|  |  |  |  |
| --- | --- | --- | --- |
| black_header_in_1cm | | | |
|  | |  |  |
| Standard Business Reporting    **SBR ebMS3**  **Web Services Implementation Guide (WIG)**  Date: 14th December 2017  Base-lined Suitable for use | | | |
|  | | | |
| attention_pms This document and its attachments are **Unclassified** | direction_pms **For further information or questions, contact the SBR Service Desk at** [**SBRServiceDesk@sbr.gov.au**](mailto:SBRServiceDesk@sbr.gov.au) **or call 1300 488 231. International callers may use +61-2-6216 5577** | | |
|  |

**VERSION CONTROL**

| Version | Revision Date | Summary of Change |
| --- | --- | --- |
| 1.7 | 14/12/2017 | **Sectrion 1.6.2.5 Implementation Guides**  Sentence added – **‘**For the ATO, variations to the implementation described in this guide can be found in the ATO ebMS3 Implementation Guide document.’.  **Section 2.6 Common Characteristics**   * The following change has been made in this section   From: “All response messages going out from SBR ebMS3 will be signed using the relevant agency’s key.”  To: “Only one-way pull response messages going out from SBR ebMS3 as a result of Bulk and Batch Pull request will be signed using the relevant agency’s key. All other responses will remain unsigned.”  **Section** **5.5 Business Event**   * Minor updates to refer to relevant agency documentation for information related to Business Event processing.   **Section 5.6 Message Events section and 5.6.2 Severity Code section**   * Added details for partially rejected requests.   **7.1.1 Service End Points**   * Table deleted – this information has been moved into a reference document ‘ATO SBR Physical End Points which also contains information on the SSL Certificates to be used. |

*Note: Prior Version History is located at Appendix C of this document.*

Copyright

© Commonwealth of Australia 2017 (see exceptions below).   
This work is copyright. Use of this Information and Material is subject to the terms and conditions in the "SBR Disclaimer and Conditions of Use" which is available at [http://www.sbr.gov.au](http://www.sbr.gov.au/software-developers/enabling-sbr-in-my-application/sbr-taxonomy/view-taxonomy). You must ensure that you comply with those terms and conditions. In particular, those terms and conditions include disclaimers and limitations on the liability of the Commonwealth and an indemnity from you to the Commonwealth and its personnel, the SBR Agencies and their personnel.

You must include this copyright notice in all copies of this Information and Material which you create.  If you modify, adapt or prepare derivative works of the Information and Material, the notice must still be included but you must add your own copyright statement to your modification, adaptation or derivative work which makes clear the nature of your modification, adaptation or derivative work and you must include an acknowledgement that the adaptation, modification or derivative work is based on Commonwealth or SBR Agency owned Information and Material. Copyright in SBR Agency specific aspects of the SBR Reporting Taxonomy is owned by the relevant SBR Agency.

**Contents**

[1. INTRODUCTION 5](#_Toc499543318)

[1.1 Purpose 5](#_Toc499543319)

[1.2 Caveats and known Issues 5](#_Toc499543320)

[1.3 Reference Documents 5](#_Toc499543321)

[1.4 Glossary 5](#_Toc499543322)

[1.5 Audience 6](#_Toc499543323)

[1.6 Context 6](#_Toc499543324)

[1.7 Terminology 9](#_Toc499543325)

[1.8 Namespaces 9](#_Toc499543326)

[2. SBR ebMS3 services 11](#_Toc499543327)

[2.1 Overview 11](#_Toc499543328)

[2.2 SBR ebMS3 Services 11](#_Toc499543329)

[2.3 Service Attributes 13](#_Toc499543330)

[2.4 Supported Service Invocation Types 15](#_Toc499543331)

[2.5 Web Service Standards 22](#_Toc499543332)

[2.6 Common Characteristics 23](#_Toc499543333)

[2.7 Message Implementation Guides 24](#_Toc499543334)

[3. Message Structure 26](#_Toc499543335)

[3.1 Security Header 27](#_Toc499543336)

[3.2 ebMS Header 28](#_Toc499543337)

[3.3 UserMessage SBR ebMS3 Profile 30](#_Toc499543338)

[3.4 SignalMessage SBR ebMS3 Profile 41](#_Toc499543339)

[3.5 Dates and Times 44](#_Toc499543340)

[3.6 Timeout Values 44](#_Toc499543341)

[3.7 Payload Format 44](#_Toc499543342)

[4. Message Packaging 45](#_Toc499543343)

[4.1 Single Request 45](#_Toc499543344)

[4.2 Single Receipt & Batch/Bulk Receipt 45](#_Toc499543345)

[4.3 Single Pull Request 45](#_Toc499543346)

[4.4 Single Response 46](#_Toc499543347)

[4.5 Batch /Bulk Request 46](#_Toc499543348)

[4.6 Batch/Bulk Pull Request (Selective Pull) 46](#_Toc499543349)

[4.7 Batch/Bulk Response 46](#_Toc499543350)

[5. Error Management 47](#_Toc499543351)

[6. Secure Communication with sbr ebMS3 64](#_Toc499543352)

[6.2 Implementation Options 65](#_Toc499543353)

[6.3 Security Token Service (STS) 65](#_Toc499543354)

[6.4 Secure Messaging 76](#_Toc499543355)

[6.5 Signature Structures 78](#_Toc499543356)

[7. Testing 81](#_Toc499543357)

[7.1 Overview 81](#_Toc499543358)

[7.2 Network Connectivity Testing 82](#_Toc499543359)

[7.3 Message Connectivity Testing 82](#_Toc499543360)

[7.4 Service Specific Testing 83](#_Toc499543361)

[8. Supporting Files 85](#_Toc499543362)

[APPENDIX A: ebMS Errors 86](#_Toc499543363)

[APPENDIX B: Supported pModes 89](#_Toc499543364)

[APPENDIX C: Previous Version History 101](#_Toc499543365)

# INTRODUCTION

## Purpose

The purpose of this document is to provide information that will assist software developers in the implementation of calls to the ebMS3 based services offered by the SBR ebMS3 platform.

## Caveats and known Issues

The current version of the SBR ebMS3 solution has following caveats and known issues:

* The SBR ebMS3 interface is subject to change and that may result in changes to this document.
* At the time of release of this document, SBR ebMS3 supports the following ebMS3 message exchange patterns:
  + Two-Way/Synchronous
  + One-Way/Push
  + One-Way/Selective Pull
  + Two-Way/Push-and-Pull
* As indicated in Section 6.4 all response messages and receipts are to be digitally signed using responding agency’s certificate. At the time of release of this document this behaviour is not yet fully implemented and only asynchronous pull responses are signed by SBR ebMS3 using the certificate of the responding agency.

## Reference Documents

| Ref | Name | Location – IPWC |
| --- | --- | --- |
|  | SOAPATTACH | SOAP Messages with Attachments, 2000.  http://www.w3.org/TR/SOAP-attachments |
|  | RFC2822 | http://www.ietf.org/rfc/rfc2822.txt |
| 3. | ebMS3 Version 3.0 of the OASIS ebXML Messaging Service specification | [Web Link](http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html) |

*Table 1: Reference Documents*

## Glossary

| Term / Acronym | Definition |
| --- | --- |
| AS4 | AS4 is a Conformance Profile of the OASIS ebMS 3.0 specification |
| ebMS3 | Version 3.0 of the OASIS ebXML Messaging Service specification |
| ebMS3 Advanced Features | Specification that complements the ebMS3.0 Core specification by specifying advanced messaging functionality for:   * messaging across intermediaries (multihop), * message bundling, * compressing, splitting and joining large messages, * and advanced processing modes. |
| SBR Core Services | The current **‘legacy’** Whole of Government platform that:   * provides the interface for SBR services for Business to Government reporting; and * routes received requests to the appropriate government agency system. |
| SBR ebMS3 | The **strategic** Whole of Government platform that:   * provides ebMS3 based SBR services for Business to Government reporting; and * routes received requests to the appropriate government agency system. |
| SDK | The software developer kit supplied by SBR to software developers to support integration of software packages with the SBR ebMS3 Platform. It includes the ebMS3 embeddable client and a ‘reference client’ that illustrates how to use the ebMS3 embedded client as well as other APIs such as the Vanguard Security Token Service (STS). |
| MSH | ebXML Messaging Service Handler (MSH) that implements the ebMS3 messaging functions. |
| Embeddable Client | The ebMS3 MSH implementation that can be embedded in business management software for ebMS3 interactions with SBR ebMS3. |
| BMS | Business Management Software: Software used by businesses and intermediaries to manage business finances and reporting obligations. It may include desktop accounting software, payroll software, components integrated with ERP systems, cloud based systems etc.  e.g., MYOB (accounting package) and SAP (payroll package). |
| Reference Client | The Reference Client is provided in the SDK as an example only. The Reference Client is included in the SDK to provide an example implementation of the components and APIs provided in the SDK. The Reference Client provides a guide for how software developers could use these in their developed products. The Reference Client is not supported for production use. It is not intended that the Reference Client be used by software developers for use with or embedded in their developed products. Ongoing support for the Reference Client and inclusion of the Reference Client in future SDK versions is not guaranteed. |

*Table 2: Glossary*

## Audience

The audience for this document is any organisation that will be building SBR ebMS3 services into their products. Typically this will be software application developers[[1]](#footnote-1).

Readers should be familiar with the following:

* SBR Program – please see <http://www.sbr.gov.au> for further information.
* XBRL – please see [www.xbrl.org](http://www.xbrl.org/) for further information.
* Web Services – please see [www.ws-i.org](http://www.ws-i.org/) for further information.
* ebMS3 – please see [ebMS3 Core features](http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html) and [Advanced features](http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/part2/201004/cs01/ebms-v3.0-part2-cs01.html), and [AS4](https://www.oasis-open.org/news/announcements/as4-profile-of-ebms-3-0-v1-0-published-as-a-committee-specification).
* AS4 – please see [AS4 profile of ebMS3.0 V1.0](https://www.oasis-open.org/news/announcements/as4-profile-of-ebms-3-0-v1-0-published-as-a-committee-specification).
* JSON – please see <http://www.json.org/> for further information.

## Context

### SBR ebMS3

The SBR program previously built and currently still offers a collection of services (SBR Core Services) that are part of the implementation of the SBR initiative to simplify Business to Government reporting obligations.

The next phase of the SBR program involves building and offering services that are based around the ebMS3/AS4 messaging standard (SBR ebMS3).

SBR ebMS3 differs from SBR Core Services mainly in the following ways:

* Messaging is based on the ebMS3 standard and AS4 Conformance Profile
* The addition of support for batch and bulk interactions
* The addition of support for asynchronous single interactions
* The addition of support for wider range of reporting obligations
* The support for additional payload formats (e.g. JSON)

### SBR ebMS3 knowledge repository

SBR ebMS3 offers a suite of documents and technical products to support software developers. These are illustrated in Figure 1 ‘SBR ebMS Artefacts’ below. Broadly speaking there are three groups of products:

* Architectural reference information such as the solution overview and taxonomy architecture that aim to explain what SBR is and how it works.
* Report specific implementation guides that provide the entry point for detailed information about how to implement specific business services such as an Activity Statement.
* General support material such as test plans, software development kits, and a knowledge repository that aim to facilitate efficient implementation.



*Figure 1: SBR ebMS3 Artefacts*

#### SBR ebMS3 Solution Overview

An overview of the SBR ebMS3 solution, including the business areas (agencies and forms) in scope and the main components of the solution, may be obtained from the SBR web site.

#### Taxonomy Architecture

This document describes the architecture of the SBR Taxonomy and shows how the library of harmonised data elements (the “SBR AU Definitional Taxonomy”) is packaged and how the data elements are re-used across government forms (the “SBR AU Reporting Taxonomies”). The document also defines the data element naming conventions, namespace conventions, file naming conventions, version control processes and provides a decision tree that defines the rules for choosing between different taxonomy implementation options.

#### SBR AU Definitional Taxonomy

This is the collection of XML schema and XML linkbases that constitute the SBR taxonomy. The “SBR AU Definitional Taxonomy” is organised into classifications representing the general functions of government and includes schema files and reference linkbases. The “SBR AU Report Taxonomies” are organised by agency/report and includes schema files, presentation, definition, label, and calculation linkbases as necessary. SBR AU Reporting Taxonomies are always built from data elements in the SBR AU Definitional Taxonomy.

#### Web Service Implementation Guides (WIG)

These documents describe common technical components and services that are re-used by all business services. The common services include whole of government gateways that expose services and supports the protocol for message exchange, standard message types, standard response time service levels, standard message structures, a security token service, and a standardised approach to handling business error conditions and transport exceptions.

There is a separate WIG document for each SBR platform i.e. SBR Core Services and SBR ebMS3.

#### Implementation Guides (MIG)

There is a Common MIG for each participating government agency where they have chosen to produce one. This contains message Implementation details that are common across all messages for that agency. It also describes the rules and guidelines for the SBR platforms that are common across all collaborations for that agency.

Some agencies have a MIG for each service/report in scope for SBR. The MIG is the entry point for an implementer wishing to support a specific SBR service or reporting obligation (e.g. Activity Statement or Payroll Tax). In many cases there are several message exchanges around a specific report (e.g. “list” previous lodgments, “pre-fill” with government data, “calculate” an obligation, and “lodge” a report).

Points in this document where the reader needs to refer to the MIG for report specific information are shown as references to “Agency Implementation Guide”.

For the ATO, variations to the implementation described in this guide can be found in the ATO ebMS3 Implementation Guide document.

#### Identity Management

The SBR ebMS3 solution leverages the AUSkey authentication credential that will be accepted by all participating agencies. It also explains how it is linked to agency business services to authorise primary credential holders or their delegates (employees or intermediaries). The SBR ebMS3 solution reuses the identity management approach that was used for SBR Core Services.

#### Software Developer Kit (SDK)

There are some common technical components that the SBR ebMS3 program expects will be needed by all implementers. The SDK is a set of components created for both Java and .NET platforms that are available for software developers to use in their products. These include:

* Client side ebMS3 Message Service Handler (MSH), i.e. ebMS3/AS4 embeddable client\*
* Identity/Authentication Management APIs
* Reference Client^

*\* IBM COTS embeddable client is only licensed for use by ELS related software packages AND for use by tax agents only.*

*^ The Reference Client is provided as an example only. The Reference Client is included in the SDK to provide an example implementation of the components and APIs provided in the SDK. The Reference Client provides a guide for how software developers could use these in their developed products. The Reference Client is not supported for production use. It is not intended that the Reference Client be used by software developers for use with or embedded in their developed products. Ongoing support for the Reference Client and inclusion of the Reference Client in future SDK versions is not guaranteed.*

#### Testing

The SBR ebMS3 program will provide implementers with a suite of test services that can be used to test the business (e.g. activity statement) implementations. Supporting the test services is a library of test credentials, Australian Business Numbers (ABN) and test data that can be assigned to developers and will be recognised by agencies.

The SBR ebMS3 program will also provide an ebMS3 ‘MSH Ping service’ which the SWD can use to ensure that they can reach the service endpoint and that their implementation can (from a technical perspective) send and receive approximately in the right format.

## Terminology

For definition of the terminology and acronyms used within this document, please refer to the [glossary](http://www.sbr.gov.au/software-developers/developer-tools/glossary) on the SBR website.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119](http://www.ietf.org/rfc/rfc2119.txt). The use of the word “Mandatory” is to be read as “MUST”.

## Namespaces

For brevity, namespace definitions are not included in all examples. The appearance of the following namespace prefixes SHALL be understood to refer to the corresponding namespaces from the table below.

| Prefix | Name Space |
| --- | --- |
| S11 | http://schemas.xmlsoap.org/soap/envelope/ |
| S12 | http://www.w3.org/2003/05/soap-envelope |
| xmime | http://www.w3.org/2005/05/xmlmime |
| xsi | http://www.w3.org/2001/XMLSchema-instance |
| wsse | http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd |
| wst | http://docs.oasis-open.org/ws-sx/ws-trust/200512 |
| iso4217 | http://www.xbrl.org/2003/iso4217 |
| eb | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/ |
| ebbpsig | http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0 |

*Table 3: Namespace Prefixes*

# SBR ebMS3 services

## Overview

The following diagram illustrates, at a high level, the design time and run time environment of the end-to-end SBR ebMS3solution.



