FRC B2M-B2B Hub System Specifications for theWA Gas Retail Market

Prepared by:	AEMO Reform Delivery
Version:	4.0
Effective date:	30 June 2023
Status:	FINAL

Approved for distribution and use by:

Approved by:	Violette Mouchaileh
Title:	Executive General Manager – Reform Delivery
Date:	10/03/2023

Contents

Figures

Current version release details

Version	Effective date	Authors	Summary of changes
3.10	30/06/2023	D Freeman Nandu Datar	• IN001/23W - Section 5.2 on obtaining certificates pointing to online instruction • IN001/23W Section 5.4 updated from SSL to current TLS 1.2 version • IN005/23 – Transfer SA specific details

Note: There is a full version history at the end of this document.

1. Introduction

1.1. Purpose

This document provides specifications and settings that apply to the transaction and messaging systems to be used for both B2M and B2B communications in the Western Australian Gas Retail Market.

This document forms part of the AEMO Specification Pack and has been prepared by AEMO for the benefit of participants in the Western Australian Gas Market.

These specifications and settings will become superseded from time to time throughout the life of the FRC B2B System (FBS). The specifications comprise standard service names, time intervals for message timeouts, standard participant identifiers, XML header mappings, and specific FBS port and network addressing details.

These specifications are the variable implementation details that apply to the FBS architecture, as defined in *FRC B2M-B2B Hub System Architecture* document.

As both B2M and B2B transactions will use the same architecture, the term 'B2B'has been used to apply to both transactions between participants and also transactions to the Market System

1.2. Audience

The document has been written for business and IT personnel within industry participants in the Western Australian gas industry, as well as AEMO's business and IT personnel. It is expected that the audience will have a familiarity with the overall business endeavour of Gas FRC in the Western Australian Gas Retail Market and with the artefacts listed in the Related Documents section of this document.

1.3. Related documents

This document should be read in conjunction with the other documents contained within the AEMO Specification Pack as defined in the AEMO Specification Pack – Usage Guide.

1.3.1. Victoria

A number of documents were referred to in the Victorian version of this document. For further information about Victorian processes and specifications of the following related documents or artefacts that have been issued as part of Participant Build Packs 1, 2 and 3 and should be read conjunction with this document. The table below defines the documents referred to in the Victorian version of this document.

Ref	Artefact Name	Version	Responsible party or authors
1	Participant Build Pack 3 FRC B2B System Architecture	Current version as published in the GIP	AEMO
2	Participant Build Pack 3 Interface Definitions (familiarity only)	Current version as published in the GIP	AEMO
3	ebXML Message Service Specification	1.0	UN/CEFACT and OASIS
4	Participant Build Pack 1 (familiarity only)	Current version as published in the GIP	AEMO
5	Participant Build Pack 2 (familiarity only)	Current version as published in the GIP	AEMO
6	Guidelines for Development of A Standard for Energy Transactions in XML (aseXML)	Current version as published at http://www.aemo.com.au/aseXML/index.htm	ASWG
1	Participant Build Pack 3 FRC B2B System Architecture	Current version as published in the GIP	AEMO

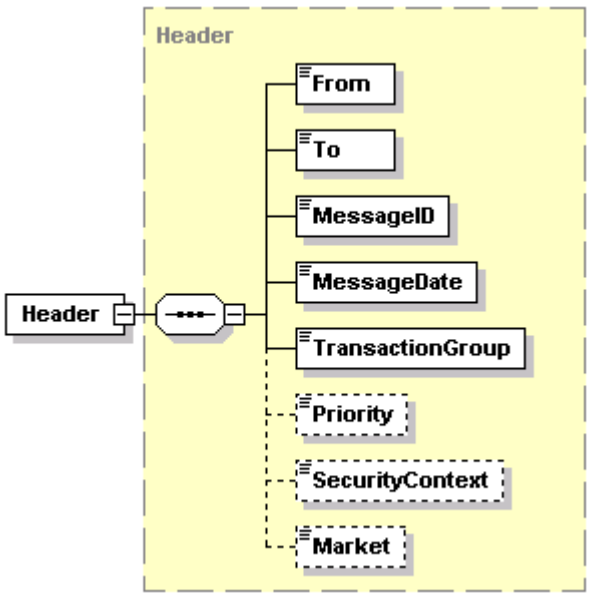
2. General information

2.1. aseXML to ebXML Overview

Chapters 3 and 4 in this document define specifically the relationships between the contents of the aseXML header and the ebXML message elements. This section gives a high level overview of these relationships; along with timing considerations in the FBS.

The structure of the aseXML Header is as follows:

Figure 1 aseXML Header Format



Five fields from the aseXML header will be used to derive ebXML message element information. The `From` and `To` fields in the ebXML header are derived according to the content of the aseXML `From` and `To` fields, in conjunction with the optional `SecurityContext` field. Message elements including `Service` names and associated message timing elements and parameters are derived from the optional `Priority` element. The `TransactionGroup` element is mapped directly to the `Action` element name.

The following table shows which ebXML elements are based on the contents of the aseXML Header elements.

aseXML Header elements	No mapping	ebXML element Direct mapping	ebXML element Derived mapping
From		From	
To		`To`	
MessageId	x		
MessageDate	x		
`TransactionGroup`		`Action`	
`Priority`			`Service`
`Priority`			`CPAId`
`Priority`			`TimeToLive`
`SecurityContext`	x		
`Market`	x		

2.2. Derivation of Gateway Identifiers

The FRC Hub maps logical names for the sender and receiver (`From` and `To` fields) to physical URL's. Throughout the life of the FBS, participants will have the need to run test data through the FBS to test both the internal and network components of their systems. It is important that test transactions are never confused with production systems. To facilitate this distinction, the FBS administration will supply all participants with two sets of digital certificates. One set will be for production systems, the other for test systems. The use of certificates to distinguish between environments means that participants can use the same Gateway Identifier for both the production and Pre-Production environments, removing the need for code changes within each participant's migration process. While Gateway Identifiers on the two hubs will be the same, each hub will hold a distinct profile for each participant, with participants supplying the URL's to which each profile maps.

Gateway Identifiers in the FBS will be mechanically derived from the Participants GBO IDs, a set of unique ten-character Id's assigned to industry participants. These Gateway Identifiers are used as the content of the aseXML `From` and `To` elements.

The content of the ebXML `From` field will be the same Gateway Identifier as is contained in the aseXML From field.

The content of the ebXML To field will be the same Gateway Identifier as is contained in the aseXML `To` field.

