set to “urn:polaris-uk.co.uk/ebxmlcpa/dynamicpricing”.
4. The MessageHeader Action element will be set to “requestquote”.

2.4.2 DP Manifest

This MIME part will be aimed at a portal providing Dynamic Pricing. Its purpose is to provide the portal with enough information to take the Dynamic Pricing Request message, slice it up into its separate MIME parts and then reassemble them into individual insurer Request Quote messages to be sent on accordingly.

This MIME part will contain an XML fragment with REQUIRED DPManifest element containing one or more InsurerService elements.

Each InsurerService element will contain:

a REQUIRED InsurerId element,
one or more Schemes elements,

and for discussion:

any number of PASS elements.

The InsurerId element will take the same form as the ebXML To element in the ebXML MessageHeader element. Note that although multiple forms of identity are permitted in any one message, each instance of InsurerService will identify one and only one insurer.

The Schemes element will conform to the XML fragment for the Schemes element in [PXRIG]. The portal can use this information to format up insurer specific Schemes payloads (see 2.2.2).

The PASS element will (if adopted) take the same form as the ebXML Manifest element. The implication being that the portal would understand the xlink:role , pick up the MIME part, invoke the ancillary service and then pass the returned MIME part(s) on to the insurer as supplementary payloads for them to process.

The MIME parts could be signed and encrypted as required and the portal could avoid any knowledge of the payloads.

Manifest elements referencing this MIME part MUST have:

1. their Reference xlink:role attribute set to “http://polaris-uk.co.uk/xmlstds/dpmanifest/role.htm”
2. their Schema element Location attribute set to “http://polaris-uk.co.uk/xmlstds/dpmanifest/v1.xsd” and
3. their Schema element Version attribute set to “1.0”.

2.5 Dynamic Pricing Quote

The Dynamic Pricing Quote will essentially conform to and extend the structure used by the Quote message.

The following components will be set as described below:

1. The MessageHeader From element will identify the portal.
2. The MessageHeader To element will identify the broker/intermediary

i.e. the From and To elements will be the reverse of those on the Request Dynamic Pricing.

3. The MessageHeader ConversationId element will be set to that of the Request Dynamic Pricing message.
4. The MessageHeader Service element will be set to “urn:polaris-uk.co.uk/ebxmlcpa/dynamicpricing”.
5. The MessageHeader Action element will be set to “quote”.
6. The MessageHeader MessageData RefToMessageId will be set to the MessageHeader MessageData MessageId of the Request Dynamic Pricing message.

An additional MIME part may be present and reports any errors with insurer services.

2.5.1 DPError

This MIME part will contain an XML fragment with a root element of DPError containing one or more InsurerService elements.

Each InsurerService element will contain:

- a REQUIRED InsurerId element
- a REQUIRED ErrorList element.

The ErrorList element will conform to the ebXML ErrorList element. The portal will provide this information for each service that returned an ebXML ErrorList or with which it has experienced problems.

Manifest elements referencing this MIME part MUST have:

1. their Reference xlink:role attribute set to “http://polaris-uk.co.uk/xmlstds/dperror/role.htm”
2. their Schema element Location attribute set to “http://polaris-uk.co.uk/xmlstds/dperror/v1.xsd” and
3. their Schema element Version attribute set to “1.0”.

Note that this information is provided over and above that given as part of the Dynamic Pricing Quotes own ErrorList element.