*Figure 2: High Level Overview of the SBR ebMS3 Solution*

SBR ebMS3 mediates machine-to-machine interactions between Business and SBR ebMS3 (a B2G style of interaction) and SBR ebMS3 to participating Government Agencies.

The primary responsibility of the SBR solution is to seamlessly and securely mediate between business service requests and agencies by performing authentication of request message, routing messages between agencies and businesses, and performing various other intermediary roles like logging, message storage, splitting, joining and compression.

## SBR ebMS3 Services

SBR ebMS3 will offer a number of "Services". These Services and their related concepts are defined as follows:

### Service

"Service" is the equivalent of what is referred to as either "Form", "Business Object", and/or "Reporting Obligation". E.g. the Service is: Activity Statement, CTR, ClientRole, ClientDetails etc. A service can have one or more actions associated to it.

### Actions

"Action" is the operation requested against the service. That is: List, Get, Validate, Submit, Add, Lodge, etc.

### Service Versioning

There are two distinct patterns for service versioning i.e. versioning at the service level or versioning at the action level. Please refer to relevant agency Implementation Guide for the choice of service versioning adopted by the agency.

#### Service Level

In case of service versioning at the service level each service will have a version associated with it i.e. Service.VVV.vv.

Where VVV is the major version and vv is the minor version - e.g. CTR.001.00, CTR.002.00

#### Action Level

In case of service versioning at the action level each action will have a version associated with it i.e. Action.VVV.vv.

Where VVV is the major version and vv is the minor version - e.g. List.001.00, List.002.00, List.002.01.

### Interactions

"Interaction" is the combination of the Service and Action. For example:

* For service level versioning: LodgeCTR.001.00, ListAS.001.00, AddClientRole.001.00.
* For Action level versioning: Lodge.001.00CTR, List.001.00AS, Add.001.00ClientLevel.

### Mapping SBR ebMS3 Services to ebMS3 Services

The above SBR ebMS3 Service Concepts map to ebMS3 Service Concepts as follows (this is reflected in the mapping of values into the ebMS3 Header Specification for SBR ebMS3):

The below table is for service versioning at the action level:

| SBR ebMS3 Service Concept | ebMS3 Service Concept |
| --- | --- |
| SBR ebMS3 Service | ebMS3 Service |
| SBR ebMS3 Action.Version e.g. List.001.00, List.001.01 | ebMS3 Action |

*Table 4: Service versioning at the action level*

And for service versioning at the service level:

| SBR ebMS3 Service Concept | ebMS3 Service Concept |
| --- | --- |
| SBR ebMS3 Service.Version e.g. CTR.001.00, CTR.001.01 | ebMS3 Service |
| SBR ebMS3 Action | ebMS3 Action |

*Table 5: Service versioning at the service level*

## Service Attributes

SBR ebMS3 services may be invoked in a number of different ways. The options for invocation of each service are described by the following "Service Attributes":

### Invocation Modes

SBR ebMS3 supports two modes of invocation for business management system (BMS):

1. **Synchronous**

In the synchronous interaction pattern the BMS will send a request and will halt the process until it receives a response. While most responses will be returned within seconds, the BMS should be designed to cater for responses received up to 1 minute.

In the initial implementation, for ATO, this invocation mode will support only single requests.

1. **Asynchronous**

In the asynchronous interaction pattern, the BMS will a send request and will not wait for a response but instead will later poll for responses.

**Note:** The two way synchronous message exchange pattern (MEP) supported by SBR ebMS3 platform is not part of the AS4 conformance profile so it is not expected that all agencies will support this MEP. Please refer to relevant agency’s Implementation Guide or the ATO ebMS3 Implementation Guide to find out which of the invocation modes listed here are supported by the agency.

### Request Message Types

In this document, a “Logical Record” is defined as the structured business request data that must be submitted for a single invocation of a particular BMS interaction that can be invoked.

SBR ebMS3 accepts three distinct request message types:

1. **Single**Requests which contain a Payload of only one Logical Record i.e. a single invocation is being requested for a specific service action e.g. ListAS, ValidateFBT, LodgeCTR,etc. A Logical Record may however, be comprised of one or more “Logical Documents”. In the initial implementation for ATO, single requests can either be submitted synchronously or asynchronously.

Additionally, the Single Message Type has a sub-type called “Collection” – which is a special type of Single Request Message that is used to collect information or communications made available by an Australian Government Agency such as the ATO.

1. **Batch**Batch messages serve as a container for multiple Logical Records (Single or Bulk). A Batch Request Message may contain multiple Logical Records of the same type (e.g. Four LodgeCTRs) to be sent in one transmission, thereby facilitating what is effectively multiple invocations of an interaction using the one Request Message. A Batch Request Message can be one of 3 sub-Types:

* Batch of Standalone Forms or Service Requests
* Batch of Base Forms with Optional Schedules
* Batch of Bulk Requests

Note that the Logical Records in a Batch Request must all be of the same type. All batch requests are asynchronous.

1. **Bulk**A Bulk Request Message contains one Logical Record that is a multi-level construct comprised of a Parent (e.g.: Payer) and one or more Children (e.g.: Payees). The “Bulk” information is provided in the Children which are all of the same type and all relate to the Parent. In a Bulk Request Message each Child has a “business link” to the other children in that Request Message which is represented by the “Parent” e.g.: Private health insurance and member contribution statements. Like a batch, these requests can only be submitted in an asynchronous interaction pattern.

**Note:** Every agency may not support each of the above listed request message types. Please refer to relevant agency’s Implementation Guide or (for the ATO) the ATO ebMS3 Implementation Guide document to find out the one’s supported for that agency.

### Response Time Service Levels

SBR ebMS3 offers the following response time service levels:

1. **Chatty**  
   Applicable to requests with single message type. For synchronous requests response time service level is targeted at under 10 seconds and for asynchronous requests it is targeted at under 15 seconds\*
2. **Intermediate**  
   Applicable to batch requests, the rules over what is 'intermediate'^ is based on number of documents in a batch; this number varies depending on service/action for the batch (e.g. LodgeCTR takes longer than LodgeActivityStatement) and the rules will be flexible over time (e.g. may include time of day). For intermediate requests the response times (i.e. when the response is ready to be pulled) should be between >30 seconds and < 1 hour\*.
3. **Delayed**  
   Applicable to batch and bulk requests. Delayed^ requests have the longest response time generally greater than 1 hour\*.

\* As measured by arrival at SBR ebMS3 and response being available to return over the network. Response time targets are aspirational and are dependent on agency response times which may vary by service interaction type, number of schedules for a message, system load and time of day. Once SBR ebMS3 has been in production for a period of time these targets will be reviewed.

^ The criteria for ‘Intermediate’ and ‘Delayed’ response time service levels will specified by the agency. For example, ATO’s response service time levels for Intermediate and Delayed are follows:

A Batch transaction with more than 1000 logical records is classified as having a ‘Delayed’ response time service level and a batch transaction with =< 1000 logical records is classified as having an ‘Intermediate’ response service time level.

A Bulk transaction with more than 1000 ‘child’ records (logical documents) across all the parents in the same transaction is classified as having a ‘Delayed’ response time service level and a bulk transaction with =< 1000 logical documents across all the parents is classified as having an ‘Intermediate’ response service time level.

**Note**: Every agency may not support each of the above listed response time service levels. Please refer to relevant agency’s Implementation Guide or (in the case of the ATO) the ATO ebMS3 Implementation Guide document to find out the one’s supported for that agency.

### Message Partition Channel

Message Partition Channels (MPCs) is an ebMS3 concept which allows for partitioning the flow of messages from a Sending ebMS3 message service handler (MSH) to a Receiving MSH into several flows that can be controlled separately and consumed differently.

For each ebMS3 MSH implementation, there is a default MPC which is implicitly used for all messages unless new MPC’s are explicitly defined and referenced in the ebMS3 messages. SBR ebMS3 is using the default MPC only with no explicit definition for additional MPCs.

### Message Pulling

The initial release of SBR ebMS3 is aimed at software developers transitioning from ELS to SBR. This program requires the use of selective pulling which is not part of the AS4 profile but is one of the advanced features of ebMS3 standard. Other government agencies may not use the selective pulling feature.

Unlike normal pulling, selective pulling consists of pulling only from a subset of messages posted on a MPC, as defined by some common attribute value in their message header e.g. messages that are responses to some previous requests i.e. with a particular RefToMessageId value.

## Supported Service Invocation Types

SBR ebMS3 supports the following combinations of service attributes for invocation of its services, however not all agencies will support all of these service invocation types. Refer to agency specific MIGs for supported service invocation types.

| Message Type | Invocation Modes | Response Time Service Level | Service Invocation Type | ebMS3 MEP |
| --- | --- | --- | --- | --- |
| Single | Synchronous | Chatty | Single-Sync-Chatty | Two-Way/Synchronous |
| Single | Asynchronous | Chatty | Single-Async-Chatty | Two-Way/Push-and-Pull |
| Batch | Asynchronous | Intermediate | Batch-Async-Intermediate | One-Way/Push  One-Way/Selective-Pull |
| Batch | Asynchronous | Delayed | Batch-Async-Delayed | One-Way/Push  One-Way/Selective-Pull |
| Bulk | Asynchronous | Intermediate | Bulk-Async-Intermediate | One-Way/Push  One-Way/Selective-Pull |
| Bulk | Asynchronous | Delayed | Bulk-Async-Delayed | One-Way/Push  One-Way/Selective-Pull |
| Collect | Asynchronous | Delayed | Collect-Async-Delayed | Two-Way/Push-and-Pull |

*Table 6: Service Invocation Types*

Invocation of an SBR ebMS3 service using a particular service invocation type involves sending an appropriately formatted request to the appropriate endpoint and, in the case of asynchronous invocations, sending a pull request to the appropriate endpoint to retrieve the corresponding response message. The corresponding SBR ebMS3 logical endpoints that BMS systems must send these requests to are as follows:

| Logical Endpoint\* | Service Invocation Type Supported | Description | P-Mode |
| --- | --- | --- | --- |
| Single Synchronous | Single-Sync-Chatty | Accepts single-sync-chatty requests | Two Way Sync |
| Single Asynchronous | Single-Async-Chatty | Accepts the following pair of associated requests:  1) single-async-chatty push request  followed by  2) single-async-chatty pull request. | Two Way Async |
| Bulk and Batch Push | Batch-Async-Intermediate  Batch-Async-Delayed  Bulk-Async-Delayed | Accepts one way bulk/batch requests. The invocation for this endpoint should be followed by invocation(s) for the bulk/batch asynchronous selective-pull. | One Way Push |
| Bulk and Batch Pull | Batch-Async-Intermediate  Batch-Async-Delayed  Bulk-Async-Delayed | Accepts one way selective-pull request to retrieve the response for the corresponding push request. | One Way Pull |
| Collect | Collect-Async-Delayed | Accepts the following pair of associated requests:  1) collect-async-delayed push request  followed by  2) collect-async-delayed pull request. | Two Way Async |

*Table 7: Logical Endpoints*

\* Refer to agency specific Implementation Guides or (for the ATO) the ATO Physical End Point document for physical endpoint specification that corresponds to each of the logical endpoints.

### Single - Sync - Chatty

When a business management system wishes to submit a single request and wait for the response before continuing processing it should:

1. Construct a Single Request in conformance with the specification in section 4 ‘Message Packaging’ below;
2. Send the request to the Single-Synch Endpoint;
3. Wait for at least the specified SLA response time for the service action sought to be invoked and receive the response from SBR ebMS3 when it arrives during that specified SLA response time.

This sequence of interactions is illustrated in the following diagram:



*Figure 3 : Service Invocation Types – Single - Sync - Chatty*

### Single - Async - Chatty

When a BMS wishes to submit a single request and does not wish to wait for the response before continuing processing, but rather retrieve the response at a later time, the BMS will need to:

1. Construct a Single Request in conformance with the specification in section 4 ‘Message Packaging’ below;
2. Send the request to the Single-Async Endpoint;
3. Receive a receipt from SBR ebMS3 and then continue with "other" BMS processing (i.e. not wait for the business response before continuing with BMS processing);
4. Wait for at least the specified SLA response time for the service-action being invoked then begin polling for the response message by sending a pull request to the Single-Async Endpoint;
5. In the event that a pull request is unsuccessful, receive an error signal response with error code EBMS:0006 and description “EmptyMessagePartitionChannel” and then send another pull request in compliance with the intervals in the guidelines in section 2.4.5 ‘Polling Interval’ below;
6. Subject to the guidelines in section 2.4.5 ‘Polling Interval’, repeat step ‘e’ until such time as a pull request is successful (i.e. there is a response that is able to be pulled from SBR ebMS3) and then receive the response and process that response.

An example of this sequence of interactions is illustrated in the following diagram:



*Figure 4 : Service Invocation Types – Single - Async - Chatty*

### Batch - Async - Intermediate, Batch - Async – Delayed, Bulk - Async - Intermediate and Bulk - Async - Delayed

Batch-Async-Intermediate, Batch-Async-Delayed, Bulk-Async-Intermediate and Bulk-Async-Delayed generally all follow the same sequence of interactions between the BMS and SBR ebMS3 and therefore a common description of SBR ebMS3 service invocation using those service invocation types is given in this section.

#### Batch and Bulk Request Message Rules

* Services for processing Batch and Bulk Request messages can only be invoked in an asynchronous mode, i.e. a request needs to be sent to SBR ebMS3 and the corresponding response needs to be separately pulled from SBR ebMS3.
* Batch - Async – Intermediate and Bulk - Async – Intermediate request message will result in the generation of single response (ebMS3 User) message that will need to be pulled by the BMS:
  + Message containing both the Validation Response and the business response
* Batch-Async-Delayed and Bulk-Async-Delayed request message will result in two separate responses that will need to be pulled separately by the BMS :
  + a Validation Response delivered first followed by

a Business Response that includes both business and validation results.

* Each Batch request message may only contain one type of service-action - i.e. all of the transactions in a Batch must be for the same service-action.

#### Invocation of Services for Batch and Bulk Request Message Processing

When a BMS wishes to submit a Bulk or Batch Request, the BMS will need to:

1. Construct the corresponding type of request in conformance with the specification in section 4 ‘Message Packaging’ below;
2. Send the request to the Bulk and Batch Push Endpoint;
3. Receive a receipt from SBR ebMS3 and then continue with "other" BMS processing (i.e. not wait for the business response before continuing with BMS processing);
4. Poll for and Pull :
   * A combined Validation Result and Business Response (in the case of an Intermediate response time service level) or
   * A Validation Result followed by a combined Validation Result and Business Response (in case of a Delayed response time service level). i.e. Poll for and Pull two responses.
5. Wait for at least the specified corresponding SLA response time (as determined by the service-action being invoked and the contents of the request message business payload) then begin polling for the response message by sending a pull request to the Batch and Bulk Pull Endpoint. This pull request will be a "selective" pull request with the selection criteria being the message ID for the corresponding Push request message. The RefToMessageId field that is contained in the SignalMessage/eb:PullRequest node will be populated in the Pull request for this purpose (note this is not the same as the RefToMessageId that is under SignalMessage/MessageInfo node);
6. In the event that the pull request is unsuccessful, receive an error signal response with error code EBMS:0006 and description “EmptyMessagePartitionChannel” and then send another pull request in compliance with the intervals in the guidelines in section 2.4.5 ‘Polling Interval’ below;
7. Subject to the guidelines in section 2.4.5 ‘Polling Interval’, repeat step ii until such time as a pull request is successful (i.e. there is a response that is able to be pulled from SBR ebMS3).