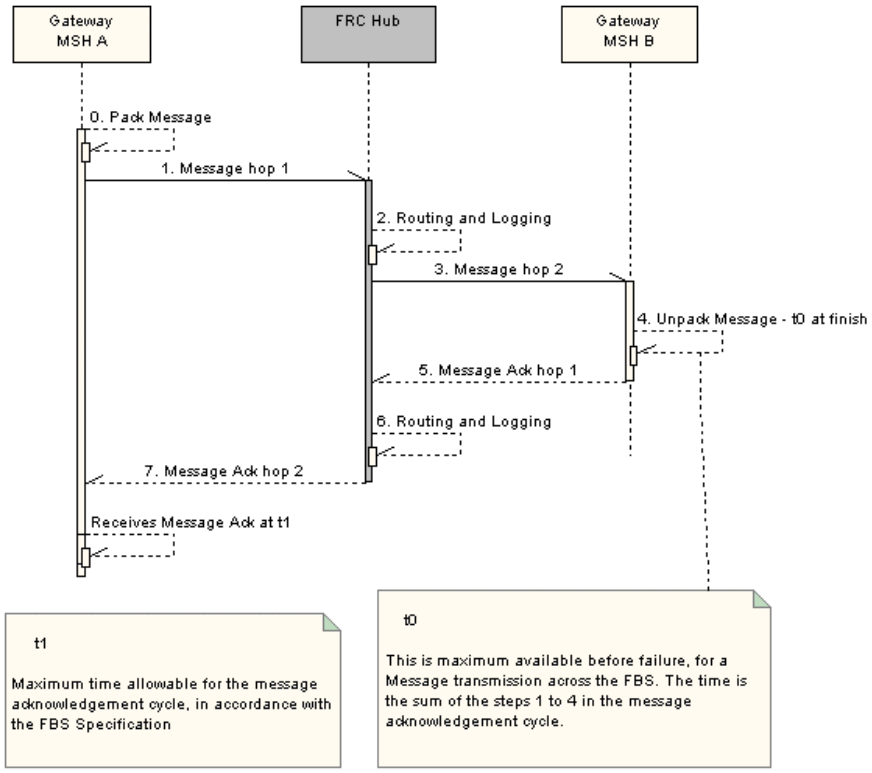
Gateway Identifiers in the FBS will be used as the PartyId element – see section 4.3.1.

2.3. Timing implications

2.3.1. Definitions

Allowable time drift on Message Service Handlers is ± 15 seconds, hence maximum time error will be 30 seconds.

Timing steps in the Message cycle are according to the following diagram:



t0 is the maximum time to pass before transaction processing can begin.

For high priority messages there are no retries

t0 = sum(steps 1 to 4 in Definition 2.)

For other medium and low priority messages t0 = t1

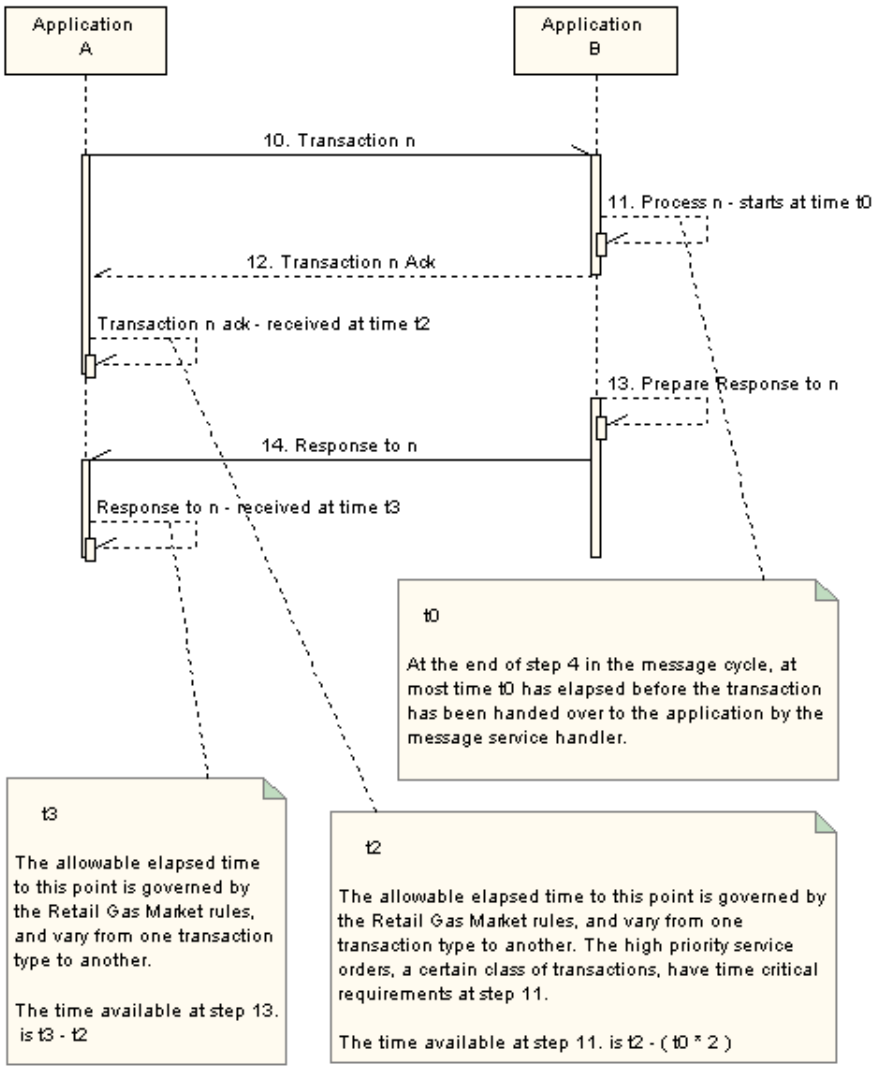
t1 is the maximum time to pass before message acknowledgment receipt

high priority t1 = 90 seconds message cycle + 30 seconds drift error

= 2 minutes

medium priority and low priority t1 = 1 hour

Timing Steps in the Transaction and Response Cycle are according to the following diagram:



t2 is the maximum time to pass before transaction acknowledgement receipt

Subsequent to various discussions and calculations regarding functionality in back end systems, the Gas Transaction Protocol Working Group has decided the various transaction-applications will not of deliver the responsiveness required to support high priority transactions through the FBS. Transaction acknowledgement receipt times are in section 2.5

t3 is the maximum time to pass before a transaction response (if appropriate) is received. These times are governed by the Retail Market Procedures and will be policed by audit.

2.3.2. Critical Considerations

The FBS is not to be used for transmitting high priority service orders. If this situation changes, these are the relevant considerations.

Critical time considerations in implementing Message Service Handlers and transaction processing applications relate to steps 4 and 11 (from the definitions diagrams) high priority messages.

step 4 is the time available to process a Message Acknowledgment

step 11 is the time available to process a Transaction Acknowledgement

If **t2** expires unacknowledged, the sender must fallback to a secondary communication mechanism.

The total time available in responding to a high priority service order is **20 minutes**.

Hence the total time available to the secondary mechanism is **20 minutes – t2**. Adequate time must be left for the secondary mechanism if it is needed in the event of the primary mechanism failing.

2.4. Timestamps in the FBS

In the FBS all time instants will be expressed in conformance with the ISO 8601 specification, using the methodology that includes the time offset.

– See http://www.w3.org/TR/xmlschema-2/#dateTime.

An example of an acceptable expression of a time instant is

```
2002-03-01T10:23:32+10:00
```

This time indicates that the current time at the locale producing the timestamp is 10:23:32 and that the locale in question is ten hours ahead of GMT.

2.5. Transaction Acknowledgement times

The Gas Transaction Protocol Working Group has determined that the times by which a transaction acknowledgement will be received shall be as follows:

High priority transaction = 29 minutes

Medium priority transaction = 270 minutes

Low priority transaction = 5pm on the following business day

***Note**: High-priority B2B Service Order transactions will not use the FBS for transmission. They will be treated as 'emergency services' per the Victorian model and will require human interaction via phone or suitable communication.*

2.6. Size of aseXML Documents

2.6.1. High Priority Transactions

None defined.

2.6.2. Medium Priority Transactions

All other non-CSV transactions will be sent as Medium Priority aseXML documents, except for those identified in the Low Priority Transactions section below.

It is recommended that these should not be larger than 50 KB in size.

2.6.3. Low Priority Transactions

All CSV transactions will be sent as Low Priority aseXML documents.

The following fully tagged XML transactions will be sent as Low Priority transactions:

- Special Read Response
- Meter Data Verification Response
- Service Order Response
- Amend Site Access Details

The total maximum transaction size shall be 2 MB unless otherwise agreed in the relevant Retail Market Procedures.