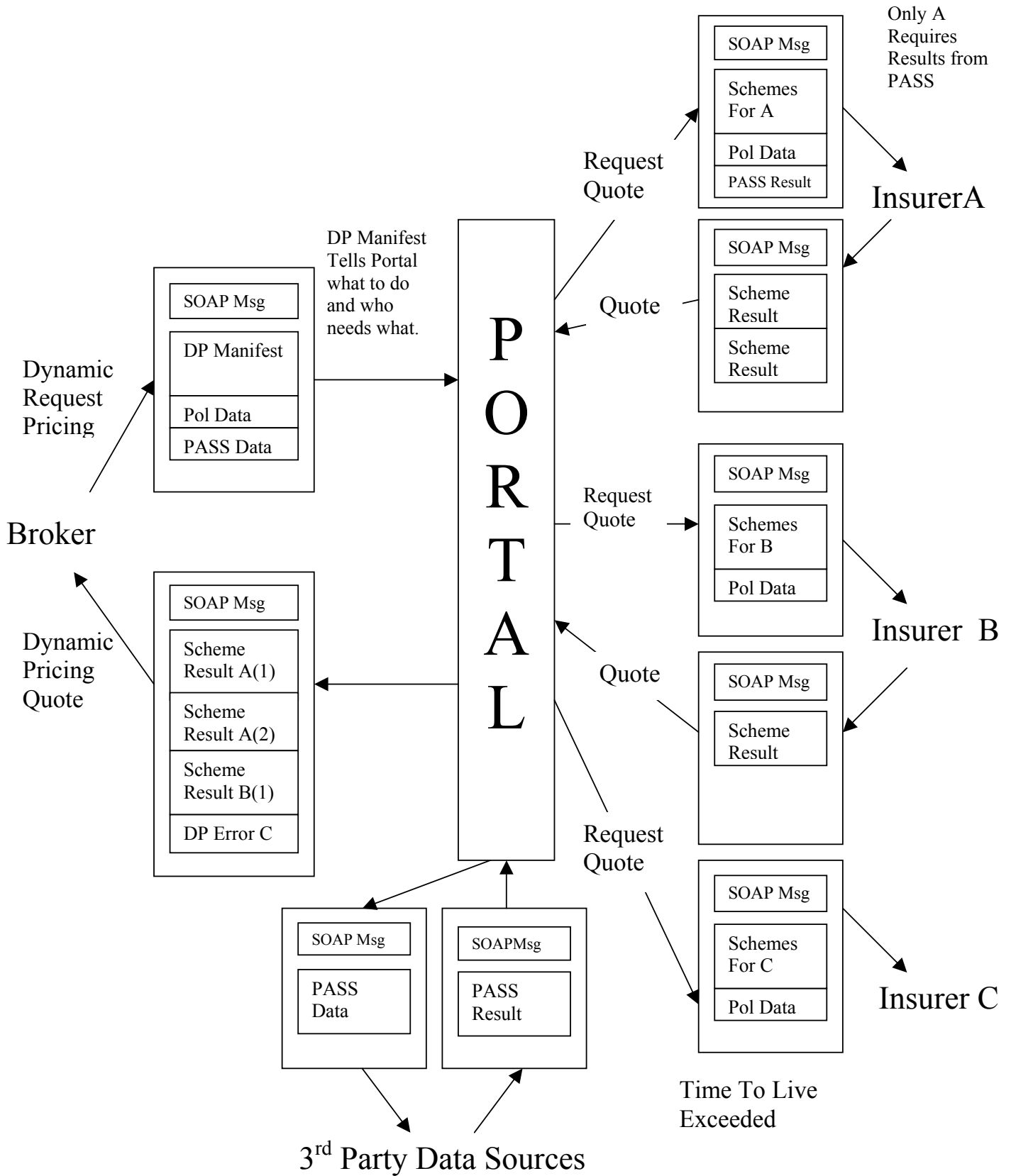
2.6 Other Messages

The following messages have been discussed, but are not yet specified:

- POS Placement
- POS Adjustment
- POS Cancellation
- POS Renewal
- POS Lapse
- Renewal Invitation
- Request Claims Detail
- Claims Detail
- Claim Notification
- Claim Update
- Query CUE
- CUE Detail
- Message Status Request
- Message Status Response
- Service Status Request
- Service Status Response
- Dynamic Pricing Statistics

Appendix A - Dynamic Pricing

The following diagram shows an example Dynamic Pricing conversation:



The broker application formats up the Dynamic Pricing Request. The SOAP Message payload identifies the message type and refers to the MIME payloads. Only the DP Manifest is aimed at the portal.

The DP Manifest contains information about the schemes to be run and identifies any Portal Ancillary Service Supplement (PASS) requirements. Using this information, the portal can format up the individual Request Quote messages bound for the Insurer services.

The Request Quote message received by any one insurer will make only reference to their own schemes. The PolData payload will be the same for each insurer and will be passed through unchanged.

In the diagram, only Insurer A is interested in the results of a PASS. This could be a CUE lookup, a HALCRO lookup, a credit check etc. This is only one approach to achieving this end. Alternatively, the portal or some other service utilised by either the portal, sender or recipient could manipulate the PolData payload before passing on the appropriate PASS Request data to the PASS provider. Various options exist, but at present, each approach to solving this type of requirement presents its own issues and further consideration is required.