An example of this sequence of interactions is illustrated in the following diagram:



*Figure 5 : Service Invocation Types – Intermediate Batch / Bulk*



*Figure 6 : Service Invocation Types – Delayed Batch or Bulk*

### Collect

When a BMS wishes to collect information or communications made available by an Australian Government Agency such as the ATO the BMS will need to:

1. Construct a Single Collect Request in conformance with the specification in section 4 ‘Message Packaging’ below;
2. Send the request to the Collect Endpoint;
3. Receive a receipt from SBR ebMS3 and then continue with "other" BMS processing (i.e. not wait for the business response before continuing with BMS processing);
4. Wait for at least the specified SLA response time for the service-action being invoked then begin polling for the response message by sending a pull request to the Collect Endpoint;
5. In the event that a pull request is unsuccessful, receive an error signal response with error code EBMS:0006 and description “EmptyMessagePartitionChannel” and then send another pull request (repeat from step d) in compliance with the intervals in the guidelines in section 2.4.5 ‘Polling Interval’ below;
6. In the event that a pull request is successful, but a payload is returned indicating that that there is nothing to collect, then send another Single Collect Pull request (repeat from step a) in compliance with the intervals in the guidelines in section 2.4.5 ‘Polling Interval’ below;
7. In the event that a pull request is successful and the response contains the communication item that was awaiting collect, then this indicates a successful completion of the collect transaction.

This sequence of interactions is illustrated in the following diagram:



*Figure 7 : Service Invocation Types – Collect Intermediate*

### Polling Interval

For asynchronous requests the BMS SHALL poll for the response after a specific time interval. The agency Implementation Guides outline the polling pattern for various asynchronous exchanges exposed by these agencies. The purpose for these directives is to police polling intervals via appropriate guidelines to ensure service is not overloaded by requests. Polling interval only applies to asynchronous interactions as for synchronous interactions the BMS will halt the thread and wait for the response.

## Web Service Standards

SBR ebMS3 web services shall conform to the key standards specified in the table below.

| Category | Applicable Recommendations |
| --- | --- |
| Transport | HTTP 1.1  HTTP over TLS |
| Messaging | ebMS3   * AS4 Conformance Profile   *OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features, OASIS Standard, 1 October 2007OASIS ebXML Messaging Services Version 3.0: Part 2, Advanced Features. 19 May 2011. OASIS Committee Specification 01*   * Advanced ebMS3 Features   *AS4 Profile of ebMS 3.0 Version 1.0, OASIS Standard, 23 January 2013*  SOAP 1.2  SOAP Messages with Attachments |
| Description | Namespaces in XML [World Wide Web Consortium 14-January-1999]  XML 1.0 (Second Edition)  XML Schema Part 1: Structures  XML Schema Part 2: Datatypes |
| Security | TLS 1.0  HTTP over TLS  RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile  WS-Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification, 1 February 2006  Web Services Security: X.509 Certificate Token Profile 1.1 OASIS Standard Specification, 1 February 2006  Web Services Security: SAML Token Profile 1.1 OASIS Standard Specification, 1 February 2006  XML Encryption Syntax and Processing  XML Signature Syntax and Processing |

*Table 8: Recommendations used by SBR*

## Common Characteristics

All the SBR ebMS3 web services have the following common characteristics:

* All requests SHALL be initiated by business management software.
* A single security approach

All requests SHALL be secured in the same way across all services. Where security related information is returned in a response, a standard approach is also employed.

* A single mechanism to indicate the success of a request.

All responses MUST employ a standard mechanism to indicate the success of a request.

If the request is successful, any business level response data MUST be returned, optionally with attachments.

* A standardised approach to the indication of failure conditions

Failures related to the transport of ebMS3 messages SHOULD be indicated using ebMS3 Faults, while business level errors as well as the authentication errors MUST be represented in a standard format within the response message (see section 5 ‘Error Management’ for more details). If the request is unsuccessful, any business information, for example documents or attachments, in the response SHOULD be ignored.

* Agency agnostic end-points

As suggested by Figure 2 ‘High Level Overview of the SBR ebMS3 Solution’, SBR provides endpoints which are not bound to any agency.

A standard field within the message structure (PartyInfo/To/PartyID as discussed in section 3.3.2 ‘eb:UserMessage/eb:PartyInfo’) is used to determine to which agency a request MUST be forwarded. Beyond this field, business management software does not require any knowledge of the mechanisms needed to communicate with the agencies involved in SBR. New agencies MAY be added to SBR with some level of client testing to retest the ebMS3 infrastructure of the business management software.

* Request sizes are limited

Business documents have been designed to ensure that requests to agencies are constrained within size limits. Such limits are described in the Message Implementation Guide (refer to section 2.7 ‘Message Implementation Guides’).

* Message Signing

All request messages coming into SBR ebMS3 SHALL be signed with the sender’s AUSKey as per the ebMS3 standard.

Only one-way pull response messages going out from SBR ebMS3 as a result of Bulk and Batch Pull request will be signed using the relevant agency’s key. All other responses will remain unsigned..

* Message Compression

All request messages coming into SBR ebMS3 SHALL employ ebMS3 compression.

Response messages going out from SBR ebMS3 for of Batch/Bulk Pull and Collect transmissions will employ ebMS3 compression. Whereas, response messages for Single-sync and Single-async will not employ ebMS3 compression.

## Message Implementation Guides

The ebMS3 services provide the building blocks from which the more complex collaborations needed to fulfil an SBR reporting obligation are built.

The way in which services are choreographed to create a composite service to fulfil a SBR reporting obligation is described within a Message Implementation Guides (MIG).

For an agency, there may be a Implementation Guide. The Implementation Guide contains message implementation details that are common across all messages for the agency. It also describes the rules and guidelines for the SBR platform that are common across all collaborations for the agency.

In addition to the agency Implementation Guide, there is a MIG for each reporting obligation (e.g. Company Tax Return) and its primary purpose is to describe:

* The business interactions required to fulfil the reporting obligation which in turn drives the required service choreography.
* The structure, content, rules and response messages of the obligation specific request and response message payloads. The payload content is also defined, in a machine readable format, by the relevant taxonomies which are referenced within the MIG. The MIG and the taxonomies MUST thus be jointly consulted in order to gain a complete understanding when implementing a business interaction.
* The interaction specific values needed for a small set of standard fields within the ebMS3 service message structure.
* Any specific use of optional fields within the service infrastructure.

MIG documents for existing SBR Core Services have been modified to include SBR ebMS3 information so that the same MIG document can be used to guide the use of SBR Core Services and SBR ebMS3 services.

# Message Structure

All SBR ebMS3 requests/responses SHALL employ the common ebMS3 message format. All messages SHALL be carried over a one way HTTPS transport, and employ the ebMS3 envelope structure. SOAP messages MUST employ UTF-8 or UTF-16 character encodings.

The SBR ebMS3 message format is a SOAP message that contains standard SOAP header node encapsulating the additional ebMS headers qualified with the ebMS namespace. The message is packaged as a SOAP 1.2 message independent from communications protocols. This envelope MUST be structured in compliance with the SOAP Messages with Attachments [1] W3C Note, referred to as a Message Package.

There are two logical sections within the Message Package:

* The first section is the ebMS Header (i.e. the eb:Messaging header block), itself contained in the SOAP Header.
* The second section is the ebMS Payload, which itself comprises two sections:

1. the SOAP Body element within the SOAP Envelope, and,
2. zero or more additional MIME parts containing additional application-level payloads.  
   The SOAP Body and MIME parts are also referred to as ebMS Payload Containers.

Two types of ebMS messages are supported by SBR:

1. **Signal message:** Message containing a signal, used for delivery receipts, errors and to initiate one way pull. Figure 6 ‘ebMS Signal Message Structure’ shows the structure of ebMS signal message.
2. **User message:** Message containing request/response payload. Figure 7 ‘ebMS User Message Structure’ shows the structure of ebMS user message.

In the ebMS SOAP header eb:Messaging, the prefix "eb" is an example prefix that corresponds to the ebMS 3.0 namespace, as defined in Section 1.8 ‘Namespaces’.



*Figure 6: ebMS Signal Message Structure*



*Figure 7: ebMS User Message Structure*

## Security Header

As SBR has adopted the Web Service Security 1.1 recommendation, all security related information MUST be carried in one or more wsse:Security elements within the SOAP header. Section 6 ‘Security’ provides a detailed description of the security requirements and implementation.

## ebMS Header

The ebMS header is the eb:Messaging element in Figure 6 ‘ebMS Signal Message Structure’ and Figure 7 ‘ebMS User Message Structure’. This element has the following children elements:

| Name | Description | Optionality |
| --- | --- | --- |
| eb:UserMessage | Contains all header information for a User message. If this element is not present, an element describing a Signal message MUST be present. | Optional |
| eb:SignalMessage | Contains all header information for the Signal message. If this element is not present, an element describing a User message MUST be present. | Optional |

*Table 9: ebMS Header Child Elements*

The ebMS header also has following mandatory attribute:

| Name | Description |
| --- | --- |
| @S12:mustUnderstand | Indicates whether the contents of the element MUST be understood by the ebMS handler. This attribute is REQUIRED, with namespace qualified to the SOAP namespace (http://schemas.xmlsoap.org/soap/envelope/). It MUST have value of '1' indicating the element MUST be understood or rejected. |

*Table 10: ebMS Header Attributes*

### eb:UserMessage

The eb:UserMessage element has following children elements:

| Name | Description | Optionality |
| --- | --- | --- |
| eb:MessageInfo | Contains data that identifies the message, and relates to other messages' identifiers. | Required |
| eb:PartyInfo | Contains data about originating party and destination party. | Required |
| eb:CollaborationInfo | Contains elements that facilitate collaboration between parties. | Required |
| eb:MessageProperties | Contains message properties that are user-specific. As parts of the header such properties allow for more efficient monitoring, correlating, dispatching and validating functions which would otherwise require payload access. | Optional |
| eb:PayloadInfo | Identifies payload data associated with the message, whether included as part of the message as payload document(s) contained in a Payload Container, or remote resources accessible via a URL. The purpose of the PayloadInfo is:  (a) Make it easier to directly extract a particular payload associated with this User message,  (b) Allow an application to determine whether it can process the payload without having to parse it. | Optional |

*Table 11: eb:UserMessage Structure*

The eb:UserMessage has following attribute:

| Name | Description | Optionality |
| --- | --- | --- |
| @ mpc | This attribute contains a URI that identifies the Message Partition Channel to which the message is assigned. The absence of this element indicates the use of the default MPC. When the message is pulled, the value of this attribute MUST indicate the MPC requested in the PullRequest message, unless the default MPC is being used. | Optional |

*Table 12: eb:UserMessage attributes*

### eb:SignalMessage

The eb:SignalMessage element has following children elements:

| Name | Description | Optionality |
| --- | --- | --- |
| eb:MessageInfo | Contains data that identifies the message, and relates to other messages' identifiers. | Required |
| eb:[SignalName] | Defines the nature of the ebMS signal. | Required |

*Table 13: ebSignalMessage Structure*

## UserMessage SBR ebMS3 Profile

This section provides further details about what is needed in the ebMS3 header of a user message for SBR ebMS3 interactions. The values for the fields and any rules that may be involved in setting them are defined in agency specific MIGs.

### eb:UserMessage/eb:MessageInfo

Table 14 below shows the children elements for eb:MessageInfo, and their use within requests and responses.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:Timestamp | Date at which the message header was created | Client MSH to set with date:time when the request message is sent to SBR ebMS3 | Client MSH to set with date:time when the request message is sent to SBR ebMS3 | SBR ebMS3 to set with date:time when the response message is sent to the BMS | Required |
| eb:MessageId | Globally unique identifier conforming to MessageId [RFC2822].  The value should not exceed 255 characters. | Set by BMS | Set by BMS | Set by SBR ebMS3 | Required |
| eb:RefToMessageId | MessageId value of an ebMS Message to which this message relates. | N/A | N/A | Set by SBR ebMS3 to copy of MessageInfo.MessageId field from related Request Message | Optional |

*Table 14: eb:MessageInfo Structure*

### eb:UserMessage/eb:PartyInfo

Table 15 below shows the children elements for eb:PartyInfo, and their use within requests and responses.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:From | Contains information describing the originating party | BMS to set with Business Details | BMS to set with Business Details | Agency to set with its details | Required |
| eb:To | Contains information describing the destination party. | BMS to set with Agency e.g. ATO details | BMS to set with Agency e.g. ATO details | Agency e.g. ATO to set with business details from related Request Message | Required |

*Table 15: eb:PartyInfo Structure*

#### **eb:PartyInfo/eb:From**

**Table 16** below shows the children elements for eb:From, and their use within requests and responses.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:PartyId | String value that identifies a party. | Set by BMS as per the rules defined in relevant agency’s Implementation Guide. | Same as single Request | Set by Agency to Agency identifier e.g. agency's ABN | Required |
| eb:PartyId@type | The type attribute indicates the domain of names to which the string in the content of the PartyId element belongs. | BMS to set as per the rules defined in relevant agency’s Implementation Guide. | Same as Single Request | Set by Agency e.g. when the PartyID is agency’s ABN this will be set to: “http://abr.gov.au/PartyIdType/ABN” | Required |
| eb:Role | Identifies the authorized role of the Party sending the message. | Set by BMS to URI representing authorised roles as per the rules defined in relevant agency’s Implementation Guide. | Same as Single Request | Set by Agency to URI representing the role of the Agency | Required |

*Table 16: eb:From Structure*

#### **eb:PartyInfo/eb:To**

Table 17 below shows the children elements for eb:To, and their use within requests and responses.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:PartyId | String value that identifies a party. | Set by BMS to Agency’s Identifier e.g. agency's ABN | Same as single Request | Set by agency to the copy of PartyInfo.From.Party Id of related Request Message (either Two-Way/ Sync Request UserMessage or One-Way/Push Request UserMessage ) | Required |
| eb:PartyId@type | The type attribute indicates the domain of names to which the string in the content of the PartyId element belongs. | Set by BMS to the type of agency’s identifier e.g. in case of Agency’s ABN the type should be “urn:oasis:tc:ebcore:partyid-type:ABN:0151”. | Same as Single Request | Set by agency to the copy of PartyInfo.From.Role of related Request Message (either Two-Way/ Sync Request UserMessage or One-Way/Push Request UserMessage ) | Required |
| eb:Role | Identifies the authorized role of the Party receiving the message. | Set by BMS to URI representing the role of the agency. See relevant agency’s Implementation Guide for more details. | Same as Single Request | Set by agency to the copy of PartyInfo.From.Role of related Request Message (either Two-Way/ Sync Request UserMessage or One-Way/Push Request UserMessage ) | Required |

*Table 17: eb:To Structure*

### eb:UserMessage/eb:CollaborationInfo

Table 18 below shows the children elements for eb:CollaborationInfo, and their use within requests and responses.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb: AgreementRef | String element that identifies the entity or artefact governing the exchange of messages between the parties | Set by BMS to URI for pMode file that relates to this MEP. pMode URI’s are specified in Appendix B. | Same as single request | Set by agency to the corresponding value in the request | Required |
| eb: AgreementRef/@pmode | Allows for explicit association of a message with a P-Mode | Set by BMS to the PMode.Id element from the PMode used for this transaction. | Same as single request | Set by agency to the same value as in the request | Required |
| eb:Service | String identifying the service that acts on the message. | Set by BMS to Service (Form e.g. CTR or Business Object) Name expressed as a URI using the following format:  <http://sbr.gov.au/agency/service/CCYY>  e.g. In case of CTR offered by ATO the service name should be  <http://sbr.gov.au/ato/ctr/2015> | Same as single request | Set by agency to the same value as in request | Required |
| eb:Service@type | Indicates how the parties sending and receiving the message will interpret the value of the element. | Not to be used | Not to be used | Not to be used | Optional |
| eb:Action | String element that identifies an operation or an activity within a Service. | Set by BMS to one of [Add, Update, List, Get, Submit, Validate] concatenated with "VVV.vv" where VVV.vv is the version number of the Agency interaction (Service-Action) that is sought to be invoked by the BMS and:  VVV = major version number and  vv = minor version number  format: ActionName.VVV.vv  example: List.002.01 | Same as single request | Set by agency to one of [List, Get, Validate, Submit, Lodge, Pre-Lodge] concatenated with "VVV.vv" where VVV.vvv is the version number of the Agency interaction (Service-Action) that this response is for. | Required |
| eb:ConversationId | String element that identifies the set of related messages that make up a conversation between Parties. | Set to value generated by BMS by concatenation of:  PartyInfo.From.PartyId  AND  MessageInfo.Timestamp  AND  Sequence Number Unique within the User organisation submitting the request identified by the PartyId | Same as Single Request | Set by agency to value of CollaborationInfo.ConversationId from corresponding request User Message | Required |