2.7. aseXML Events

All aseXML transaction acknowledgements and some transactions may contain the aseXML Event element. The use of the Event element is described in the *Guidelines for Development of a Standard for Energy Transactions in XML (aseXML)* document and will not be described further here.

The purpose of this section is to describe how each element within the Event element will be used within the FBS.

2.7.1. class Attribute

Can be omitted as it has a default value of "Application". If used, must be set to "Application".

2.7.2. severity Attribute

As per the list of event codes defined by the industry. One severity will be assigned to each event code (ie same event code can not be sent with different severities).

The aseXML Guidelines indicate that an error that prevents further processing of a fully tagged aseXML transaction shall have a severity of "Fatal", while an error that prevents a single CSV row from being processed shall have a severity of "Error".

However as most participant back-end systems will not know if the data arrived in CSV or aseXML format, “Error” will be used as the severity for both fully tagged aseXML and CSV transactions if the information cannot be processed.

2.7.3. <Code> Element

As per list of event codes developed by industry.

Note that the “Description” attribute of the code remains optional. If this attribute is populated then it must match the description in the event code table developed by the industry and published as part of Participant Build Pack 3 as well as part of the SAWA B2M ICD and SAWA B2B Interface Definitions.

2.7.4. <KeyInfo> Element

Preferred

Key data required to uniquely identify the transaction or record in error (eg, if DPI was all that was needed to identify a particular record then KeyInfo would contain 1234567890). These fields have been identified by transaction and are included in the Event Code documentation in the B2M Interface Control Document and B2B System Interface Definitions document in the AEMO Specification Pack.

Minimum

Not required for aseXML transactions as the initiating transaction ID is included in the header (assumes back-end system knows the transaction was aseXML).

MIRN for majority of CSV records (this will at least narrow it down to a couple of records).

2.7.5. <Context> Element

Preferred

The portion of the data that is in error, including the field name and the data value. For aseXML transactions the field name as defined in the AEMO Specification Pack is to be used, whereas for CSV records the field name will be taken from the CSV Format Specification. Multiple values will be separated by a comma. (eg “MIRN=1234567890,Current_Index_Value=1234567”).

Minimum

Values of fields that are in error (may not be able to provide field names initially).

2.7.6. <Explanation> Element

Preferred

In the case of a CSV record, this should contain the entire CSV record containing the error.

In the case of an aseXML record, this may contain:

A freeform comment (including internal application errors), or

If the event code can be further clarified by providing more information, a description of the error condition.

Minimum

May contain any supporting comments or may be left blank.

2.7.7. <SupportedVersions> Element

Probably won’t be implemented by most systems, but shall currently be “r9” when used.

3. aseXML Header Content

This section contains information for formatting the aseXML header content specific to the FBS. This is to be read in conjunction with the *Guidelines for Development of A Standard for Energy Transactions in XML (aseXML)*.

3.1.1. From element

This mandatory element is to be populated with the GBO ID that applies to the enterprise sending the aseXML document. No context attribute is to be used.

3.1.2. To element

This mandatory element is to be populated with the GBO ID that applies to the enterprise for whom the aseXML document is intended. No context attribute is to be used.

3.1.3. MessageId element

This mandatory element is to be populated according to the Guidelines for Development of A Standard for Energy Transactions in XML (aseXML.

3.1.4. MessageDate element

This mandatory element is to be populated directly according to the Guidelines for Development of A Standard for Energy Transactions in XML (aseXML).

3.1.5. TransactionGroup element

This mandatory element is to be populated according to the *Guidelines for Development of A Standard for Energy Transactions in XML (aseXML)*. Each Transaction Group will be represented in the ebXML Action element under the same name as the Transaction Group. Transaction groups for each of the Vic Gas transactions are defined in the interface specifications within Participant Build Packs 2 and 3.

3.1.6. Priority element

This optional element has three allowable values for this field are “High”, “Medium”, and “Low”. When this element is not present “Medium” will be assumed to be the default by the ebXML Message Service Handlers. This element will be used to assign the ebXML message the appropriate service level, as is discussed in detail in the next chapter.

3.1.7. SecurityContext element

The security context element only has one use within the FBS system. That use is to define the aseXML document as one that contains test transaction(s). If the aseXML document does contain test data this element must be populated with “Test”. If it is a production document, this element shall be empty, or not present.

In the context of the FBS, “Test” is the only content of the SecurityContext element that has meaning.

3.1.8. Market element

While this element is optional according to the schema, it must be present, and be populated with the content “SAGAS” or WAGAS, as the default identifies AEMO.”

4. ebXML Message Service Handler Configuration

For any ebXML message, an ebXML message acknowledgement will need to be received within a time interval according to priority, as defined below; or an alert will need to be raised in the sending gateway. It is important to note that prior to sending the ebXML message acknowledgement the aseXML payload must be parsed.

It is the responsibility of the sending participant to take action if an acknowledgement to an ebXML message is not received within the timeframes for 'time-to-live' specified in this document. Each participant must have the necessary operational procedures to identify ebXML messages that have failed in this way and to transmit the contents of these messages as new ebXML messages. As specified in the B2M-B2B Hub System Architecture Document, if the persistDuration has passed since the message was first sent, a sending MSH should not resend a message with the same MessageId.

Participants can determine the approach and level of automation of the operations processes required to identify failed ebXML messages and re-transmit the contents. However, participants should ensure that these operations processes are carried out at least once per business day.

The parameters and elements referred to here are all described in detail in the associated *FRC B2M-B2B Hub System Architecture* document. The following element tree shows in bold type the ebXML elements that require special configuration the FBS, according to these details; those in plain type shall be treated as per the ebXML specification;

MessageHeader

- From
  - PartyId
- To
  - PartyId
- CPAId
- ConversationId
- Service
- Action
- MessageData
  - MessageId
  - Timestamp
  - RefToMessageId
  - TimeToLive
- QualityOfServiceInfo
- SequenceNumber
  - Description

TraceHeaderList

- **TraceHeader**

Manifest

Reference

Schema

ErrorList

Error

Signature

4.1. Services & Actions

The ebXML `Service` and `Action` elements will have specific names derived from the content of the aseXML header. It is incumbent on participating gateways to populate these fields according to these specifications. Failure to do so will render the ebXML message invalid, so while the message may be successfully transmitted, there will be no reason to assume that the receiver will be able to appropriately manage the aseXML payload.

Services and actions can be thought of as Classes and Methods in the Message Service Interface. Participants are free to implement these services and actions however they see fit, bearing in mind that these `Service` and `Action` names are the only mechanism by which specified processing of ebXML messages can be achieved.

It is by the use of these `Service` and `Action` names that appropriate interaction with the Message Service Interface is achieved.

In the event that an action is associated with the wrong service in an ebXML message, the receiving MSH will reject this ebXML message. Details about these mappings will be forthcoming when they are finalised. They are dependent on agreement on assigning priorities to Transaction groups.

4.1.1. Service element naming

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `Service` element.

When composing an ebXML message, where the message is to contain an aseXML document as payload

The `Service` element has a type attribute which shall be set to type="fbs"

Where the `Priority` element in the aseXML header is not present, the `Service` name will be MediumPriorityAseXMLDocument

Where the `Priority` element in the aseXML header is High, the `Service` name will be HighPriorityAseXMLDocument

Where the `Priority` element in the aseXML header is Medium, the `Service` name will be MediumPriorityAseXMLDocument

Where the `Priority` element in the aseXML header is Low, the `Service` name will be LowPriorityAseXMLDocument

4.1.2. Action Naming

The ebXML **MessageHeader** element is a composite element comprised of ten subordinate elements, one of which is the `Action` element.