Each insurer service will return a Quote message to the portal. Where more than one product is quoted for a particular insurer, the results will be returned in separate Scheme Result payloads. This is a divergence from the XML structure defined in [PXRIG] where multiple responses are held in subsequent instances of the SchemeResult element. This approach is suggested as it will afford easier creation and interpretation of the Dynamic Pricing Quote.

Finally the portal will combine the respective insurer Quotes into a Dynamic Pricing Quote before returning it to the Broker.

In the example given, Insurer C was unable to respond within the Time To Live period and so the portal recorded an error for this service.

In the event that none of the services respond in time, the portal will issue a SOAP Message level error in the ebXML ErrorList element.

Appendix B – Message Examples

The following examples are included for illustration only – they have not been tested. Please also note that they have been formatted to make them easier to read and would need adaptation before they could be used.

B1 - Request Quote

The following example demonstrates the Request Quote message. This would be typical of a message aimed at a particular insurer's service. In addition to the ebXML SOAP message, the Schemes payload identifies two schemes for which quotes are required and the Two MIME payloads exist

```
Content-Type: multipart/related; type=text/xml;
boundary=PolarisebXMLMsgBoundary; start=PolarisMsgPackageExample@Polaris-uk.co.uk

--PolarisebXMLMsgBoundary
Content-ID: PolarisMsgPackageExample1Polaris-uk.co.uk
Content-Type: text/xml; charset="UTF-8";

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:schemaLocation="http://schema.xmlsoap.org/soap/envelope/
http://ebxml.org/project_teams/transport/envelope.xsd">
  <SOAP-ENV:Header xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
    xsi:schemaLocation="http://www.ebxml.org/namespaces/messageHeader"
    "http://ebxml.org/project_teams/transport/messageHeaderv0_99.xsd">
    <eb:MessageHeader id="HdrEx" eb:version="1.0" SOAP-
ENV:mustUnderstand="1">
      <eb:From>
        <eb:PartyId
          eb:type="http://insteponline.co.uk/stds/codes/1">
          1234
        </eb:PartyId>
      </eb:From>
      <eb:To>
        <eb:PartyId
          eb:type="http://insteponline.co.uk/stds/codes/1">
          5678
        </eb:PartyId>
      </eb:To>
      <eb:CPAid>http://polaris-uk.co.uk/ebxmlcpa</eb:CPAid>
      <eb:ConversationId>2001-08-17-1530-0001</eb:ConversationId>
      <eb:Service>urn:polaris-uk.co.uk/ebxmlcpa/newbusiness</eb:Service>
      <eb:Action>requestquote</eb:Action>
      <eb:MessageData>
        <eb:MessageId>2001-06-26-1530-845999@
1234.co.uk</eb:MessageId>
        <eb:Timestamp>2001-06-26T15:30:34Z</eb:Timestamp>
      </eb:MessageData>
    </eb:MessageHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
    xsi:schemaLocation="http://www.ebxml.org/namespaces/messageHeader"
    http://ebxml.org/project_teams/transport/messageHeaderv0_99.xsd">
    <eb:Manifest eb:id="Manifest" eb:version="1.0">
      <eb:Reference eb:id="RefToSchemes"
        xlink:href="cid:polaris-uk.co.uk.schemes"
        xlink:role="http://polaris-
uk.co.uk/xmlstds/schemes/role.htm" >
      <eb:Schema eb:location="http://polaris-
uk.co.uk/xmlstds/schemes/V1.xsd" eb:version="1"/>
    </eb:Manifest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```

        </eb:Reference>
        <eb:Reference eb:id="RefToPolData"
            xlink:href="cid:polaris-uk.co.uk.poldata"
            xlink:role="http://polaris-
            uk.co.uk/xmlstds/poldata/role.htm" >
            <eb:Schema eb:location="http://polaris-
            uk.co.uk/xmlstds/poldata/personalmotor/0421.xsd"
            eb:version="42.1" />
        </eb:Reference>
    </eb:Manifest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--PolarisebXMLMsgBoundary
Content-ID: polaris-uk.co.uk.schemes
Content-Type: application/xml
<Schemes xmlns:poluk="http://polaris-uk.co.uk/xmlstds">
    <Scheme Ref="Insurer A Super Cover">
    <Scheme Ref="Insurer A Super Cover Plus">
</Schemes>

--PolarisebXMLMsgBoundary
Content-ID: polaris-uk.co.uk.poldata
Content-Type: application/xml
<PolMsg xmlns:poluk="http://polaris-uk.co.uk/xmlstds">
    <PolData Type="Input">
        Risk data goes here...

        <Policy>
            etc...

        </Policy>
    </PolData>
</PolMsg>