*Table 18: eb:CollaborationInfo Structure*

### eb:UserMessage/eb:MessageProperties

eb:MessageProperties element is user defined and contains on or more eb:Property elements. Table 19 lists out the custom eb:Property elements used by SBR ebMS3. Please note that eb:Property@name values are case sensitive, also all required message properties must be provided otherwise the message will be rejected by SBR ebMS3 platform.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:Property@name | An ID assigned by SBR or relevant agency to the BMS product | Set by BMS to ‘ProductID’ | Set by BMS to ‘ProductID’ | N/A | Required |
| eb:Property | Unique Product ID | Set by BMS to [Unique identifier assigned by SBR/Agency for the BMS product] | Same as Single Request | N/A | Required |
| eb:Property@name | BMS Vendor name | Set by BMS to ‘BMS Vendor’ | Set by BMS to ‘BMS Vendor’ | N/A | Required |
| eb:Property | BMS Vendor Name | Set by BMS to [Name of entity who created the BMS product] | Same as Single Request | N/A | Required |
| eb:Property@name | BMS Product Name | Set by BMS to ‘BMS Name’ | Set by BMS to ‘BMS Name’ | N/A | Required |
| eb:Property | BMS Product Name | Set by BMS to [Name of BMS product] | Same as Single Request | N/A | Required |
| eb:Property@name | BMS Product Version | Set by BMS to ‘BMS Version’ | Set by BMS to ‘BMS Version’ | N/A | Required |
| eb:Property | BMS Product Version | Set by BMS to [Version Number of BMS product] | Same as Single Request | N/A | Required |

*Table 19: eb:MessageProperties Structure*

In addition to the above mentioned properties there may be some additional properties specific to the agency and/or the service being requested. These will be specified in the relevant agency Implementation Guide and form/service MIG respectively.

### eb:UserMessage/eb:PayloadInfo

This element identifies payload data associated with the message.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:PartInfo | The PartInfo element is used to reference a MIME attachment | One Per Attachment | One Per Attachment | One Per Attachment | Optional |

*Table 20: eb:PayloadInfo Structure*

#### **eb:PayloadInfo/eb:PartInfo**

Table 21 below shows the children elements for eb:PartInfo, and their use within requests and responses

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:PartInfo@href | Content-ID (CID) URI of the payload object referenced | Set by BMS to cid:{Attachment Filename}  {Attachment Filename} must be unique amongst the Payload Parts within the Message Package | Set by BMS to cid:{Attachment Filename}  {Attachment Filename} must be unique amongst the Payload Parts within the Message Package | Set by agency to cid:{Attachment Filename}  {Attachment Filename} must be unique amongst the Payload Parts within the Message Package | Required |
| eb:PartProperties | This element has zero or more eb:Property child elements. One for each separate "Property" that is to be used to process the Payload Part |  |  |  | Required |

*Table 21: eb:PayloadInfo Structure*

#### **eb:PayloadInfo/eb:PartInfo/eb:PartProperties**

This element has zero or more eb:Property child elements. One for each separate "Property" that is to be used to process the Payload Part. Table 22 below shows eb:Property elements and associated attribute values to be used.

| Name | Description | Single Request 2 Way Sync Push (Single/Collect) | Bulk/Batch 1 Way Push | Response 2 Way Sync 1 Way Pull  Pull (Single/Collect) | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:Property@name | Property Name | Set by BMS to ‘PartID’ | Set by BMS to ‘PartID’ | Set by agency to ‘PartID’ | Required |
| eb:Property | PartID | BMS to set to a unique identifier within this message for this payload part (which will allow the BMS to match a payload part specific response with the corresponding request payload part). | BMS to set to a unique identifier within this message for this payload part. | Agency to Copy the corresponding PartID value from the request message which will allow the BMS to match the response contained in this payload part with the corresponding request payload part. | Required |
| eb:Property@name | Property Name | Set by BMS to ‘DocumentName’ | Set by BMS to ‘DocumentName’ | Set by agency to ‘DocumentName’ | Required |
| eb:Property | DocumentName | Set by BMS to the "Business Name" of the type of document contained in the MIME Part.  e.g. in case of ATO as the relevant agency, CTR or "NIPSS" | Set by BMS to the "Business Name" of the type of document contained in the MIME Part.  e.g. in case of ATO as the relevant agency, CTR or "NIPSS" | Set by agency to the type of the document contained in the MIME Part.  e.g. in case of ATO as the relevant agency, CTR or Schedule A etc. | Required |
| eb:Property@name | Property Name | Set by BMS to ‘DocumentType’ | N/A | Refer to relevant agency’s Implementation Guide | Required |
| eb:Property | Document Type | Set by BMS to the type of attachment.  Please refer to Implementation Guide for the relevant agency to get the list of expected values for that agency. | N/A | See agency Implementation Guide for the list of expected values. | Required |
| eb:Property@name | Property Name | Set by BMS to ‘MimeType’ | N/A | Refer to relevant agency’s Implementation Guide | Required |
| eb:Property | MIME Type | Set by BMS to the type of MIME.  Please refer to Implementation Guide for the relevant agency to get the list of expected values for that agency. | N/A | See agency Implementation Guide for the list of expected values. | Required |
| eb:Property@name | Property Name | Set by BMS to ‘filename’ | N/A | Refer to relevant agency’s Implementation Guide | Required |
| eb:Property | File Name | Set by BMS to the Name of the file.  Please refer to Implementation Guide for the relevant agency to get the list of expected values for that agency. | N/A | See agency Implementation Guide for the list of expected values. | Required |

*Table 22: eb:PartProperties Structure*

In addition to the above mentioned part properties there may be some additional properties specific to the agency and/or the service being requested. These will be specified in the relevant agency Implementation Guide and form/service MIG respectively.

## SignalMessage SBR ebMS3 Profile

This section provides further details about what is needed in the ebMS3 header of a signal message for various SBR ebMS3 interactions.

### eb:SignalMessage/eb:MessageInfo

Table 23 below shows the children elements for eb:MessageInfo, and their use within different types of Signal requests

| Name | Description | Pull Request from BMS | Receipt from SBR ebMS3 | Error from SBR ebMS3 | Optionality |
| --- | --- | --- | --- | --- | --- |
| eb:Timestamp  Date/time element values SHALL be provided with time to the second as a minimum e.g.  2009-03-25T13:53:48Z  Date/time values in Message Timestamps (see section 3.3.3.2) SHOULD be provided to millisecond accuracy to assist in problem resolution and performance management. | Date at which the message header was created | BMS-embedded client to set with DateTimestamp for the date:time that the pull request is sent to ATO | SBR ebMS3 Platform to set with DateTimestamp for the date:time that the receipt message is sent to the BMS | SBR ebMS3 Platform to set with DateTimestamp for the date:time that the receipt message is sent to the BMS | Required |
| eb:MessageId | Globally unique identifier conforming to MessageId [RFC2822] | Set by ebMS3 Client | Set by SBR ebMS3 | Set by SBR ebMS3 | Required |
| eb:RefToMessageId | MessageId value of an ebMS Message to which this message relates | N/A | Copy of MessageInfo.MessageId field from related Request Message i.e. the MessageId of the message for which this signal is the receipt. | Copy of MessageInfo.MessageId field from related user Request Message i.e. the MessageId of the message to which this error relates. | Required if the Signal Message is Error or Receipt. |

*Table 23: ebMessageInfo Structure*

### eb:SignalMessage/eb:[SignalName]

This REQUIRED element defines the nature of the ebMS signal. There is only one eb:[SignalName] child element when [SignalName]=PullRequest or [SignalName]=Receipt. There may be several children elements when SignalName=Error. Structure of these signal messages is detailed below:

#### **SignalMessage/eb:PullRequest (Selective)**

SBR ebMS3 requires using selective pull for all asynchronous interactions i.e. Batch/Bulk Pull and the Pull leg in a Single-ASync-Chatty/Collect service invocation types.

| Name | Description | Pull Request from BMS |
| --- | --- | --- |
| @mpc | This OPTIONAL attribute identifies the Message Partition Channel from which the message is to be pulled. The absence of this attribute indicates the default MPC. | SBR EBMS3 uses only the default message partition channel so BMS doesn’t need to specify any value for this attribute. |
| eb: RefToMessageId | Messages from the targeted MPC (specified by the @mpc attribute above) will be pulled only if there PullRequest.RefToMessageId field matches this value. | Set by BMS to the copy of PushRequest.MessageId field from related Push Request Message. |

*Table 24: eb:PullRequest (selective)*

#### **eb:SignalMessage/eb:Error**

Table 26 below shows the children elements for eb:Error, one of these must be present for each error being reported.

| Name | Error from SBR ebMS3 | Optionality |
| --- | --- | --- |
| @origin | "ebMS", "reliability" or "security" | Optional |
| @category | Identifies the type of error related to a particular origin. For example: Content, Packaging, UnPackaging, Communication, and InternalProcess. | Optional |
| @errorcode | See APPENDIX A | Required |
| @severity | "warning" or "failure" | Required |
| @refToMessageInError | Copy of MessageInfo.MessageId field from related Request Message i.e. the MessageId of the message to which the error relates. | Optional |
| @shortDescription | Short readable description of error - See APPENDIX A | Optional |
| eb:Description | Detailed description of error - See APPENDIX A | Optional |
| eb:ErrorDetail | Additional details about the context in which the error occurred - See APPENDIX A | Optional |

*Table 25: eb:Error Structure*

#### **eb:SignalMessage/eb:Receipt**

The eb:Receipt contains a single child element ebbpsig:NonRepudiationInformation.

Table 27 below shows the children elements for eb:Receipt. NonRepudiationInformation, one of these must be present for each message part being receipted.

| Name | Receipt from SBR ebMS3 | Optionality |
| --- | --- | --- |
| eb:MesssagePartNRInformation.MessagePartIdentifier | Either this field or the reference URI to be populated.  Set to the identifier of the part contained in the related payload info in the related request message. | Optional |
| eb:MesssagePartNRInformation.Reference@URI | Either this field or the MessagePartIdentifier to be populated. | Optional |
| eb:MesssagePartNRInformation.Reference.DigestMethod | Message signing information | Required |
| eb:MesssagePartNRInformation.Reference.DigestValue | Message signing information | Required |

*Table 26: eb:Receipt Structure*

## Dates and Times

All dates and times MUST be expressed in messages as per the standard XSD built-in "datetime" data type, as specified in http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime which is a subset of the ISO-8601 standard. Timestamps MUST be expressed in UTC (also called Zulu) time.

Date/time element values SHALL be provided with time to the second as a minimum e.g.

2009-03-25T13:53:48Z

Date/time values in Message Timestamps (see 3.4.1 eb:SignalMessage/eb:Message Info) SHOULD be provided to millisecond accuracy to assist in problem resolution and performance management.

It should be noted that where date/time values are displayed to users, they SHOULD first be converted into the local time zone.

## Timeout Values

There are a number of sources of delay between the issuing of a request by client software, and the delivery to the software of the resulting response. These include transmission delays between the client software and SBR ebMS3 and between SBR ebMS3 and the relevant agency, as well as processing delays at SBR ebMS3 and the Agency.

In order to ensure requests have the maximum chance of being successfully processed while allowing for the variable nature of the delays involved, a staggered approach to timeout values has been adopted.

### Single - Sync - Chatty

While most of these requests are expected to be processed within the order of 10 seconds, SBR ebMS3 SHALL allow up to 5 minutes for a request to be processed by an agency.

Developers SHOULD thus configure their products to use a timeout of somewhat more than 5 minutes, depending on the transmission delays to SBR ebMS3 expected as a result of the capacity of their client's internet connections and the nature of the obligations being supported.

### Asynchronous Interactions

For asynchronous requests the timeout interval should be as per the guidelines in Section 2.4.5 (Polling Interval)

Any timing considerations unique to a particular obligation SHALL be documented in the relevant agency service MIG.

## Payload Format

SBR ebMS3 is agnostic to message payload. The payload for most requests should mostly be in XBRL format; however there can be some agency/service exceptions which should be listed under agency Implementation Guide and/or service MIG.

# Message Packaging

As discussed in section 3 ‘Message Structure’, all messages to and from SBR MUST be packaged using the SOAP messages with Attachments (SwA) standard. This section outlines the SwA message packaging format for various message exchange patterns (as outlined in Section 2 ‘SBR ebMS3 Services’) supported by SBR.

## Single Request

Both synchronous and asynchronous single requests are requests where a single transaction is being requested for a specific service action e.g. ListAS, ValidateFBT, LodgeCTR and etc.

The Collect-Async requests may be packaged with an empty payload or may contain a payload with some retrieval parameters depending on the agency’s implementation a service.

Refer to agency specific MIGs for a detailed specification on how the Single Request messages are packaged.

## Single Receipt & Batch/Bulk Receipt

For asynchronous single requests (e.g. Single Push) and Batch/Bulk requests, SBR will send back a generic technical receipt after successful authentication of the incoming request. This receipt will comprise a single MIME part having an ebMS3 signal message with empty SOAP body. The ebMS3 header will be structured as per the guidelines in section 3.2 ‘ebMS Header’.



*Figure 8: Single and Batch/Bulk Receipt Message Packaging*

## Single Pull Request

The single pull requests SHALL comprise of a single MIME part having an ebMS3 signal message with empty SOAP body. The ebMS3 header in the signal message SHALL be structured as per the guidelines in section 3.4.2.1 ‘SignalMessage/eb:PullRequest’, with eb:RefToMessageId element being the criteria for selective pulling.



*Figure 9: Single Pull Request Message Packaging*

## Single Response

Refer to agency specific MIGs for a detailed specification on how the Single Response messages are packaged.

## Batch /Bulk Request

Refer to agency specific MIGs for a detailed specification on how the Bulk Request messages are packaged and if applicable, also how the XBRL documents are packaged.

## Batch/Bulk Pull Request (Selective Pull)

The Batch/Bulk pull requests SHALL be structured similar to a single pull request as per guidelines in section 4.3.

## Batch/Bulk Response

Refer to agency specific MIGs for a detailed specification on how the Batch/Bulk Response message is packaged.

# Error Management

#### OVERVIEW

#### This section details the approach to be taken to the handling of errors and exception conditions associated with the submission of requests to agencies via SBR ebMS3 and agency message processing channels. It divides the conditions that MUST be addressed into four broad areas;

1. User Errors
2. Client Software Errors
3. Transport Exceptions
4. Business Events

The nature of user and client software errors is such that their detection and remediation is largely the province of client software providers, an example being the failure of a user to turn on their internet connection. Thus, this document is largely silent on these types of errors, but where possible, information is provided as part of interactions with SBR ebMS3 to assist in this process.

On the other hand, this section provides detailed information on the reporting and management of errors in the latter two areas.

In a nutshell, Transport Exceptions and Business Events SHALL both be handled via event message block structure and returned to the client as an event message blocks (Overall Event Message Block and Business Event Message Block), contained within a MIME part. The only exception to this is for the errors occurring within the ebMS MSH component, which SHALL be handled using its internal framework, specified by the ebMS Core v3. (Ref3) and returned to the client as an ebMS Error Signal message.