When composing an ebXML message, where the message is to contain an aseXML document as payload the `Action` name will be the same as the name in the `TransactionGroup` element in the aseXML document Header.

4.2. Reliable Messaging Parameter Settings

The parameters `mshTimeAccuracy, retries, retryInterval,` and `persistDuration` can be managed in either of two ways by a participant receiving an ebXML message.

They can be set according to the assertions in this specification.

The participant can parse the `CPAId` element which is a subordinate element of the ebXML message header element and extract the parameter values from that element, according to the methodology described in section 4.3.4 pertaining to the `CPAId` element.

4.2.1. mshTimeAccuracy parameter

The time accuracy of Message Service Handlers in the FBS must be managed to be accurate within 15 seconds. That implies the total time drift between any two participants should be no more than 30 seconds. If participants can deliver better time accuracy than this, they should do so. The time accuracy is expressed in `mm:ss` format. For example, a gateway with an accuracy of +- 5 seconds would be expressed in the `CPAId` element as

00:05

4.2.2. retries parameter

Number of Retries
2

This shall be expressed directly as the appropriate integer value in the `CPAId` element.

4.2.3. retryInterval parameter

Retry Interval
60 minutes

The retry interval is expressed as a duration in accordance with the XML Schema time duration data type – see http://www.w3.org/TR/xmlschema-2/#duration Here for example the retry interval in the `CPAId` element should be `PT60M`

4.2.4. persistDuration parameter

persistDuration in minutes
180

The `persistDuration` is expressed as a duration in accordance with the XML Schema time duration data type – see http://www.w3.org/TR/xmlschema-2/#duration Here for example the persist duration in the `CPAId` element should be `PT180M`

4.2.5. TimeToLive element

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `MessageData` element. The **`MessageData`** element is a composite element comprised of four subordinate elements, one of which is the `TimeToLive` element.

`TimeToLive` will be set a by adding a certain number of minutes to the current time according the priority, as per the following table

Priority	Time message is sent + duration
High	current time + 1 minute 30 seconds
Medium and Low	current time + 180 minutes

The `TimeToLive` element must be an XML schema time instant – see http://www.w3.org/TR/xmlschema-2/#dateTime. In the FBS all time instants will be expressed according to the ISO 8601 specification. An example of an acceptable expression of a time instant is

`2002-03-01T14:23:35+10:00`

See section 2.4.

4.3. ebXML Identifier elements

4.3.1. `PartyId` element

The content of this element is based on the GBO ID and the derivations for this are described in section 2.2. The `PartyId` values in the FBS are logical names that will be resolved to physical addresses in the FRC Hub. The complete list of current GBO IDs is available from AEMO website.

These logical identifiers for participants will be subject to additions and deletions on a dynamic basis.

The `PartyId` element has a type attribute, which in the FBS shall be present, as follows

type="urn:frchub.net"

4.3.2. From element

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `From` element. The `From` **element** shall contain the `PartyId` element of the sending participant gateway.

4.3.3. To element

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `To` element. The `To` element shall contain the `PartyId` element of the intended recipient participant gateway.

4.3.4. CPAId element

In the FBS the content of this element must be constructed as follows.

It will be comprised of six components. They are, in order,

From PartyId

To PartyId

mshTimeAccuracy

retries

- retryInterval
- persistDuration

These values shall be separated by a single whitespace. An example of a `CPAId` in a message from AEMO's gateway to the FBS test gateway would be as follows:

```
AEMOTEST FBSTEST 00:15 2 PT60M PT180M
```

4.3.5. MessageId element

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `MessageData` element. The **`MessageData`** element is a composite element comprised of four subordinate elements, one of which is the `MessageId` element.

The sender of the ebXML message is required to populate this element with a `Message Id` that is unique to the `PartyId` that identifies the sending Message Service Handler gateway. In the FBS the content of this element must be constructed using the `From PartyId,` white space separator followed by an identifier unique to that gateway. For example:

AEMO 654323456

`From PartyId` is not required in the case of messages that are automatically generated by the participant's Message Service Handler (eg Message Acknowledgements, Error Messages, Pings and Pongs').

4.3.6. ConversationId element

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `ConversationId` element. In accordance with the *ebXML Message Service Specification v1.0* conversations that do implicitly exist within ebXML such as a Message and Acknowledgement require the response (Acknowledgement) to contain the `ConversationId` that was supplied in the request (Message). Such conversation pairs are:

Message and Acknowledgement

Ping and Pong

Message Status Request and Message Status Response

Error messages should contain the `ConversationId` of the missive they are reporting an error on if they are capable of determining it.

For Messages, this mandatory element shall be populated by combining the `From PartyId`, the `To PartyId` and the `MessageId`; whitespace separated. For Pings and Message Status Requests any identifier unique to the sending gateway is sufficient.

An example of a `ConversationId` in a Message from AEMO's visa test gateway to the FBS test gateway would be as follows:

AEMO FBSTEST AEMO 5432112345

This element is not used in the FBS to span multiple payload bearing messages.

4.3.7. SequenceNumber element

The ebXML **`MessageHeader`** element is a composite element comprised of ten subordinate elements, one of which is the `SequenceNumber` element. Currently the *FRC B2B Hub System Architecture* document asserts that this element is not used in the FBS. This has been the subject of review and currently there are no plans to use this element.

4.4. Routing through the FRC Hub

Two hubs are in operation. The first is the Pre-Production hub, which is available for testing and debugging purposes. The second is the production hub, which is used for production transactions.

No messages that are for testing purposes will be routed through the production hub.

Routing messages through these hubs is achieved as follows.

4.4.1. TraceHeaderList Element

The `TraceHeaderList` element is a composite element made up of one or more `TraceHeader` elements. In the FBS there will be one `TraceHeader` element in outgoing messages, and two `TraceHeader` elements in incoming messages. The `TraceHeader` element contains information that enables an ebXML message to make a single hop. This implies for all participants that the outgoing `TraceHeader` element contains the information to get an ebXML message from MHS gateway to the FRC Hub.

It also implies for all participants that all received messages will have two `TraceHeader` elements in the `TraceHeaderList`. The first will be the `TraceHeader` element that applies to the sender to hub hop, and the second will be the hub to receiver hop.

1.1.1. `TraceHeader` element

The `TraceHeader` is a composite element comprised of four subordinate elements, `Sender`, `Receiver`, `Timestamp`, and `#wildcard`. The `#wildcard` element is not used. The use of the `TraceHeader` element will be as per the ebXML specification, in conjunction with the following criteria.

The `TraceHeader Receiver` element is a composite element comprising two subordinate elements. These elements are the `PartyId` and the `Location`. For all participants this `Receiver` element will refer to the FRC Pre-Production Hub or the production FRC Hub. Addressing is as follows:

Routing through the FRC Pre-Production Hub

Prior to the use of certificates, participants will need a username-password pair to connect. These are generated as part of the form based process of a participant registering their physical address with the FBS administration.

Participants populate the `PartyId` element of the `Receiver` element with the FRC Hub `PartyId` and this is

RELAY

During the testing phase, participants populate the `Location` element of the `Receiver` element with the FRC Pre-Production Hub. For HTTP this is

`http://preprod.frchub.net:5318/invoke/relay/inbound/`

When certificates are used this is

`https://preprod.frchub.net:5319/invoke/relay/inbound/`

Routing through the production FRC Hub

Participants populate the `PartyId` element of the `Receiver` element with the FRC Hub `PartyId` and this is

RELAY

Participants populate the `Location` element of the `Receiver` element with the FRC Pre-Production Hub.