```

B2 - Dynamic Pricing Request

Apart from differences in the header and manifest detail, the main difference between a Request Quote and a Dynamic Pricing Request transaction, is the inclusion of the DP Manifest Payload Component in place of the Schemes payload. An example is given below.

```

--PolarisebXMLMsgBoundary
Content-ID: polaris-uk.co.uk.dpmanifest
Content-Type: application/xml
<DPManifest xmlns:poluk="http://polaris-uk.co.uk/xmlstds"
    xmlns:eb="http://www.ebxml.org/namespaces/messageHeader">
    <InsurerService>
        <InsurerId>
            <eb:To>
                <eb:PartyId
                    eb:type="http://insteponline.co.uk/stds/codes/1">
                    5678
                </eb:PartyId>
            </eb:To>
        </InsurerId>
        <Schemes>
            <Scheme Ref="Insurer A Super Cover">
            <Scheme Ref="Insurer A Super Cover Plus">
        </Schemes>
    </InsurerService>
    <InsurerService>
        <InsurerId>
            <eb:To>
                <eb:PartyId
                    eb:type="http://insteponline.co.uk/stds/codes/1">
                    91011
                </eb:PartyId>
            </eb:To>
        </InsurerId>
    </InsurerService>

```

```

        </eb:PartyId>
    </eb:To>
</InsuerId>
<Schemes xmlns:poluk="http://polaris-uk.co.uk/xmlstds"
    <Scheme Ref="Insurer B Protection Plus">
</Schemes>
<PASS>
    <eb:Reference eb:id="RefToPASSPayload"
        xlink:href="cid:pretendpayload"
        xlink:role="http://somerole/role.htm" >
    <eb:Schema
        eb:location="http://someschemalocation/v1.xsd"
        eb:version="1"/>
    </eb:Reference>
</PASS>
<InsuerService>
</DPManifest>

```

In this example the insurer service for insurer B (91011) is also expecting to receive the results of a callout to the somerole PASS.

B3 - Dynamic Pricing Quote

The following example shows a dynamically priced quotation.

```

Content-Type: multipart/related; type=text/xml;
boundary=PolarisebXMLMsgBoundary; start=PolarisMsgPackageExample@Polaris-uk.co.uk

--PolarisebXMLMsgBoundary
Content-ID: PolarisMsgPackageExample1Polaris-uk.co.uk
Content-Type: text/xml; charset="UTF-8";

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xsi:schemaLocation="http://schema.xmlsoap.org/soap/envelope/
http://ebxml.org/project_teams/transport/envelope.xsd">

<SOAP-ENV:Header xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
    xsi:schemaLocation="http://www.ebxml.org/namespaces/messageHeader"
    "http://ebxml.org/project_teams/transport/messageHeaderV0_99.xsd">
<eb:MessageHeader id="HdrEx" eb:version="1.0" SOAP-
ENV:mustUnderstand="1">
    <eb:From>
        <eb:PartyId
            eb:type="http://insteponline.co.uk/stds/codes/1">
            1011
        </eb:PartyId>
    </eb:From>
    <eb:To>
        <eb:PartyId
            eb:type="http://insteponline.co.uk/stds/codes/1">
            1234
        </eb:PartyId>
    </eb:To>
    <eb:CPAid>http://polaris-uk.co.uk/ebxmlcpa<eb:CPAid>
    <eb:ConversationId>2001-08-17-1530-0001</eb:ConversationId>
    <eb:Service>urn:polaris-
uk.co.uk/ebxmlcpa/dynamicpricing</eb:Service>
    <eb:Action>quote</eb:Action>
    <eb:MessageData>
        <eb:MessageId>2001-08-17-1531-845999@
            1011.co.uk</eb:MessageId>
        <eb:RefToMessageId>2001-08-17-1530-115969@
            1234.co.uk</eb:RefToMessageId>
    </eb:MessageData>
    </eb:MessageHeader>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

        <eb:Timestamp>2001-08-17T15:31:34Z</eb:Timestamp>
    </eb:MessageData>
</eb:MessageHeader>
</SOAP-ENV:Header>

<SOAP-ENV:Body xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
  xsi:schemaLocation="http://www.ebxml.org/namespaces/messageHeader"
  http://ebxml.org/project_teams/transport/messageHeaderv0_99.xsd">
  <eb:Manifest eb:id="Manifest" eb:version="1.0">
    <eb:Reference eb:id="RefToSchemeResult1"
      xlink:href="cid:polaris-uk.co.uk.schemeresult.1"
      xlink:role="http://polaris-
        uk.co.uk/xmlstds/schemeresult/role.htm" >
      <eb:Schema eb:location="http://polaris-
        uk.co.uk/xmlstds/schemeresult/V1.xsd" eb:version="1"/>
    </eb:Reference>
    <eb:Reference eb:id="RefToSchemeResult2"
      xlink:href="cid:polaris-uk.co.uk.schemeresult.2"
      xlink:role="http://polaris-
        uk.co.uk/xmlstds/schemeresult/role.htm" >
      <eb:Schema eb:location="http://polaris-
        uk.co.uk/xmlstds/schemeresult/V1.xsd" eb:version="1"/>
    </eb:Reference>
    <eb:Reference eb:id="RefToSchemeResult3"
      xlink:href="cid:polaris-uk.co.uk.schemeresult.3"
      xlink:role="http://polaris-
        uk.co.uk/xmlstds/schemeresult/role.htm" >
      <eb:Schema eb:location="http://polaris-
        uk.co.uk/xmlstds/schemeresult/V1.xsd" eb:version="1"/>
    </eb:Reference>
  </eb:Manifest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

--PolarisebXMLMsgBoundary
Content-ID: polaris-uk.co.uk.schemeresult.1
Content-Type: application/xml
<SchemeResult xmlns:poluk="http://polaris-uk.co.uk/xmlstds"
  Ref ="Insurer A Super Cover">
  <PolData Type="Output">
    Risk data goes here...

    <CalculatedResult>
      etc...

    </CalculatedResult>
  </PolData>
</SchemeResult>