#### CONTEXT

Figure11 ‘Sources of Error’ provides a high level overview of the possible sources of errors associated with the process of a business user employing their client software to submit reports to agencies via SBR ebMS3 and agency message processing channels.

Errors may result from a number of sources, each identified by a coloured triangle.

### User errors (shown in green)

* Triangle 1 : the user has not obtained an SBR credentials
* Triangle 2 : the user has not correctly set-up internal configurations like user not connected to Internet
* Triangle 3 : the user has not undertaken the appropriate registration or  
  authorisation procedures required by the agency or the software used to generate the request is not registered with SBR ebMS3
* Triangle 4 : the information content (provided by the user) of the payloads is not correct. This error can result from either:
  + - * agency messaging channel processing i.e. “channel validation” (a request received by the SBR ebMS3 platform will be routed to the appropriate agency and that agency will have its own “messaging channel system” in which that request will undergo various types of processing which will include validating that a particular request payload has been correctly constructed in accordance with all the rules for a particular reporting form), or
      * From validation in the agency backend business system.

### Client software problems (shown in yellow)

* Triangle 5 : there are defects in the client software
* Triangles 6 : the messages generated by the client software do not conform to the required SBR ebMS3 standards
* Triangles 7 : the payload is not correctly formed. This error will result from agency “channel validation”.

### Unavailability of components (shown in blue)

* Triangle 8 : connectivity issues within the systems comprising SBR, or component unavailability due to scheduled maintenance
* Triangles 9 : agency processing systems are not available

### Internal technical (system) errors (shown in red)

* Triangles 10 : unexpected technical errors in SBR ebMS3 core components
* Triangles 11 : unexpected technical errors in agency message channel or its backend
* Triangles 12 : unexpected technical errors detected in external components



*Figure 11: Sources of Error*

#### High level Categorisation of Error Conditions

This document asserts that the various errors conditions described in Section 5.2 ‘Context’ can be placed into four high level areas, as shown in Figure 11 ‘Sources of Error’.

Two areas, client software or user errors, will be manifested via the various programming APIs used by the client software. Where these APIs are provided by SBR ebMS3, the associated documentation will indicate the nature of the possible errors. This category is outside the scope of this document.

The remaining conditions result from the client software interacting with SBR ebMS3, and are split into the remaining two high level areas.

Errors associated with the physical transport of messages from businesses to agencies fall into the “**Transport Exceptions**” category. This area covers any problems related to ensuring messages are successfully passed from business to government and back again. This document completely describes the architecture to handle transport exceptions, as well as enumerating the conditions which fall within this category.

Having established an error free flow of messages, all remaining error conditions will be as a result of business activities within the participating agency’s processing systems. The term business event is used since some of the conditions flagged at this level are of an informational or warning nature, and won’t necessarily represent an error condition per se.

#### Transport Exceptions

Transport Errors are errors occurring in SBR ebMS3 or Agency Messaging channels, which can be as a result of either technical system errors due to the unavailability or non-functioning of software components or business exceptions due to non-compliance of certain business rules or expectations enforced within SBR ebMS3 or agency messaging channels.

SBR ebMS3 gateway is the first place where majority of initial errors will be detected (e.g. authentication check failed, message integrity check failed, the specified agency, service and/or action values are incorrect, the software product is not whitelisted and so on). As soon as any of those errors are detected by SBR ebMS3 it will reject the incoming message by returning an error message to the user so the Agency will never get those client messages.

Then for the messages which passed SBR ebMS3 checks, the Agencies will perform channel validation and despatch for agency backend processing.

#### Transport Error Communication

Due to the composed nature of SBR ebMS3, which includes relatively independent sub-components like the ebMS MSH, Router and Agency specific adapters, Transport exceptions in SBR ebMS3 will be communicated in two different ways. All ebMS errors will be handled and reported as per ebMS specifications and other core sub-components will use the event structure (overall event message block) within the payload to communicate errors.

Agency specific components like the messaging channel will also use the overall event message block to communicate errors, merging any event metadata and intermediate results that may be available, incorporating everything into the response message.

Error Events generated by agencies SHALL pass unaltered through SBR ebMS3 and be provided as is to the client software, so it is expected that the agency components perform necessary harmonisation of error codes used when generating the event structure.

In summary, all SBR ebMS3 errors, including agency specific errors will be reported in one of the following ways:

* As an ebMS Signal Message or
* As an Overall Event Message Block within the response message envelope
* As a SOAP fault
* As a HTTP Error Response

#### SOAP Fault/HTTP Error Response

Errors detected in the ebMS3 Message Service Handler before ebMS processing will be reported as a SOAP Fault or a HTTP Error Response. Example of such an error condition is when a wrong SOAP Envelope or HTTP header is detected in the request message.

SOAP Fault / HTTP error response can also be returned as a result of unexpected technical errors detected in the Security Infrastructure (Triangle 12 in Figure 11 above).

#### ebMS Error Signal Message

These types of error responses are usually returned as a result of ebMS processing errors occurring within the MSH, handled out-of-the-box by MSH’s error handling framework as per ebMS specifications. These will be returned to the client as-is.

A good majority of the error response from SBR ebMS3 will be of this type and as per ebMS specification will be packaged as an ebMS Signal Message and sent back to the client as a response. The Signal message will contain at least one *eb:Error* element and will always be combined with a SOAP Fault (as per SOAP12 specification).

**Packaging as ebMS Error Message**

ebMS error is represented by an *eb:Error* XML infoset and will be packaged as ebMS Signal Message with an appropriate error code and description and sent back to the client as a response. The error message will include the following elements:

* **Error origin** - Module where the error occurred.
* **Error category** - Identifies the type of error that is related to a particular origin.
* **Error code** - Unique identifier for the error.
* **Severity** - Indicates the severity of the error.
* **Error description** - Provides a description of the error.

The following is an example of an Error Signal Message:

<?xml version="1.0" encoding="utf-8"?>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"><soapenv:Header>

<eb:Messaging xmlns:eb="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" soapenv:mustUnderstand="true">

<ns2:SignalMessage xmlns:ns2="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/">

<ns2:MessageInfo>

<ns2:Timestamp>2014-05-16T07:49:38.929Z</ns2:Timestamp>

<ns2:MessageId>1751669608.238.1400227014921.JavaMail.acqqw@SYD02H131GV</ns2:MessageId>

</ns2:MessageInfo>

<ns2:Error category="Communication" refToMessageInError="1143653756.0.1396402978054.JavaMail.ucp2w@ATODE-EDG7CAWHJ" errorCode="EBMS:0006" origin="EBMS" severity="Warning" shortDescription="EmptyMessagePartitionChannel">

<ns2:Description lang="en">There is no message available for pulling from this MPC at this moment.</ns2:Description>

<ns2:ErrorDetail>DeliveryFailureException: Pulling message from pull destination failed</ns2:ErrorDetail>

</ns2:Error>

</ns2:SignalMessage>

</eb:Messaging>

</soapenv:Header>

<soapenv:Body>

<soapenv:Fault xmlns:axis2ns27="http://www.w3.org/2003/05/soap-envelope">

<soapenv:Code>

<soapenv:Value>axis2ns27:EBMSInternalError</soapenv:Value>

</soapenv:Code>

<soapenv:Reason>

<soapenv:Text xml:lang="en">There is no message available for pulling from this MPC at this moment.</soapenv:Text>

</soapenv:Reason>

<soapenv:Detail>

<DetailError>DeliveryResponseException: Pulling message from pull destination failed</DetailError>

</soapenv:Detail>

</soapenv:Fault>

</soapenv:Body>

</soapenv:Envelope>

*Figure 12: Error Signal Message*

Section 3.4.2.3 ‘eb:SignalMessage/eb:Error’ in this document shows the structure of eb:Error and its children elements. For more information about the ebMS error format, see the Error handling chapter in the ebMS Core 3.0 specification ([link](http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html)). The standard ebMS errors is excerpted from the ebMS core spec and included as **Appendix A: ebMS Errors** below for quick reference.

#### Overall Event Message Block to Report Transport Exceptions

The event structure leverages off the event message block schema (see Figure 20: Overall Event Message Block Schema ) that is defined as part of the SuperStream messaging standards and used to report and communicate business events in SBR ebMS3. Section 5.6 ‘Message Events’ below provides a detailed explanation of how this structure should be used.

SBR ebMS3 platform further leverages this event structure to communicate Transport errors to the client software.

Transport exception communicated using the Event Message Block will set the**MaximumSeverity.Code** to “Error”, both indicating that the overall transaction is in error and the message could not be successfully processed by SBR ebMS3 platform.

“Error” carries with it an expectation that the sender will change/correct the request (Sender side problem).

#### Transport Exception Conditions

Figure 13 ‘SBR ebMS Error Conditions’ below shows a range of conditions that may occur while exchanging messages with the SBR ebMS3 platform. To provide certainty as to the action to be taken in regard to an error they are grouped into three categories – Client side errors, Platform Unavailability and Platform internal errors. Errors in outbound processing should be considered as an extension to internal Platform Internal errors. At the highest level, these categories identify the distinct actions needed to be taken on receipt of the Error Response.



*Figure 13: SBR ebMS Error Conditions*

#### Client Side Errors

All errors in this category result from a defect or non-compliance within the software used by business causing the generation of invalid requests. It is intended that the SBR ebMS3 testing regime will allow the detection and rectification of this category of errors. However the possibility still exists that such conditions will occur at runtime, perhaps triggered by edge cases in data contents.

The bulk of these errors will be detected and reported by SBR ebMS3, with the remainder being detected by agencies.

As explained in previous sections, errors detected within the ebMS MSH component will be handled by its internal framework specified by ebMS Core 3.0 specification. Refer to Error Handling chapter (chapter 6) in specification for a list of errors that belong to this category.

Appendix A: ‘ebMS Errors’ is an excerpt from the specification that included in this document for a quick reference. The table describes the ebMS Errors that may occur within the ebMS, security and processing modules of the service side MSH.

One of the most common client side errors returned by SBR ebMS3 MSH is the failure to find a matching exchange profile. This is a client side error because the exchange profile lookup criteria must be correctly supplied in the ebMS3 header.

|  |  |  |
| --- | --- | --- |
| Error Code | Short Description | Long Description |
| EBMS:0010 | The ebMS header or another header (e.g. reliability, security) expected by the MSH is not compatible with the expected content, based on the associated P-Mode. | Exchange Profile not found for Criteria: OwnerOrgId: {eb:To/PartyId}; PartnerOrgId: {eb:From/PartyId}; Service: {eb:Service}; ReceiverId: {MSH Internal Receiver Id}; OwnerBusinessIdType: {eb:To/PartyId@type}; PartnerBusinessIdType: {eb:From/PartyId@type}; ServiceType: {eb:Service@type} |

Note, Errors detected in the MSH before ebMS processing will be reported as a SOAP Fault or a HTTP Error Response. Example of such an error condition is when a wrong SOAP Envelope or HTTP header detected in the request message.

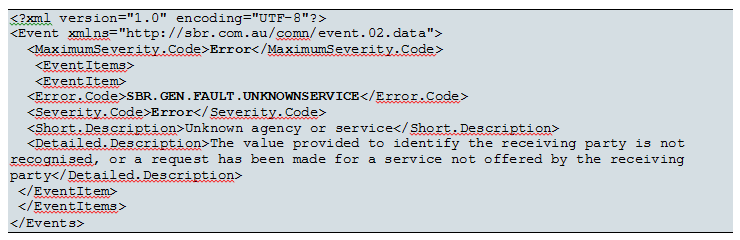
When client software errors are reported using the event structure then the ‘*EventItem’* element that captures the error information will have its ***Severity.Code*** set to ‘***Error’*** and a specific ***Error.Code*** will be assigned based on the current SBR ebMS3 platform rules. The **MaximumSeverity.Code** of the overall event message block will also be set to ‘**Error’** indicating the overall outcome of the transaction was a failure.

The codes below SHALL be used to indicate the detection of client software errors.

| Error Code | Short Description | Long Description |
| --- | --- | --- |
| SBR.GEN.FAULT.UNKNOWNSERVICE | Unknown agency or service | The value provided to identify the receiving party is not recognised, or a request has been made for a service not offered by the receiving party. |
| SBR.GEN.FAULT.UNKNOWNACTION | Unknown Action | The operation requested against the service is not offered by the receiving party. |
| SBR.GEN.FAULT.SOFTWARENOTREGISTERED | The software used to generate this request has not been registered with SBR | SBR ebMS3 has the capability to reject requests from client software that has not been self-certified by its developer organisation.  This fault is generated in the situation where this capability is enabled. |
| SBR.GEN.GEN.2 | Not authorised to lodge this report | Your SBR ebMS3 credential does not allow you to lodge a {report} with {agency}. Please contact {agency} to obtain a valid SBR ebMS3 credential and then re-submit. |
| SBR.GEN.GEN.3 | Intermediaries not accepted for this service | As an intermediary, you are not authorised to use this {*Report/Form*} with {*agency*}. Please arrange for a person with the correct SBR ebMS3 credential to resubmit. |

*Table 27: Client software errors*

The message excerpt below shows examples of an Overall Event Message that will be generated as a result of client software errors.



*Figure 14: Error indicating an Unknown Service detected in SBR ebMS3*

#### Platform Unavailability

A reality of the SBR ebMS3 solution, given the number of parties, components and business processes involved, is that at times, portions of the overall system may be unavailable. It is important that client software is aware of the “normality” of this error condition and takes the necessary steps to resubmit the request at a later time. This may involve automatic queuing of the request for resubmission at a later time, or notification to the user that they should initiate the resubmission after a suitable delay. Where resubmission is automated, it is recommended that an increasing delay be added between resubmission attempts.

In some cases, the time at which the service will be available again is known. In this case, the **Short.Description** text SHOULD contain the date and time after which the service is expected to be available again.

It is important that the indications provided to business users ensure they understand there is no need to contact either their software provider or the agency to which the request is being submitted. It is also important to realise that, given the independence of agency operations, it SHOULD NOT be assumed that because one interaction with one agency fails that all interactions with all agencies will fail. Software developers SHOULD adopt an optimistic approach to request submission, taking into account any information provided in regards to the date and time at which the service will be available again.

A specific Error.Code will be assigned based on the current platform rules to clearly indicate system unavailability.

| Error Code | Reason | Description/comment |
| --- | --- | --- |
| SBR.GEN.FAULT.CANTCONNECTTOAGENCY | A connection could not be established to an agency | SBR ebMS3 was not able to initiate a connection to the agency. Please try again later |
| SBR.GEN.FAULT.AGENCYNOTRESPONDING | The connection with the agency timed out | SBR ebMS3was able to connect to the agency, but did not receive a response to the submitted request. Please try again later |

*Table 28: Agency errors*

|  |
| --- |
|  |

*Figure 16: Error indicating an Agency processing systems is unavailable (scheduled outage)*

#### Platform Internal errors

As with the client software, SBR and agency systems may, at times, exhibit defects. While the testing regime SHOULD eliminate the bulk of these, it is nevertheless important that the SBR exception handling regime allow for their possibility at runtime.

As Figure 13 (SBR ebMS Error Conditions) shows, there is a broad range of errors that may occur internal to SBR. Errors that occur during response processing will also fall into this category.

The actions needed to be taken by business are the same – re-queue the request, and possibly contact the software provider or agency to ensure the failure is known to SBR. The latter action recognises that the error may be unique to the request, for example as a result of particular data included in the request.

Internal Errors detected within the ebMS MSH component will be handled by the framework specified by ebMS Core 3.0 specification. Refer to Error Handling chapter (chapter 6) in specification for a list of system errors that belong to this category (Ref3).