`https://www.frchub.net:5319/invoke/relay/inbound/`

4.5. Identifying payload data

In the FRC HUB, prior to routing a message to a recipient, the aseXML payload document will be parsed and determined to be schema valid or otherwise. In the event that it is schema invalid, the payload reference features of ebXML will be deployed and used in conjunction with the error handling described in section 4.6 of this document.

4.5.1. Manifest element

The manifest element is a composite element consisting of one or more `Reference` elements. The manifest element is used as per the *ebXML Message Service Specification v1.0.*

4.5.2. Reference element

The `Reference` element is comprised of a number of attributes and component elements as detailed in the *ebXML Message Service Specification v1.0.* The only attributes of the `Reference` element that are used in the FBS shall be populated as follows:

xlink:type=”simple”

xlink:href="cid:aseXML"

It is important to note that the `xlink:href` attribute refers to the Content-Id of the payload. In the FBS all payloads are aseXML, hence a MIME part with a Content-Id of aseXML must be present in the payload.

4.5.3. Schema element

The Schema element is the only component element of the Reference element that is required in the FBS. This element has two attributes as shown in the following example.

location=

"http://www.aemo.com.au/aseXML/schemas/r9/1.3/aseXML_r9.xsd"

version="r9"

4.6. ebXML Error reporting and handling

The FRC HUB will employ the ebXML error reporting and handling as per the *ebXML Message Service Specification v1.0* with the following addition.

In the FRC HUB, prior to sending an ebXML message acknowledgement, a schema validation parse is applied to the aseXML payload. If the parse fails a “message reporting the error” is sent. This is a message that contains an ebXML `ErrorList` element.

There are no custom error codes in the FRC HUB and the `codeContext` attribute of the `ErrorList` element will be the *ebXML Message Service Specification v1.0* default value.

4.6.1. ErrorList element

In the event of the payload being aseXML schema invalid this element shall contain at least one `Error` element, which will be discussed shortly. As the severity in that error element will be “Error”, the highest severity attribute of the `ErrorList` element shall be “Error”, as per the specification. All other attributes shall be as per the specification.

4.6.2. Error element

In the event of the payload being aseXML schema invalid, these attributes of this element shall be set as follows:

errorCode=”OtherXML”

severity=”Error”

location=

”//soap:Envelope/soap:Body/eb:Manifest/eb:Reference/eb:Schema”

The content of the `Error` element shall say at least:

aseXml schema validation failure on payload

If any additional text information about the schema validation failure is available, this should also be included in the content of the error element.

The remainder of the error element shall be as per the *ebXML Message Service Specification v1.0*

4.7. Signature element

The FBS employs the ebXML Signature handling as per the *ebXML Message Service Specification v1.0* with the following clarification.

The SignedInfo element, contained within the Signature element shall contain the recommended second Reference element that refers to the payload object. This Reference element is important in supporting non-repudiation. As per the *ebXML Message Service Specification v1.0* this element shall have a URI attribute that resolves to the payload object.

5. System Security

Note: *The following Security architecture only applies to ebXML/aseXML messages routed via the HUB. Other message transport methods, as per Appendix C, may not use the certificate based security architecture discussed here.*

5.1. Custom ports

The FRC Hub will use custom ports as a defence against simple Denial Of Service attacks.

for HTTP/S the port will be 5319

for HTTP the port will be 5318. This port is for testing only and will only be available on the FRC Pre-Production Hub

Participants may nominate any port between 5318 and 5330 in the URLs they supply to the FBS administration. These URL’s apply to the participant gateways.

5.2. Obtaining certificates

Obtaining X509 certificates for use with TLS and digital signatures in the FBS is achieved by following the Manage TLS Certificates online instruction {Placeholder TBA}.

5.3. FRC Hub Security Services

In order to solve the “many to many” problem with regard to certificate distribution and revocation, participants in the FBS, the hub will provide security services. Participants will only need one certificate - that of the FRC Hub for TLS and digital signature services.

Messages will be signed by the From party private key and be routed to the To party via the FRC Hub.

The hub will verify this signature against the participant’s certificate. The hub is the only repository of certificates issued by the associated FBS Certificate Authority (FBS-CA). Individual participants will not need certificates from other participants.

The hub will then re-sign the message with the FRC Hub private key and forward the message to the To party. The To party then verifies the message signature against the FRC Hub certificate

A motivation for using ebXML was for a technology that provided for a signed payload to provide for non-repudiation of receipt. Using the webMethods hub security services does not change the certainty of non-repudiation of receipt; it simply changes the procedures to be followed in the event of dispute resolution. The FBS Administration will describe these in detail in due course; but in short non-repudiation of receipt will be verifiable as follows:

In the event there is a dispute about an alleged difference in an aseXML payload document both parties will supply their copies of the aseXML payload document in question to the FBS Administration. If these documents do differ:

The sender will be asked to sign and send both documents using the same hash-code algorithm. The hash-codes that are generated will be checked against the hash-code and algorithm reference that is archived at the hub.

The FRC Hub will re-sign both documents against the hash-code algorithm reference that is archived at the hub and compare these hash-codes with the appropriate hash-code reference that is archived at the FRC hub.

The outcome of this process will reveal exactly what content was sent.

The ebXML Message Service Specification provides for multiple signatures on the payload to facilitate the sort of process described here, whilst providing the benefit of leaving the sending signature intact. Vendor tool support for this functionality is not yet present, so this methodology will not be used in the FBS.

5.4. TLS

The FRC Hub is capable of TLS connectivity in compliance with TLS TLSv1.2. TLS is based on RFC5246:

https://www.ietf.org/rfc/rfc5246.txtThe certificates used in the FBS will not employ any extensions. Sometimes these certificates extensions are referred to as server certificates by certain vendors. Certificates without extensions are sometimes referred to as client certificates by certain vendors.

Appendix A. ebXML MSv1.0 FBS Reference Schema

As there is no online reference schema, the FBS will publish a reference schema for the *ebXML Message Service v1.0* for use within the FRC B2B System. If a more widely available schema becomes available, the FBS Administration may drop this reference.

Participants should note that the use of tools such as XMLSpy may report that schema's referenced within this schema are written to an old standard, and prompt to update. Future versions of this schema will resolve this issue.

This is the FBS Administration messageHeader.xsd version 0.1

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="http://www.ebxml.org/namespaces/messageHeader"

    xmlns:xml="http://www.w3.org/XML/1998/namespace"

    xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"

    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"

    xmlns:xlink="http://www.w3.org/1999/xlink"

    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"

    xmlns="http://www.w3.org/2000/10/XMLSchema" version="1.0">

    <import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.ebxml.org/project_teams/transport/xmldsig-core-
schema.xsd"/>

    <import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://www.ebxml.org/project_teams/transport/xlink.xsd"/>

    <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
schemaLocation="http://www.ebxml.org/project_teams/transport/envelope.xsd"/>

    <import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.ebxml.org/project_teams/transport/xml_lang.xsd"/>

    <!-- MANIFEST -->

    <element name="Manifest">

        <complexType>

            <sequence>

                <element ref="eb:Reference" maxOccurs="unbounded"/>

                <!-- <any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>  -->

            </sequence>

            <attribute ref="eb:id"/>

            <attribute ref="eb:version"/>

            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
```

```
        </complexType>
    </element>
    <element name="Reference">
        <complexType>
            <sequence>
                <element ref="eb:Schema" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="eb:Description" minOccurs="0"
maxOccurs="unbounded"/>
                <!-- <any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>  -->
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="xlink:type" use="fixed" value="simple"/>
            <attribute ref="xlink:href" use="required"/>
            <attribute ref="xlink:role"/>
        </complexType>
    </element>
    <element name="Schema">
        <complexType>
            <attribute name="location" type="uriReference" use="required"/>
            <attribute name="version" type="eb:non-empty-string"/>
        </complexType>
    </element>
    <!-- MESSAGEHEADER -->
    <element name="MessageHeader">
        <complexType>
            <sequence>
                <element ref="eb:From"/>
                <element ref="eb:To"/>
                <element ref="eb:CPAId"/>
                <element ref="eb:ConversationId"/>
                <element ref="eb:Service"/>
                <element ref="eb:Action"/>
                <element ref="eb:MessageData"/>
                <element ref="eb:QualityOfServiceInfo" minOccurs="0"/>
```