```

```

--PolarisebXMLMsgBoundary
Content-ID: polaris-uk.co.uk.schemeresult.2
Content-Type: application/xml
<SchemeResult xmlns:poluk="http://polaris-uk.co.uk/xmlstds"
  Ref ="Insurer A Super Cover Plus">
  <PolData Type="Output">
    Risk data goes here...

    <CalculatedResult>
      etc...

    </CalculatedResult>
  </PolData>
</SchemeResult>

```

```

--PolarisebXMLMsgBoundary
Content-ID: polaris-uk.co.uk.schemeresult.3
Content-Type: application/xml
<SchemeResult xmlns:poluk="http://polaris-uk.co.uk/xmlstds"
  Ref ="Insurer B Protection Plus">
  <PolData Type="Output">
    Risk data goes here...

```

```
<CalculatedResult>
    etc...

</CalculatedResult>
</PolData>
</SchemeResult>
```

Note that the example shows separate MIME parts for each scheme – even though quotes for some of the schemes may have originated at the same insurer service. This would appear to make sense as it provides a single structure of interface for all quotes. Also, there has been no attempt to mandate the structure of the href attributes in the manifest. The community could agree (for example) to have them identify the logical scheme name by including the Ref from the Schemes element in the Request. This is not done in the example and so it will be necessary to parse the Scheme Result MIME parts to know which part applies to what scheme. Further consideration is needed.

Appendix C – Polaris XML RTE Implications

The following section describes proposed (but not agreed) changes to the XML RTE interface to support the new message structures.

To maintain compatibility with existing implementations, it is expected that any changes made to the Polaris XML-RTE interface will be delivered as additional functionality.

The new message structures break the PolMessage element described in [PXRIG] down into several discrete MIME parts.

The Dictionary element will not be used as this information can be obtained from the ebXML Manifest element.

It is proposed that the Transaction element will not be supported as a remote feature and has not been included in the message definition. The ebXML Service and Action elements identify the transaction to be performed. The Transaction element also provided for the *ad hoc* definition of transaction steps which is not currently felt to be appropriate. The only other purpose of the element was to identify the Effective Date of the Transaction, which is still required and will need to be included in the message. The most logical place for this information to live would be as part of the Action element. It is however not clear as to whether extension in this way will be problematic and further consideration is necessary.

The Schemes element has been broken out into a separate MIME part to make the information more readily available for portal type applications.

TranResult has been dropped in favour of the ebXML ErrorList element although some more thought is required in this area.

SchemeResult has been made a separate MIME part.

PolData has been made a separate MIME part although it is still integral to the SchemeResult element.

Other areas of change are likely to include a more compact element naming convention for dictionary objects and properties.