The Short.Description and Detailed.Description SHALL reflect the specific condition that resulted in the internal error. An equivalent Error.Code SHALL also be provided as per the table below:

| Error Code | Reason | Description/comment |
| --- | --- | --- |
| SBR.GEN.FAULT.GENERALERRORINCORE | An unhandled error occurred within SBR ebMS3 | This error is generated whenever an unhandled error is detected in SBR ebMS3. Record details and contact the agency service desk |
| SBR.GEN.FAULT.FAULTFROMAGENCY | A fault has been detected within the agency processing systems | A fault has been detected within the agency processing systems, which was not related to system availability. Record the details of the message and contact the agency service desk.  This event cannot be addressed independently by the end user or the software providers. The expectation is for the end user to provide the ebMS3 message ID or conversation ID for agency to investigate the transaction in their system. |

*Table 29: Platform errors*

Internal error events detected and generated by agencies SHALL pass unaltered through SBR ebMS3 and be provided as is to the client software.

The following Error.Code SHALL also be used to indicate Internal Agency Errors:

| Error Code | Reason | Description/comment |
| --- | --- | --- |
| SBR.GEN.FAULT.GENERALERRORINAGENCY | An unhandled error occurred within Agency | This error is generated whenever an unhandled error is detected in Agency |

*Table 30: Unhandled errors*

#### Business Event

Business Events SHALL be handled via the event message block structure. Please refer to relevant agency Implementation Guide for information related to Business Event processing.

The figure below shows an example of a Business Event, employing parameters to provide dynamic content and indicating the field in error in the input document (with an XBRL payload) via a location.

|  |
| --- |
|  |

*Figure 19: MessageEvent indicating an ABN in the input was invalid*

#### Message Events

In order that every message exchange has an explicit indication of its result, every response to a service request MUST include one ***Event*** within the Overall Event Message. An ***Event*** MUST include at least one ***EventItem***. Note that items MAY not necessarily be ordered by severity within a ***Event***.



*Figure 20: Message event schema structure*

#### Error Code

Every item SHALL carry a code to uniquely identify the condition that has occurred. In order to allow codes to be managed in a distributed fashion, codes SHALL take the following format:

**{Jurisdiction}.{Agency}.{Function}.{Id}**

Represented by the regular expression

**([A-Z0-9])+.([A-Z0-9])+.([A-Z0-9])+.([A-Z0-9])+**

Initially

**Jurisdiction** = SBR | CMN | QLD | NSW | ACT | VIC | SA | WA | NT | TAS

**Agency** = Jurisdiction specific agency code

For **CMN** (Commonwealth), = ATO, ASIC, APRA, ABS

For **SBR** = GEN (i.e. SBR wide codes)

For **States** = OSR (Offices of State Revenue)

**Function** = Agency specific functional area or GEN for agency wide codes

For **SBR** = GEN or FAULT

**Id** = function specific identifier (format may vary across agencies).

Examples are shown below;

**SBR.GEN.FAULT.TOOMANYINSTANCES**

**CMN.ATO.TFN.OK**

**QLD.OSR.PRL.000001**

The above structure recognises and caters for the current situation where agency errors are un-harmonised, and will need to be passed through to client software. This is not ideal, however as it implies the possibility of inconsistency in the messages business will receive for what are equivalent conditions in different agency reports.

Thus, the above scheme also caters, via the SBR jurisdiction, for efforts at harmonisation of error codes and messages.

In order to allow for the possibility of local councils being involved in SBR in the future, all agency code values commencing with “LCL” SHALL be reserved.

#### Severity Code

***EventItems*** can be categorised by ***Severity.Code*** with values of Error, Warning or Information (in descending order of importance).

The ***MaximumSeverity.Code*** field in an ***Event*** indicates the most severe level or error present in the associated ***EventItems***. However in the scenario a request has been partially rejected with some Errors present, the MaximumSeverity.Code will be Partial.

If an ***Event*** does not contain any ***EventItems*** with a ***Severity.Code*** of Error, then this indicates a successful request.

In the common situation of successful requests, the ***Event*** MAY contain a single ***EventItem*** with a ***Severity.Code*** of Information and a ***MaximumSeverity.Code*** of Information. The example below shows the minimum information that would need to be provided in this situation.

|  |
| --- |
|  |

*Figure 21: Minimal Event indicating success of the request*

Conversely, in order to indicate failure, the response MUST include an ***Event*** with at least one ***EventItem*** which has a ***Severity.Code*** of Error. This will result in a value of Error or Partial in the ***MaximumSeverity.Code*** field.

#### Descriptions

Descriptions on an item are intended to provide human readable text describing the error that has occurred. At present, descriptions are only provided in English.

Markup MAY be included within the description, but must be escaped, since descriptions are typed as strings (see the example below). Some platforms will automatically perform this escaping based on the values assigned to the field, converting “<” to “&lt;” for example.

At this stage, the only vocabulary of markup SHALL be XHTML. In addition, the only construct that MUST be supported is hyperlinks, via <a href=”…”> </a> tags, to support the ability to refer business users to online resources.

Any unrecognised tags SHOULD be ignored and removed from the description. This will allow the graceful introduction of other tags as the need arises.

#### Short Descriptions

Each item SHOULD include a short description, which provides a concise description of the condition that has occurred. It is intended for use in visual components such as tool tips, and it is thus RECOMMENDED that it be no longer than 100 characters including any parameter values.

Given the suggested constraint on length, it is RECOMMENDED that markup only be used for inclusion of a hyperlink in the short description.

#### Long Descriptions

Where a more extensive explanation of a condition needs to be provided than that reasonably contained within the short description, the event item MAY include a detailed description.

An example of the use of the detailed description might be to provide the information for a “More” button associated with the short description.

Where no detailed information is available, the detailed description SHOULD NOT be provided.

The short description SHOULD NOT be replicated verbatim in the detailed description.

#### Parameters

Item parameters support the insertion of dynamic information into descriptions. The location in the description where a parameter SHOULD be inserted is represented as the identifier for the parameter, surrounded by curly braces. Use of identifiers allows the parameters to be self-documenting. Substitution of parameters SHOULD occur before any other interpretation of the description occurs, for example before markup processing. Where a parameter reference uses an identifier for a non-existent parameter, the parameter value SHOULD be assumed to be an empty string.

Each parameter has a simple string as a value. Parameters MUST NOT be embedded within parameters.

It should be noted that even where only English descriptions are provided, parameters in combination with error codes allow client software to provide multiple language translations, or to replace an agency provided message with one of their preference, while maintaining the dynamic content from the original description.

Because the length of text provided by a parameter is limited, it is possible that the value of a parameter being supplied by an agency may exceed this limit. In this case, an agency MAY replace the parameter reference in the message descriptions with the actual value of the parameter.

#### Locations

It is common practice in user interface design to highlight fields in which errors have occurred. An item MAY thus include one or more locations, which allow client software to intelligently indicate the scope of information affected by the item.

When a location is not able to be determined, the Sequence Number will be set to ‘1’ and Location Path Text will be set to ‘n/a’. In such a scenario the item is assumed to apply to the entire request transaction.

If locations with sequence numbers only are provided, the item applies to the associated documents in the request transaction.

If at least one location includes a path, then all locations within the item SHOULD include a path. In this case, the locations indicate one or more data fields affected by the item. In the common case of a field validation failure, the associated item would have one location, which in turn would have one sequence number and one location path.

#### Location Instance Identifier

Each location MUST include a location instance identifier, which indicates to which logical document in the incoming request the event item applies. The location instance identifier SHALL have the same value as the document ID for the logical document.

#### Location Path Text

The location path field is optionally included in the location to indicate, via an XPath expression, the element in the incoming logical document to which the event item refers. It needs to be interpreted in conjunction with the location instance identifier field, which identifies the particular logical document in the incoming request to which the event item applies.

The XPath expression MUST be interpreted relative to the ***corresponding request***  payload, and will assume the namespace mappings active within this payload. Where possible, XPath expressions SHOULD uniquely identify elements using their ***contextRef*** attribute.

# Secure Communication with sbr ebMS3

This section will only describe the security aspects associated with the ‘message on the wire’. It is assumed that the business has already acquired their AUSkey from the Australian Business Register (ABR) and has installed it in their software package. Details of the registration and certificate issuing process are provided on the AUSkey website.

#### Overview



*Figure 22: Security interactions*

Figure 22 above shows the main security interactions.

1. The business software presents their business certificate to the Security Token Service (STS), is authenticated, and receives an encrypted token.
2. The business software passes the token and certificate to the embeddable client which signs the ebMS message using the certificate, and incorporates the signature, together with the encrypted token from the STS into the WS-Security header.
3. The complete ebMS message is then sent to the SBR ebMS3 platform.
4. The SBR ebMS3 platform decrypts the token and checks various aspects of the security information provided.
5. The agency identifies and authorises the business using data in the decrypted token, processes the business message and returns a response.
6. The business software receives the business response via the SBR ebMS3 platform.

The STS interaction (Token.Request and Token.Response) is an implementation of the standard Web Service Protocol WS-Trust ([http://docs.oasis-open.org/ws-sx/ws-trust/  
200512/ws-trust-1.3-os.html](http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html)).

The SBR interaction (Service.Request & Service.Response) is an implementation of the standard Web Service WS-Security protocol (http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)

The security token is an implementation of the standard SAML 2.0 protocol (<http://saml.xml.org/saml-specifications>).

## Implementation Options

The SBR program provides software developers with two implementation choices for the security framework:

1. Security APIs are provided as part of the SDK (Software Developer Kit) provided by the ATO for use by software developers. The SDK is available in Java, .NET and C versions. It provides a simple API for the STS call for the security token and also the secure messaging to the SBR ebMS3 platform. Software developers should refer to the SDK documentation for details on how to use the APIs. The SDK also includes an example Reference Client. The Reference Client is provided as an example only. The Reference Client is included in the SDK to provide an example implementation of the components and APIs provided in the SDK. The Reference Client provides a guide for how software developers could use these in their developed products. The Reference Client is not supported for production use. It is not intended that the Reference Client be used by software developers for use with or embedded in their developed products. Ongoing support for the Reference Client and inclusion of the Reference Client in future SDK versions is not guaranteed.
2. Software developers who prefer to build their own implementation without dependencies on the SBR SDK can use their preferred platforms to build client code to support the WS-Trust, SAML2, and WS-Security protocols exposed by the STS and SBR ebMS3 platforms.

The remainder of this section provides interface details for software developers who will build directly to the STS/SBR interfaces without using the SDK API.

## Security Token Service (STS)

In a typical WS-Trust scenario, a “relying party” (e.g. SBR ebMS3) specifies a security policy that clients must satisfy. Clients may obtain an identity credential from a registration authority that is not the same as the relying party. In the SBR case, the registration authority is the ABR. The STS has a trust relationship with both the registration authority (ABR) and the relying party (SBR ebMS3). A client that wishes to invoke a service offered by a relying party will normally:

1. Request the security policy from the relying party – which is returned as a set of “claims”.
2. Authenticate to the STS using a valid credential and provide the set of claims it requires to the STS.
3. The STS will provide and sign a set of assertions that validate the client identity.
4. The client then passes these STS signed assertions to the original relying party service end-point and is allowed (or not) to invoke the service.
5. The STS can also provide a session key that is used to encrypt or sign the exchange with the relying party.

In the SBR case, there is no need to request the security policy from SBR ebMS3 because there is a predefined set of claims that are valid across all participating agencies. These claims are listed in the table below, and packaged with the SDK.

Therefore the WS-Trust interactions for SBR are just the STS Token.Request and Token.Response shown in the figure below;



*Figure 23: STS Request and Response envelopes*

In the Token.Request message, the client passes a set of SBR claims to the STS, with confidentiality being provided using SSL. To indicate the source of the request, a timestamp is signed using the business private key, along with an identifier of which relying party the token “AppliesTo”. The token MUST use a “holder-of-key” subjectConfirmation.

The Security Token Service maintains meta-data that relates to a business credential and will return a Token.Response that contains a set of signed assertions (the values associated with the claims), packaged as a security token with a lifetime of 30 minutes. The STS also provides the “holder-of-key” session key that can be used for any number of secure interactions with SBR until expiry.

Within a Token.Request, claims may be marked optional. If a claim is marked optional, the STS will return an assertion if it has a value for the claim and no assertion otherwise. If a claim is not marked optional and the STS does not have a value for the claim, this error condition will be flagged by the return of a SOAP fault rather than a Token.Response.

The table below lists the 16 claims that must be included in each request to the STS. It also shows, for each type of credential offered for use in SBR, whether the resulting assertions must be present within a token included in a request to SBR ebMS3.

Given the STS behaviour in terms of optionality of claims, and in order to simplify the logic necessary to support both credential types, it is RECOMMENDED that all 16 claims be requested as optional in each STS request, except for those in the table below where the claim is marked as mandatory for both credential types. The latter claims SHOULD be marked as mandatory. An example is shown below in Section 6.3.1 ‘Creating the STS Request’.

| Claim URI | Description | ABR\_User | ABR\_Device |
| --- | --- | --- | --- |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/abn | Business ABN | Mandatory | Mandatory |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/commonname> | User full name (ABR\_User)  Server name (ABR\_Device) | Mandatory | Mandatory |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/credentialtype> | Credential Type  (ABR\_User or ABR\_Device) | Mandatory | Mandatory |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/samlsubjectid | SAML subject ID (composite global unique ID) | Mandatory | Mandatory |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/fingerprint> | SH1 hash of the business certificate. | Mandatory | Mandatory |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/sbr\_personid | Business user ID (unique serial within one business) | Mandatory | Not Applicable |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/givennames> | User given names | Mandatory | Not Applicable |
| <http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname> | User family name | Mandatory | Not Applicable |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress | User e-mail address | Mandatory | Not Applicable |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/credentialadministrator | Boolean indicator if user is administrator | Mandatory | Not Applicable |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/previoussubject | ID of any previous credential subject ID (e.g. an ATO certificate) | Optional | Not Applicable |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/stalecrlminutes | Client credential was checked against a Certificate Revocation List that was overdue for replacement by this many minutes. | Mandatory | Mandatory |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/subjectdn> | User X.509 distinguished name | Mandatory | Mandatory |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/issuerdn> | Issuer (ABR) X.509 distinguished name | Mandatory | Mandatory |
| http://vanguard.ebusiness.gov.au/2008/06/identity/claims/notafterdate | Certificate validity expiry date time | Mandatory | Mandatory |
| <http://vanguard.ebusiness.gov.au/2008/06/identity/claims/certificateserialnumber> | Certificate serial number | Mandatory | Mandatory |

*Table 31: STS Claims*

Note that the client software does not need to provide values for these claims. The Token.Request message simply lists the claim URIs shown above and the STS will return the values as a set of assertions within an encrypted token.

The provision of the STS WSDL and sample token request and token response envelopes is discussed in section 7 ‘Testing’.