```
                <element ref="eb:Description" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="eb:SequenceNumber" minOccurs="0"/>
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="eb:version"/>
            <attribute ref="soap:mustUnderstand"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
        </complexType>
    </element>
    <element name="CPAId" type="eb:non-empty-string"/>
    <element name="ConversationId" type="eb:non-empty-string"/>
    <element name="Service">
        <complexType>
            <simpleContent>
                <extension base="eb:non-empty-string">
                    <attribute name="type" type="eb:non-empty-string"/>
                </extension>
            </simpleContent>
        </complexType>
    </element>
    <element name="Action" type="eb:non-empty-string"/>
    <element name="MessageData">
        <complexType>
            <sequence>
                <element ref="eb:MessageId"/>
                <element ref="eb:Timestamp"/>
                <element ref="eb:RefToMessageId" minOccurs="0"/>
                <element ref="eb:TimeToLive" minOccurs="0"/>
            </sequence>
        </complexType>
    </element>
    <element name="MessageId" type="eb:non-empty-string"/>
    <element name="TimeToLive" type="timeInstant"/>
```

```
    <element name="QualityOfServiceInfo">
        <complexType>
            <attribute name="deliverySemantics"
type="eb:deliverySemantics.type" use="default" value="BestEffort"/>
            <attribute name="messageOrderSemantics"
type="eb:messageOrderSemantics.type" use="default" value="NotGuaranteed"/>
            <attribute name="deliveryReceiptRequested"
type="eb:signedUnsigned.type" use="default" value="None"/>
        </complexType>
    </element>
    <!-- TRACE HEADER LIST -->
    <element name="TraceHeaderList">
        <complexType>
            <sequence>
                <element ref="eb:TraceHeader" maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="eb:version"/>
            <attribute ref="soap:mustUnderstand" use="required"/>
            <attribute ref="soap:actor" use="required"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
        </complexType>
    </element>
    <element name="TraceHeader">
        <complexType>
            <sequence>
                <element ref="eb:Sender"/>
                <element ref="eb:Receiver"/>
                <element ref="eb:Timestamp"/>
                <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="eb:id"/>
        </complexType>
    </element>
```

```
    <element name="Sender" type="eb:senderReceiver.type"/>
    <element name="Receiver" type="eb:senderReceiver.type"/>
    <element name="SequenceNumber" type="positiveInteger"/>
    <!-- DELIVERY RECEIPT -->
    <element name="DeliveryReceipt">
        <complexType>
            <sequence>
                <element ref="eb:Timestamp"/>
                <element ref="ds:Reference" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
                <attribute ref="eb:id"/>
                <attribute ref="eb:version"/>
                <anyAttribute
namespace="http://www.w3.org/2000/10/XMLSchema-instance"
processContents="lax"/>
                <!-- <attribute name="signed" type="boolean"/> -->
        </complexType>
    </element>
    <!-- ACKNOWLEDGEMENT -->
    <element name="Acknowledgment">
        <complexType>
            <sequence>
                <element ref="eb:Timestamp"/>
                <element ref="eb:From" minOccurs="0"/>
                <element ref="ds:Reference" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="eb:version"/>
            <attribute ref="soap:mustUnderstand" use="required"/>
            <attribute ref="soap:actor" use="required"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
        </complexType>
    </element>
```

```
    <!-- ERROR LIST -->
    <element name="ErrorList">
        <complexType>
            <sequence>
                <element ref="eb:Error" maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="eb:version"/>
            <attribute ref="soap:mustUnderstand" use="required"/>
            <attribute name="highestSeverity" type="eb:severity.type"
use="default" value="Warning"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
        </complexType>
    </element>
    <element name="Error">
        <complexType>
            <attribute ref="eb:id"/>
            <attribute name="codeContext" type="uriReference"
use="required"/>
            <attribute name="errorCode" type="eb:non-empty-string"
use="required"/>
            <attribute name="severity" type="eb:severity.type" use="default"
value="Warning"/>
            <attribute name="location" type="eb:non-empty-string"/>
            <attribute ref="xml:lang"/>
        </complexType>
    </element>
    <!-- STATUS RESPONSE -->
    <element name="StatusResponse">
        <complexType>
            <sequence>
                <element ref="eb:RefToMessageId"/>
                <element ref="eb:Timestamp" minOccurs="0"/>
            </sequence>
            <attribute ref="eb:id"/>
```

```
            <attribute ref="eb:version"/>
            <attribute name="messageStatus" type="eb:messageStatus.type"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
        </complexType>
    </element>
    <!-- STATUS REQUEST -->
    <element name="StatusRequest">
        <complexType>
            <sequence>
                <element ref="eb:RefToMessageId"/>
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="eb:version"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
        </complexType>
    </element>
    <!-- VIA -->
    <element name="Via">
        <complexType>
            <sequence>
                <element ref="eb:CPAId" minOccurs="0"/>
                <element ref="eb:Service" minOccurs="0"/>
                <element ref="eb:Action" minOccurs="0"/>
            </sequence>
            <attribute ref="eb:id"/>
            <attribute ref="eb:version"/>
            <attribute ref="soap:mustUnderstand" use="required"/>
            <attribute ref="soap:actor" use="required"/>
            <attribute name="syncReply" type="boolean"/>
            <attribute name="reliableMessagingMethod" type="eb:rmm.type"/>
            <attribute name="ackRequested" type="eb:signedUnsigned.type"
use="default" value="None"/>
            <anyAttribute namespace="http://www.w3.org/2000/10/XMLSchema-
instance" processContents="lax"/>
```

```
        </complexType>
    </element>
    <!-- COMMON TYPES -->
    <complexType name="senderReceiver.type">
        <sequence>
            <element ref="eb:PartyId" maxOccurs="unbounded"/>
            <element name="Location" type="uriReference"/>
        </sequence>
    </complexType>
    <simpleType name="messageStatus.type">
        <restriction base="NMTOKEN">
            <enumeration value="UnAuthorized"/>
            <enumeration value="NotRecognized"/>
            <enumeration value="Received"/>
            <enumeration value="Processed"/>
            <enumeration value="Forwarded"/>
        </restriction>
    </simpleType>
    <simpleType name="type.type">
        <restriction base="NMTOKEN">
            <enumeration value="DeliveryReceipt"/>
            <enumeration value="IntermediateAck"/>
        </restriction>
    </simpleType>
    <simpleType name="messageOrderSemantics.type">
        <restriction base="NMTOKEN">
            <enumeration value="Guaranteed"/>
            <enumeration value="NotGuaranteed"/>
        </restriction>
    </simpleType>
    <simpleType name="deliverySemantics.type">
        <restriction base="NMTOKEN">
            <enumeration value="OnceAndOnlyOnce"/>
            <enumeration value="BestEffort"/>
        </restriction>
```