### Creating the STS Request

This section defines the structure of the Token.Request message. The XML snippet below is an example Token.Request sent to the STS. Note that the base64 data representing the business certificate has been removed for readability.

|  |
| --- |
| <?xml version='1.0' encoding='utf-8'?>  <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">  <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="true">  <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-19714461">  <wsu:Created>2009-10-12T04:02:23.890Z</wsu:Created>  <wsu:Expires>2009-10-12T04:07:23.890Z</wsu:Expires>  </wsu:Timestamp>  <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="CertId-11658721"><!-- Binary data removed --></wsse:BinarySecurityToken>  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-620055">  <ds:SignedInfo>  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  <ds:Reference URI="#Timestamp-19714461">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  <ds:DigestValue>RUqAWdHk+v4Xkx+9Sw0HLVKijpE=</ds:DigestValue>  </ds:Reference>  <ds:Reference URI="#id-3125250">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  <ds:DigestValue>8M8SCNohyZHQEhzru8hIZzIudu8=</ds:DigestValue>  </ds:Reference>  </ds:SignedInfo>  <ds:SignatureValue>  ZHSrVAqDjlZzLdFaz9gEnXFuvpf/q3rQ9XWI3Bu9ZtVkuubhvXXnQo1zeIyiY84uiS/J5lLMpqTN  dQv5JRWZk6Y6XzrEQgfofp5VSmDeyNgVXJtm2FFePnYq7OTZerA3c7jhQO9xc0MpNLfs8NX7zCnI  UNbhgeKR6LFZG8EW1hU=  </ds:SignatureValue>  <ds:KeyInfo Id="KeyId-15834478">  <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="STRId-7789321">  <wsse:Reference URI="#CertId-11658721" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" />  </wsse:SecurityTokenReference>  </ds:KeyInfo>  </ds:Signature>  </wsse:Security>  <wsa:To xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-3125250">https://thirdparty.authentication.business.gov.au/R3.0/vanguard/S007v1.1/service.svc</wsa:To>  <wsa:MessageID>urn:uuid:CC8BEAE32759FDD5821255320143481</wsa:MessageID>  <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>  </soapenv:Header>  <soapenv:Body>  <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">  <RequestType xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</RequestType>  <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">  <EndpointReference xmlns="http://www.w3.org/2005/08/addressing">  <Address> https://test.ato.sbr.gov.au</Address>  </EndpointReference>  </wsp:AppliesTo>  <TokenType xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</TokenType>  <Claims xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512" xmlns:i="http://schemas.xmlsoap.org/ws/2005/05/identity" Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">  <i:ClaimType Optional="false" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/abn" />  <i:ClaimType Optional="false" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/commonname" />  <i:ClaimType Optional="false" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/credentialtype" />  <i:ClaimType Optional="false" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/samlsubjectid" />  <i:ClaimType Optional="false" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/fingerprint" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/sbr\_personid" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/givennames" />  <i:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" />  <i:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/credentialadministrator" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/stalecrlminutes" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/subjectdn" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/issuerdn" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/notafterdate" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/certificateserialnumber" />  <i:ClaimType Optional="true" Uri="http://vanguard.ebusiness.gov.au/2008/06/identity/claims/previoussubject" />  </Claims>  <Lifetime xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512">  <wsu:Created xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-10-12T04:02:22.984Z</wsu:Created>  <wsu:Expires xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-10-12T04:32:22.984Z</wsu:Expires>  </Lifetime>  <KeyType xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</KeyType>  <KeySize xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512">512</KeySize>  </wst:RequestSecurityToken>  </soapenv:Body>  </soapenv:Envelope> |

*Figure 24: Token.Request sample message*

The Token.Request message includes the following elements:

* The soapenv:Header element contains the WS-Addressing elements that define the location end-point of the STS, a unique message ID, and the requested action (to Issue a token).
* The <wst:RequestSecurityToken> element in the SOAP Body defines the specifics of the request to the STS and includes the following key elements:
* The <wsp:AppliesTo> element defines the URL of the service that requires the SAML token. It can be scoped either to individual services within SBR (in which case a token will be needed for each service), or at a point in the path one above, in which case a single token can be used across the four services. The latter approach SHOULD be used (as shown in the example) as it minimises the number of token requests needed.
* The <TokenType> element defines the required token as a SAML 2.0 token.
* The <Claims> element and all the related <i:ClaimType> elements specify exactly which identity assertions the STS is to include in the Token.Response message (as encrypted assertions).
* The optional <Lifetime> element defines the required lifetime of the session key to be returned by the STS. Note that the STS may enforce a maximum allowed validity period and so it is possible that the lifetime actually provided may be less than the lifetime requested. The minimum lifetime for an SBR session key will be 30 minutes.
* The <KeyType> element and <KeySize> element specify that the returned session key SHOULD be a 512 bit symmetric key.

The time on the system requesting the token MUST NOT differ from that of the STS Service by more than 5 minutes.

### Processing the STS Response

This section provides guidelines on how client software should process the security token and session key that is returned from the STS in the Token.Response message.

The XML snippet below is an example response from the STS. Note that the base64 data representing the encrypted token has been removed for readability.

|  |
| --- |
| <?xml version='1.0' encoding='utf-8'?>  <s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:a="http://www.w3.org/2005/08/addressing">  <s:Header>  <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</a:Action>  <a:RelatesTo>urn:uuid:CC8BEAE32759FDD5821255320143481</a:RelatesTo>  <ActivityId xmlns="http://schemas.microsoft.com/2004/09/ServiceModel/Diagnostics" CorrelationId="6da8a0dc-89cf-4210-9d6b-36b16ffb3788">00000000-0000-0000-0000-000000000000</ActivityId>  <o:Security xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" s:mustUnderstand="1">  <u:Timestamp u:Id="\_0">  <u:Created>2009-10-12T04:02:26.742Z</u:Created>  <u:Expires>2009-10-12T04:07:26.742Z</u:Expires>  </u:Timestamp>  </o:Security>  </s:Header>  <s:Body>  <trust:RequestSecurityTokenResponseCollection xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">  <trust:RequestSecurityTokenResponse>  <trust:KeySize>512</trust:KeySize>  <trust:Lifetime>  <wsu:Created xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-10-12T04:02:26.454Z</wsu:Created>  <wsu:Expires xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-10-12T04:32:22.984Z</wsu:Expires>  </trust:Lifetime>  <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">  <a:EndpointReference>  <a:Address> https://test.ato.sbr.gov.au</a:Address>  </a:EndpointReference>  </wsp:AppliesTo>  <trust:RequestedSecurityToken>  <EncryptedAssertion xmlns="urn:oasis:names:tc:SAML:2.0:assertion">  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  <e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">  <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  </e:EncryptionMethod>  <KeyInfo>  <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">  <X509Data>  <X509IssuerSerial>  <X509IssuerName>CN=Australian Government Notary Services OCA, OU=For Development purposes ONLY, OU=Australian Authentication and Notary Services, O=Australian Government, C=AU</X509IssuerName>  <X509SerialNumber>116425329959729741023280816821386492610</X509SerialNumber>  </X509IssuerSerial>  </X509Data>  </o:SecurityTokenReference>  </KeyInfo>  <e:CipherData>  <e:CipherValue>G10F8BxPQNcshpWwiPgvoofH74IsiNpL1h9bP4pZPHwyxrlO+xirH5XAMqi+BkTCBbFojAEJaYIvu9NIqFZ8THUZVIbhjBge6miNrsx+kRz70+QJKv6F9WmCmH+rQgWNi1T1MjEP9xIcVLcZDzvFBPEJlvK13KLV5Hoimxp/8F8=</e:CipherValue>  </e:CipherData>  </e:EncryptedKey>  </KeyInfo>  <xenc:CipherData>  <xenc:CipherValue><!-- Binary data removed --></xenc:CipherValue>  </xenc:CipherData>  </xenc:EncryptedData>  </EncryptedAssertion>  </trust:RequestedSecurityToken>  <trust:RequestedProofToken>  <trust:BinarySecret>DolCv6k0OrHRKqMZa5AgH28SC7ntQN1EgOXybaYq9GYh3ppK6gfpuHRR4NplJo2sEcnQ6+djWRs8orObzXDclQ==</trust:BinarySecret>  </trust:RequestedProofToken>  <trust:RequestedAttachedReference>  <SecurityTokenReference xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:b="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" b:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">  <KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">\_d5511ae3-5ab6-474a-b58f-5752b847ab15</KeyIdentifier>  </SecurityTokenReference>  </trust:RequestedAttachedReference>  <trust:RequestedUnattachedReference>  <SecurityTokenReference xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:b="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" b:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">  <KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">\_d5511ae3-5ab6-474a-b58f-5752b847ab15</KeyIdentifier>  </SecurityTokenReference>  </trust:RequestedUnattachedReference>  <trust:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</trust:TokenType>  <trust:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</trust:RequestType>  <trust:KeyType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</trust:KeyType>  </trust:RequestSecurityTokenResponse>  </trust:RequestSecurityTokenResponseCollection>  </s:Body>  </s:Envelope> |

*Figure 25: Token.Response sample message*

The token response includes the following elements:

1. The <trust:RequestSecurityTokenResponseCollection> is the envelope element that contains all the token data. It contains one <trust:RequestSecurityTokenResponse>.
2. The <trust:KeySize> element defines the key size of the symmetric session key. The session key is used to sign envelopes sent to SBR. This is for information only.
3. The <trust:Lifetime> element defines the validity period for the symmetric session key. The client should request a new session key before expiry in order to continue any interactions with SBR.
4. The <wsp:AppliesTo> element is the identifier for the service end point (in this case the SBR ebMS3 platform) that is the “relying party” in the WS-Trust interaction. This is for information only.
5. The <trust:RequestedSecurityToken> element contains the <EncryptedAssertion> structure. The client should insert the entire <EncryptedAssertion> structure into the SOAP header of the SBR Service.Request envelope as described in 5.5. This element contains all the identity claims (encrypted) together with the necessary key reference information that allows SBR to decrypt the claims.
6. The <trust:RequestedProofToken> element contains the symmetric key that is used by the client to sign interactions with SBR ebMS3 until it expires and must be renewed. This is not currently used but is included for future use.
7. The <trust:RequestedAttachedReference> element contains the unique identifier for the symmetric session key (the <trust:RequestedProofToken>). This is not currently used.
8. The <trust:TokenType> (SAML2), <trust:RequestType> (Issue), and <trust:KeyType> (Symmetric) elements contain informational data about the nature of the Token.Response. No specific action is required from client software.

### STS Faults

(Information in this section has been extracted from section 4 of the VANguard S007 Security Token Service Technical Service Contract, Contract Version V3.0, and Document Revision Number 1.7)

Errors are returned from the STS Service via the SOAP 1.2 fault mechanism.

It should be noted that the WS-Trust 1.3 specification requires all faults to use a Code of “env:Sender” irrespective of the source of the error. While inconsistent with the description for the Code field in the SOAP 1.2 specification, faults generated by the STS nonetheless comply with the requirements of the WS-Trust specification. When processing a fault from the STS, the footnoted codes below should be processed as if they were “env:Receiver” faults.

#### BusinessContext Element

This element provides structured detail regarding the nature of the fault. The information is reproduced in the fault Reason/Text element.

#### EventCode Element

This element contains the VANguard specific error code. The table below documents the possible fault codes that can be received from the Security Token Service. The code is available in the sub code tree consistent with the approach used by SBR ebMS3 faults, but is also available as part of the information in the BusinessContext element.

#### EventSeverity Element

This element contains the severity of the error. The value will be one of Normal, Warning, Severe or Critical. This element may be used for diagnostic and debugging purposes.

It should be noted that, regardless of the value of this element, the fact that a fault has been returned means that a token has not been provided, and submission to SBR cannot proceed until the source of the error has been rectified.

#### EventDescription Element

This element provides a verbose, human readable description of the fault, and complements that provided in the Reason/Text element. It should be used for diagnostic and debugging purposes.

#### UserAdvice

This element provides advice targeted at a non-technical user. It may assist in resolving the conditions that produced the fault.

| Fault/Code/Value | Fault/Code/Subcode/Value | Vanguard SubCodes | Reason for Error |
| --- | --- | --- | --- |
| env:Sender | wst:InvalidRequest | E2183 | A mandatory request was made for an unrecognised claim. |
| env:Sender  env:Sender  env:Sender  env:Sender  env:Sender  env:Sender  env:Sender | wst:FailedAuthentication | E2014 | The credential supplied by the initiating party has been revoked. |
|  |  | E2169 | The credential supplied by the initiating party is not recognized. |
|  |  | E2015 | The credential supplied by the initiating party has expired. |
|  |  | E2017 | The validity start date of the credential supplied by the initiating party is in the future. |
|  |  | E2029 | The credential supplied by the initiating party could not be processed and may be corrupt. |
|  |  | E2020 | The Credential Authority that issued the credential supplied by the initiating party is not recognized. |
|  |  | E2180 | No usage policy for the credential supplied could be found. This would occur if a certificate that was valid but not supported by the STS was presented. |
| env:Sender  env:Sender  env:Sender  env:Sender  env:Sender  env:Sender | wst:RequestFailed | E2003 | The relying party specified in the AppliesTo element is not recognized. |
|  |  | E10012 E10032 E10042 | The request could not be satisfied due to an internal VANguard error. |
|  |  | E2001 | The token type specified in the request was not recognised. Only SAML2.0 tokens should be requested. |
|  |  | E2001 | An unknown request type was encountered in the message. Typically the request type should be: http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue |
|  |  | E2190 | Claim data could not be found due to an internal VANguard error. Attempt the request again. |
|  |  | E2182 | A mandatory claim specified in the request could not be provided. Check the claim types being specified in the request. |
| env:Sender | wst:MissingAppliesTo | E2001 | The AppliesTo element of the RST was not supplied. This element must be supplied in any request to the STS. |
| env:Sender | wsse:UnsupportedSecurityToken |  | An unsupported token was provided. |
| env:Sender | wsse:UnsupportedAlgorithm |  | An unsupported signature or encryption algorithm was used. |
| env:Sender | wsse:InvalidSecurity |  | An error was discovered processing the <wsse:Security> header. |
| env:Sender | wsse:InvalidSecurityToken |  | An invalid security token was provided. |
| env:Sender | wsse:FailedAuthentication |  | The security token could not be authenticated or authorized. |

*Table 32: STS Fault Codes*

The figure below shows an example of a fault returned from the Security Token Service.

|  |
| --- |
| <?xml version="1.0" ?>  <s:Envelope xmlns:s="**http://www.w3.org/2003/05/soap-envelope**" xmlns:a="**http://www.w3.org/2005/08/addressing**">  <s:Header>  <a:Action s:mustUnderstand="**1**">**http://www.w3.org/2005/08/addressing/soap/fault**</a:Action>  <a:RelatesTo>**urn:uuid:dd729e3e-7c97-4515-814a-980564ab48c8**</a:RelatesTo>  </s:Header>  <s:Body>  <s:Fault>  <s:Code>  <s:Value>**s:Sender**</s:Value>  <s:Subcode>  <s:Value xmlns:a="**http://docs.oasis-open.org/ws-sx/ws-trust/200512**"> **a:FailedAuthentication**</s:Value>  <s:Subcode>  <s:Value xmlns:v=" **http://vanguard.business.gov.au/2009/02**">**v:E2015**</s:Value>  </s:Subcode>  </s:Subcode>  </s:Code>  <s:Reason>  <s:Text xml:lang="**en-AU**"> **The Initiating Party Certificate (Issuer:CN=Test Australian Business Register CA, OU=Certification Authority, O=Australian Business Register, C=AU , SerialNumber:00CD) had a status of 'Expired.'. Event Code: [E2015]. Event Severity: [Normal]. Event Description: [Business User certificate expired on [25/04/2009 12:02:56 AM].]. User Advice: [Advise Business User that their certificate has expired and to contact the issuing CA to apply for a new certificate.]. Agency Reference: []. VANguard Reference: []. Transaction Id: [].**</s:Text>  </s:Reason>  <s:Detail>  <BusinessContext xmlns="**http://vanguard.business.gov.au/2009/02**">  <EventCode>**E2015**</EventCode>  <EventSeverity>**Normal**</EventSeverity>  <EventDescription> **The Initiating Party Certificate (Issuer:CN=Test Australian Business Register CA, OU=Certification Authority, O=Australian Business Register, C=AU , SerialNumber:00CD) had a status of 'Expired.'.**</EventDescription>  <UserAdvice> **Advise Business User that their certificate has expired and to contact the issuing CA to apply for a new certificate.**</UserAdvice>  </BusinessContext>  </s:Detail>  </s:Fault>  </s:Body>  </s:Envelope> |

*Figure 26: SOAP fault indicating presentation of an expired credential to the STS*

**NOTE:** The inclusion of application specific processing detail within the Fault/Reason/Text element is deprecated and may be removed in future versions. Developers SHOULD NOT rely on parsing this field and instead SHOULD use the Fault/Code tree or Fault/Detail element to obtain any VANguard specific data.

## Secure Messaging

SBR Web Service security has the following goals:

* To ensure confidentiality of business data. This is achieved through transport layer security (SSL).
* To ensure tamper proofing and non-repudiation of origin for business reports. This is achieved through a digital signature of the ebMS message using the business certificate.
* To support identification and authentication of business users independent of government agency. This is supported through the STS call and SAML assertions within the encrypted token.
* To support non-repudiation of receipt by government agencies. This is achieved through agency digital signatures on the response (Synchronous interaction) and receipt (Asynchronous interaction).