```
    </simpleType>
    <simpleType name="non-empty-string">
        <restriction base="string">
            <minLength value="1"/>
        </restriction>
    </simpleType>
    <simpleType name="rmm.type">
        <restriction base="NMTOKEN">
            <enumeration value="ebXML"/>
            <enumeration value="Transport"/>
        </restriction>
    </simpleType>
    <simpleType name="signedUnsigned.type">
        <restriction base="NMTOKEN">
            <enumeration value="Signed"/>
            <enumeration value="Unsigned"/>
            <enumeration value="None"/>
        </restriction>
    </simpleType>
    <simpleType name="severity.type">
        <restriction base="NMTOKEN">
            <enumeration value="Warning"/>
            <enumeration value="Error"/>
        </restriction>
    </simpleType>
    <!-- COMMON ATTRIBUTES and ELEMENTS -->
    <attribute name="id" type="ID" form="unqualified"/>
    <attribute name="version" type="eb:non-empty-string" use="fixed"
value="1.0"/>
    <element name="PartyId">
        <complexType>
            <simpleContent>
                <extension base="eb:non-empty-string">
                    <attribute name="type" type="eb:non-empty-string"/>
                </extension>
```

```
            </simpleContent>
        </complexType>
    </element>
    <element name="To">
        <complexType>
            <sequence>
                <element ref="eb:PartyId" maxOccurs="unbounded"/>
            </sequence>
        </complexType>
    </element>
    <element name="From">
        <complexType>
            <sequence>
                <element ref="eb:PartyId" maxOccurs="unbounded"/>
            </sequence>
        </complexType>
    </element>
    <element name="Description">
        <complexType>
            <simpleContent>
                <extension base="eb:non-empty-string">
                    <attribute ref="xml:lang"/>
                </extension>
            </simpleContent>
        </complexType>
    </element>
    <element name="RefToMessageId" type="eb:non-empty-string"/>
    <element name="Timestamp" type="timeInstant"/>
</schema>
```

Appendix B. sample ebXML message

The following sample is of an unsigned ebXML message with a CATS transaction group aseXML payload.

```
Message-ID: <422518658.1019188709812.JavaMail.chatton@VENCORP00231>
Content-ID: <ac1b3ce8f70dc61800003fdd>

<?xml version="1.0"?>
<soap:Envelope
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
</soap:Envelope>

------=_Part_15_1170120589.1019188709621

Content-Type: application/xml
Content-Transfer-Encoding: binary
Content-ID: <aseXML>

<?xml version="1.0"?>
<ase:aseXML
</ase:aseXML>

------=_Part_15_1170120589.1019188709621--
```

Appendix C. Additional Transport Information in Interface Control Document (ICD)

The reader is referred to the SAWA Interface Control Document (ICD) for additional transport treatment information on:

Type of Dataflow	Description
AseXML	An automated ebXML/aseXML transaction, routed via the Hub
Bulk electronic file	A csv file which has a defined structure, which can be automatically processed, but does not have aseXML wrapping. This file has a non-specific method of transport. It can be delivered by any means other than ebXML/aseXML or secure FTP.
automated electronic file	A csv file which has a defined structure, which can be automatically processed, but does not have aseXML wrapping. This file type will be transferred using secure ftp, as defined in ICD.
Notice	An unstructured instruction *in writing*, such as fax, physical letter, email etc. Minimum requirements for this type of communication are detailed in ICD
Acknowledgement	Dependent upon the method of transport for the initiating transaction, the acknowledgement of that transaction will be one of the following: An aseXML transaction acknowledgement, as defined in the aseXML guidelines document and outlined in ICD section 3.2.5 (defined) and 3.2.7 (outlined) An ftp csv acknowledgement, as defined in ICD section 3.4.2.5

***Note**: The System Security specification proposed in this document only applies to ebXML/aseXML messages routed via the HUB. Other message transport methods, as per this Appendix, may not use the Hub certificate based security architecture or certificates.*

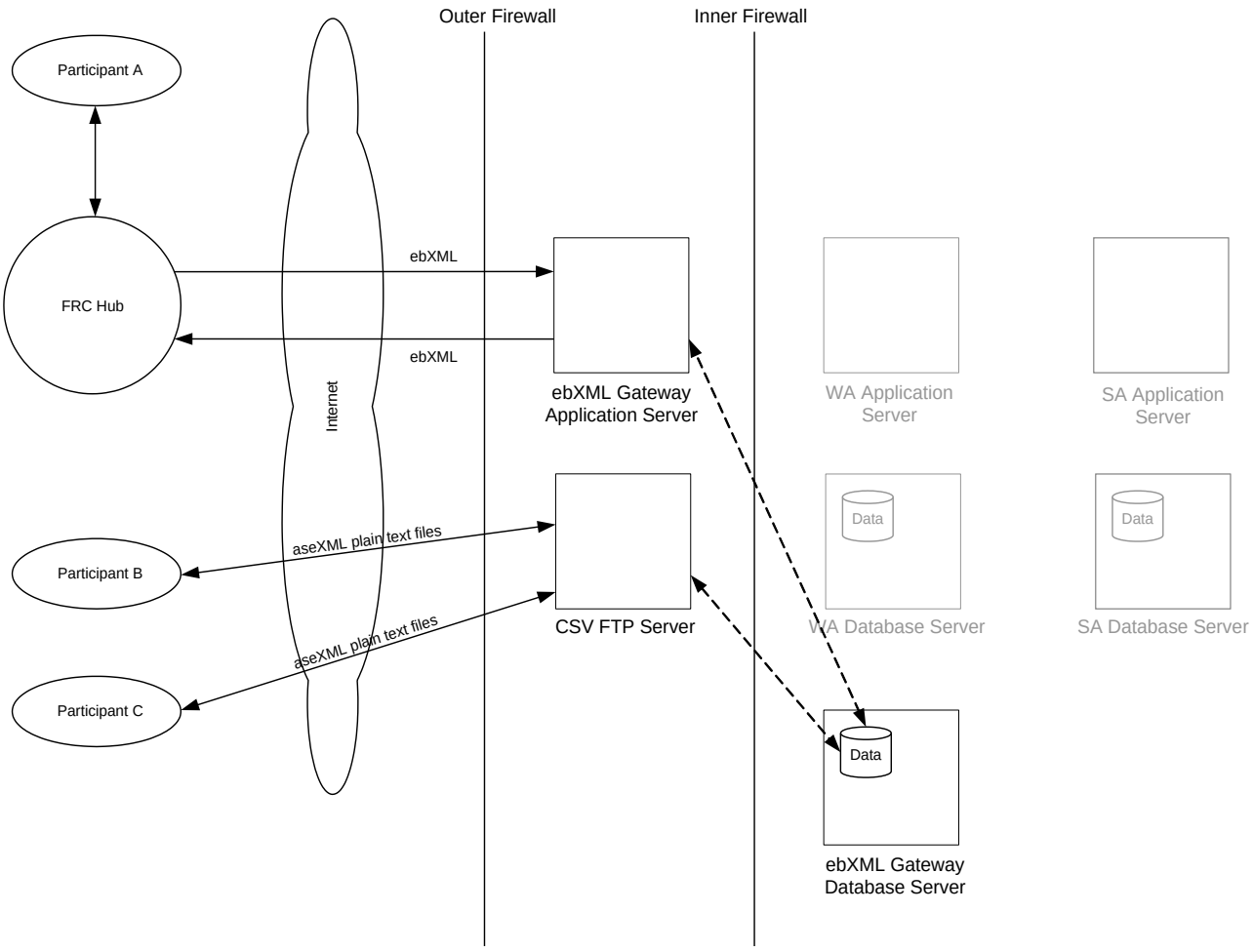
Appendix D. The Low Volume Interface (WA only)

D.1 Overview

AEMO will make availble the following low-volume interface service for Users that have a market share of no more than 500 MIRNs.

The low-volume interface provides participants who have a relatively small number of customers with an alternative mechanism for sending B2M and B2B transactions. By using this interface, these low-volume participants do not need to incur the costs associated with the implementation of an ebXML gateway.

The solution leverages the existing infrastructure and hardware in place for Gas Retail Market System (GRMS). The figure below shows how the solution works.



In the figure, Participant A is a participant with an ebXML gateway, whilst Participants B and C utilise the low-volume participant interface.

D.2 Restrictions for use of the Low Volume User Interface

The Low Volume Interface provides Participants with the ability to send and receive transactions via the FRC Hub without incurring the costs of implementing a full ebXML gateway. However, the interface is only suitable for low volumes of transactions:

- This interface is not designed to support high volumes of transactions. The performance of the low-volume interface would not be guaranteed at the levels expected of the ebXML gateway.
- The FTP side of the interface does not have the functionality of ebXML e.g. non-repudiation, security, guaranteed once-only delivery, retries and timeouts.
- AEMO will not check a transaction to ensure that it is valid. It will be passed on via an automated process. It is therefore crucial that the submitting participant validates their transactions against the correct version of the aseXML schema (using, for example, XMLSpy) prior to sending to the GRMS FTP server.

The residential market is, by nature, a volume market. Retailers who are competing in the mass residential market would be expected to have an ebXML gateway to the FRC Hub.

Use of the low volume interface is therefore suitable only for those organisations which are focused on commercial and industrial customers. The retailer may have some residential accounts and some customers with basic meters installed. However, based on AEMO's assessment of its operational costs and numbers of transactions, a retailer using the Low Volume Interface should have a market share of no more than 500 MIRNs.

D.3 Operation of the Low Volume User Interface

The simplest way to describe the operation of the interface is through a series of examples.

Example 1: Low Volume Participant to Participant with ebXML gateway

Imagine Participant B wishes to send a transaction to Participant A. Participant B would connect to the GRMS FTP server and transfer a plain text aseXML file to their 'aseXML in' folder. This would have "Participant B" as the 'FROM' party and "Participant A" as the 'TO' party. The GRMS CSV server would poll this folder and transfer the file to the ebXML interface table (located on the ebXML Gateway Database Server).

Participant B would know that the file was being forwarded when the file is removed from the inbox by the GRMS.

The ebXML gateway outgoing transaction processor would then pick up the aseXML from the table, put it into an ebXML message and send it to the FRC Hub.

The FRC Hub would then forward the message to Participant A. Participant A would send a message acknowledgement back through the Hub to the GRMS Gateway. The message acknowledgement would not be forwarded back to participant B (however a transaction acknowledgement would be forwarded back – see below).

Example 2: Participant with ebXML gateway to Low Volume Participant

Participant A now wishes to send a transaction to participant B. This could be the transaction acknowledgement to the transaction described above, or a totally new transaction.

Participant A sends a message to the hub with the transaction (or TACK) they wish to send to Participant B in the payload. The payload will have 'Participant B' as the 'TO' GBO ID.

The Hub is configured such that the GBO IDs of the low volume participants are associated with the GRMS Gateway ID.

The Hub will now see that the 'TO' GBO ID is 'Participant B'. The Hub will look up the associated Gateway Identifier and see that the message should be forwarded to the GRMS Gateway.

The GRMS Gateway incoming transaction processor will then receive the message and reply with a message acknowledgement. It will examine the message, see that the 'TO' GBO ID is a low volume interface participant and place the aseXML transaction payload into the ebXML interface table.

A new process on the FTP server will poll the ebXML interface table for records of type 'FTP aseXML' and forward to Participant B's 'aseXML out' folder. Participant B would then poll the folder to obtain his plain text aseXML transaction from Participant A.

Example 3: Low Volume Participant to GRMS

This works in the same way as example 1, except that when the hub receives the message (From Participant B over the GRMS gateway), it would see that the recipient GBO ID was 'GRMS' and would send the message back through the GRMS gateway connection. The GRMS processes in the GRMS gateway would treat this as if it came from an external participant and process the message and transaction as normal.

Example 4: GRMS to Low Volume Participant

Again, the transaction would go out through the GRMS gateway to the hub, the hub would see that the GBO ID was 'Participant B' and forward the message back through the GRMS Gateway ID connection. The GRMS Gateway Incoming transaction processor would see this transaction was addressed to 'Participant B' and forward through the same process as described in Example 2 above.

Example 5: Low Volume Participant to Low Volume Participant

In this example, Participant B sends a transaction to Participant C. This situation would also be catered for, as the GBO IDs of both Participant B and Participant C will be registered against the GRMS Gateway ID in the AEMO hub. The message would be routed in the same manner as described in the above examples.

1.2. Key Benefits of this Solution

- Works for B2M and M2B
- Works for B2B in both directions between a gateway-enabled participant and a low-volume participant
- Works for B2B in both directions between two low-volume participants
- Messages (with aseXML payload) logged and stored in GRMS ebXML gateway
- Full ebXML benefits between GRMS and hub (and gateway enabled participants) e.g. non-repudiation, security, guaranteed once only delivery, retries and timeouts. Note this does not apply on the ftp server side of the interface.

1.3. Assumptions

- AEMO will not check a transaction to ensure that it is valid. It will be passed on via an automated process. It is therefore crucial that the low-volume participant validates their transactions against the correct version of the schema (using, for example, XMLSpy) prior to sending to the GRMS FTP server.
- This interface is by its nature very low volume and will not be designed to support high volumes of transactions. The performance of the low-volume interface would not be guaranteed at the levels expected of the ebXML gateway.
- The low-volume participant would have separate in and out folders for their aseXML transactions (as opposed to their BAR CSV folders), but could connect via the same connection. The participant would have a separate set of in and out boxes for each GBO ID.

Version release history

Version	Effective date	Author/s	Summary of changes
3.7	30/06/2023	D Freeman Nandu Datar	• IN001/23W Updated from SSL to TLS • IN005/23 – Transfer SA specific details
3.6	04/12/17	D. McGowan	SA Only – IN026/16 which include IN025/15.
3.5	3110/16	D. McGowan	• Update to include: WA – C02/16C – REMCo to AEMO transition changes. • SA – IN029/16 – REMCo to AEMO transition
3.4	01/01/2016	D. Martin	Update to include WA only change: • C04/15S - Hansen Reference in the Specification Pack
3.3	17/05/2015	Allan Ng	IN023/14 – FRC Hub Upgrade Project
3.2	1/1/14	D. McGowan	IN004/12 – Redundant Provision and minor GIP and Spec Pack changes
3.1	1/10/10	T Sheridan	Updated to reflect the relevant Market Operator requirements following the transfer of REMCo's SA retail market operations to AEMO.
3.0	9/3/04	B. Eaves	Minor amendments and clarifications to the introduction section.
2.01	28/11/03	D. Bone	• Correcting a versioning issue. • Renamed document from FRC (B2M&B2B) Transaction and Messaging Architecture to FRC B2M-B2B Hub System Architecture for clarity and market familiarity. • Change cross references from FRC (B2M&B2B) Transaction and Messaging Specifications to FRC B2M-B2B Hub System Specifications for clarity and market familiarity.
2.0	20/11/03	B. Eaves	Amendments following review with VENCorp
1.0	3/10/03	C Madden	Consistency check
0.2	2/10/03	D Bone	Consistency checks and disclaimers.
0.1	30/9/03	D. Bone	Minor changes to VENCorp FRC B2B System Architecture document to change terminology in relation to REMCo and South Australia and Western Australia.