The message security implementation employs standards defined in the WS-I Basic Security Profile v1.1.

The conceptual security structure of any SBR service request and response is shown in the figure below.



*Figure 27: SBR Secure Messaging*

The Service.Request message envelope includes:

* In the SOAP header;
  + ebMS3 header
  + WSSE security header containing:
    - Encrypted SAML token exactly as received from the STS.
    - XML Digital Signatures (signed using the business digital certificate) for:
      * Root ebMS3 Messaging element
      * Soap Body
      * SAML token
      * MIME parts containing ebMS3 attachments
* Any number of payload documents (0 or more) as zero or more MIME parts.

The Service.Response message envelope includes:

* In the SOAP header;
  + ebMS3 header
  + A WSSE security header element containing the digital signature of the ebMS response message using the Responding Agency’s key.
* Any number of business response documents as one or more MIME parts.

## Signature Structures

The sample envelope below provides an example of the SBR security header. Note that carriage returns have been inserted and Base64 strings have been truncated for readability.

The SOAP header contains a <wsse:Security> structure which will contain:

* A <saml2:EncryptedAssertion> that carries identity information for the agency. This information is provided by the STS.
* A <wsse:BinarySecurityToken> that carries the business certificate. This is used by the agency to validate the document signature.
* A <ds:Signature > element that carries the signature of the ebMS message, and is signed with the business certificate. This signature is for non-repudiation of origin.

All these structures are described in more detail in subsequent sections.

|  |
| --- |
| <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="true">  <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="soapheader-2">  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Type="http://www.w3.org/2001/04/xmlenc#Element">  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"></xenc:EncryptionMethod>  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">  <xenc:EncryptedKey Recipient="name:RelyingParty-Test-STS">  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5"></xenc:EncryptionMethod>  <dsig:KeyInfo>  <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">  <X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">  <X509IssuerSerial>  <X509IssuerName>CN=Australian Government Notary Services OCA, OU=For Development purposes ONLY, OU=Australian Authentication and Notary Services, O=Australian Government, C=AU</X509IssuerName>  <X509SerialNumber>116425329959729741023280816821386492610</X509SerialNumber>  </X509IssuerSerial>  </X509Data>  </o:SecurityTokenReference>  </dsig:KeyInfo>  <xenc:CipherData>  <xenc:CipherValue> /+euKyWoJmES+ghWa/hnSkMPvHQTn6B0sSSTAVJu5c=</xenc:CipherValue>  </xenc:CipherData>  </xenc:EncryptedKey>  </dsig:KeyInfo>  <xenc:CipherData>  <xenc:CipherValue> / +euKyWoJmES +ghWa/hnSkMPvHQTn6B0sSSTAVJu5c= </xenc:CipherValue>  </xenc:CipherData>  </xenc:EncryptedData>  </saml2:EncryptedAssertion>  <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="signingCert"> MIIEHDCCAwSgAwIBAgICCkMwDQYJKoZIhvcNAQEFBQAwgYUxCzAJB</wsse:BinarySecurityToken>  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  <ds:SignedInfo>  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:CanonicalizationMethod>  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"></ds:SignatureMethod>  <ds:Reference URI="#soapheader-1">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>  <ds:DigestValue>gUDWcR5d9S9rgFJAdvyxirwSCLk=</ds:DigestValue>  </ds:Reference>  <ds:Reference URI="#soapheader-2">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>  <ds:DigestValue>DIV/E4evJSS35XK0Y8OlexLBppo=</ds:DigestValue>  </ds:Reference>  <ds:Reference URI="#soapbody">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>  <ds:DigestValue>lVf6gg3lYIQe95cm3r25FwQ7j6U=</ds:DigestValue>  </ds:Reference>  <ds:Reference URI="cid:Attachment1">  <ds:Transforms>  <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Signature-Transform"></ds:Transform>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>  <ds:DigestValue>yx77UStpFeqfigoArHkKxuZPHsI=</ds:DigestValue>  </ds:Reference>  </ds:SignedInfo>  <ds:SignatureValue>itb5yjzUiVeoEt6mIwiiwIrqZEST60Y7E4anLO/n/X4iFSQd75rWBVwACKYNBwCuZj3QZqLjuqyg 0hmFDglyWWB0KkBtok+03LT7oV8t/Ox/T4pkDIlY4oLAvuhoXloM9Q+QRtBPghbziCEgWdC621An L6pusC4WfVA4QzqBLp4=</ds:SignatureValue>  <ds:KeyInfo Id="KeyId-56DB2BD279F414595214264879074171">  <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="STRId-56DB2BD279F414595214264879074212">  <wsse:Reference URI="#signingCert" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"></wsse:Reference>  </wsse:SecurityTokenReference>  </ds:KeyInfo>  </ds:Signature>  </wsse:Security> |

*Figure 28: Sample Security Header*

### Identity Token <saml2:EncryptedAssertion>

This structure contains the list of assertions that provide identity information to the Agency. The actual assertions are represented as an encrypted string contained in the <xenc:CipherValue> element near the end of the structure. The remainder of the elements provide the agency with the necessary data to decrypt the assertions:

* The assertions are encrypted with a symmetric key that must be passed to SBR ebMS3. The symmetric key is itself encrypted using SBR ebMS3’s public key. The <xenc:CipherValue> element just above the encrypted assertions contains the encrypted symmetric key.
* The <o:SecurityTokenReference> element provides the id reference of the public key used to encrypt the symmetric key.
* The encryption algorithm for the assertions is AES-256 (symmetric key encryption). The encryption algorithm for the symmetric key is RSA (asymmetric key encryption).

The structure is part of the Token.Response from the STS and can be inserted into the wsse security header without change.

### Business Certificate <wsse:BinarySecurityToken>

This structure contains the digital certificate of the business. The certificate is encoded as a base64 string and is identified using the attribute wsu:Id="signingCert". The agency will use the public key contained in this certificate to validate the ebMS message signature.

### Message Signature <ds:Signature>

This structure contains the digital signature of the ebMS message. There are three main sub-structures:

* The <ds:SignedInfo> element is the information that is actually signed. It identified the parts of the SOAP envelope that are signed. These parts are referenced using the ds:Reference URI elements that locate elements within the envelope identified by a wsu:id. In the case of the document signature, the signed parts are the ebMS3 root eb:Messaging element (<ds:Reference URI=" #soapheader-1">), the SAML token (<ds:Reference URI=”#soapheader-2”>), the soap body (<ds:Reference URI=” #soapbody”) and all MIME parts containing the payload attachments (<ds:Reference URI=”cid:[Content-ID header value of the MIME part]”)
* The <ds:SignatureValue> element contains signature value for the canonicalized SignedInfo element.
* The <ds:KeyInfo> element contains the reference to the certificate used to create the signature. In this case the certificate is identified by reference to the business certificate described in the previous section using the <wsse:Reference URI=" #signingCert" element.

# Testing

## Overview

The SBR program supports the following types of testing to assist software developers in the testing of their products, allowing software developers to progressively “step up” from basic tests to more sophisticated tests. The supported testing types are:-

1. Network Connectivity Testing
2. Message Connectivity Testing
3. Service Specific Testing

Further explanation of these testing types is provided below in sections 7.2, 7.3 and 7.4.

### Service End Points

The two environments that software developers have access to are:

1. The testing environment called the External Vendor Test Environment (EVTE) which will be available to software developers to test their own systems and is intended to be used for all three types of testing supported by the SBR program.
2. The production environment (PROD) which can be used for network and message connectivity testing as part of executing diagnostic functions within a software package.

This information has been moved into a reference document ‘ATO SBR Physical End Points’ which also contains information on the SSL Certificates to be used.

## Network Connectivity Testing

Network connectivity testing can be carried out using a web browser on the client computer where the client MSH is hosted (“Client System”). This test will help determine if the Client System is able to physically connect to the SBR ebMS3 Server across the internet.

To use a browser to test the connectivity to the SBR ebMS3 Server from the Client System:

1. On the Client System open a Web browser, such as Internet Explorer.
2. In the browser command line, copy-paste/type a service endpoint URL from the above table (Table 2). For example, <https://test.ato.sbr.gov.au/services/Single-sync>
3. When the Web page appears, look for one of the following results :
   * If the Web Page shows a HTTP status code ‘404 Not Found’ error it indicates that the browser was able to connect to the website, but the page you wanted was not found. This result confirms that network connectivity is present between the Client System and the SBR ebMS3 Server ( at EVTE or PROD – depending on the URL used) .
   * If the browser displays a webpage, with another HTTP status code then it indicates an unsuccessful network connection between the Client System and the SBR ebMS3 Server.   
     Check the Client System's network connections and that network services are running on the Client System.

## Message Connectivity Testing

After confirming the establishment of network connectivity between the Client System and the SBR ebMS3 Server , the next step is to generate a syntactically valid ebMS3/AS4 request message, including the necessary SBR ebMS3 security ”tokens” in the message and confirm that the SBR ebMS3 Server is able to process the request message and respond correctly.

To assist in this testing, the SBR ebMS3 Server offers a simple connectivity testing service, called the “MSH Ping” service, which is based on the ebMS3 specification. The MSH Ping service enables one MSH to determine if another MSH is operating. It consists of:-

* + sending a ‘Ping’ request message from the Client MSH (which is the MSH on the Client System) to one of the available end points that is listed in the above table (Table 2)
  + the MSH – which is on the SBR ebMS3 Server responding with a Message Service Handler ‘Pong’ response message.

### Ping Request

Follow the instructions provided in the SDK Guide to create an ebMS3 request message for one of the supported service invocation types and set the Service and Action within the ‘CollaborationInfo’ element to the values provided below to invoke the ‘MSH Ping Service’:

|  |  |
| --- | --- |
| **Service**: | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/service |
| **Action**: | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/test |

Submit the request as per the service invocation instructions and look for one of the following results:

* **Ping Test is Successful**: when the client MSH receives a Pong response. Subsequent section provides more details on the Pong Response.
* **Ping Test is Unsuccessful**: when no response is received by the MSH client.

### Pong Response

Pong Response is the message that is sent from the MSH on the SBR ebMS3 Server to the client MSH as a response to a Ping Request.

In case of a Two Way Sync invocation, the response is an ebMS3 User Message with the SOAP Body containing the message ‘This *is test response from MEIG’*.

|  |
| --- |
| <soapenv:Body>  <testresponse>This is test response from MEIG</testresponse>  </soapenv:Body> |

In the case of an Async invocation (One/Two way Push), the Pong Response is a standard ebMS3 Signal receipt message

Receipt of the Pong Response message confirms/verifies the following:

* Network connectivity between the Client MSH and SBR ebMS3 Server MSH
* Correctness of ebMS3 related configuration including Exchange Profiles and Conformance Policies.
* Security Compliance and Message Signature check success.
* SAML Authentication success.

## Service Specific Testing

Having determined that ebMS3 messages can be successfully generated, secured and sent to SBR ebMS3, and that the resulting response can be interpreted, full testing of the desired reports and business obligations using the ebMS3 Service Actions (Interactions) offered by the SBR ebMS3 Server can commence.

SBR ebMS3 offers a range of artefacts to assist this stage of testing including test credentials, test cases and test data. For each Service Action delivered into EVTE a conformance suite consisting of test scenarios, list of ABN’s/TFN’s and request and response messages will be provided.

# Supporting Files

For each service interaction delivered into EVTE a conformance suite consisting of test scenarios, list of ABN’s/TFN’s, request and response messages will be provided as part of the SDK.

The conformance suite will include the following artefacts:

* All required AUSkey Credentials - provided by ATO · A document outlining all test scenarios including expected result.
* A spread-sheet containing test input data values.
* Expected request message/Payload instance should be produced by the software package.
* The expected ATO response message/Payload instance.
* The conformance suite template will be based on ATO template

# 

# APPENDIX A: ebMS Errors

**Standard ebMS Errors (excerpted from ebMS Core 3.0 Spec -** [**Link**](http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html)**)**

This section defines the standard error codes expected to be generated and processed by a conformant MSH. They are segmented according to the stage of processing they are likely to occur: during reliable message processing, security processing, and general ebMS processing.

1. **ebMS Processing Errors**

The table below describes the Errors that may occur within the ebMS Module itself (ebMS Errors that are not Escalated Errors), i.e. with @origin="ebms". These errors MUST be supported by an MSH, meaning generated appropriately, or understood by an MSH when reported to it.

| Error Code | Short Description | Recommended Severity | Category Value | Description or Semantics |
| --- | --- | --- | --- | --- |
| EBMS:0001 | ValueNotRecognized | failure | Content | Although the message document is well formed and schema valid, some element/attribute contains a value that could not be recognized and therefore could not be used by the MSH. |
| EBMS:0002 | FeatureNotSupported | warning | Content | Although the message document is well formed and schema valid, some element/attribute value cannot be processed as expected because the related feature is not supported by the MSH. |
| EBMS:0003 | ValueInconsistent | failure | Content | Although the message document is well formed and schema valid, some element/attribute value is inconsistent either with the content of other element/attribute, or with the processing mode of the MSH, or with the normative requirements of the ebMS specification. |
| EBMS:0004 | Other | failure | Content |  |
| EBMS:0005 | ConnectionFailure | failure | Communication | The MSH is experiencing temporary or permanent failure in trying to open a transport connection with a remote MSH. |
| EBMS:0006 | EmptyMessagePartitionChannel | warning | Communication | There is no message available for pulling from this MPC at this moment. |
| EBMS:0007 | MimeInconsistency | failure | Unpackaging | The use of MIME is not consistent with the required usage in this specification. |
| EBMS:0008 | FeatureNotSupported | failure | Unpackaging | Although the message document is well formed and schema valid, the presence or absence of some element/ attribute is not consistent with the capability of the MSH, with respect to supported features. |
| EBMS:0009 | InvalidHeader | failure | Unpackaging | The ebMS header is either not well formed as an XML document, or does not conform to the ebMS packaging rules. |
| EBMS:0010 | ProcessingModeMismatch | failure | Processing | The ebMS header or another header (e.g. reliability, security) expected by the MSH is not compatible with the expected content, based on the associated P-Mode. |
| EBMS:0011 | ExternalPayloadError | failure | Content | The MSH is unable to resolve an external payload reference (i.e. a Part that is not contained within the ebMS Message, as identified by a PartInfo/href URI). |

1. **Security Processing Errors**

The table below describes the Errors that originate within the Security Module, i.e. with @origin="security". These errors MUST be escalated by an MSH, meaning generated appropriately, or understood by an MSH when reported to it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Error Code | Short Description | Recommended Severity | Category Value | Description or Semantics |
| EBMS:0101 | FailedAuthentication | failure | Processing | The signature in the Security header intended for the "ebms" SOAP actor could not be validated by the Security module*.* |
| EBMS:0102 | FailedDecryption | failure | Processing | The encrypted data reference the Security header intended for the "ebms" SOAP actor could not be decrypted by the Security Module. |
| EBMS:0103 | PolicyNoncompliance | failure | Processing | The processor determined that the message's security methods, parameters, scope or other security policy-level requirements or agreements were not satisfied. |

1. **Reliable Messaging Errors**

The table below describes the Errors that originate within the Reliable Messaging Module, i.e. with @origin="reliability". These errors MUST be escalated by an MSH, meaning generated appropriately, or understood by an MSH when reported to it.

| Error Code | Short Description | Recommended Severity | Category Value | Description or Semantics |
| --- | --- | --- | --- | --- |
| EBMS:0201 | DysfunctionalReliability | failure | Processing | The reliability function as implemented by the Reliability module is not operational, or the reliability state associated with this message sequence is not valid. |
| EBMS:0202 | DeliveryFailure | failure | Communication | Although the message was sent under Guaranteed delivery requirement, the Reliability module could not get assurance that the message was properly delivered, in spite of resending efforts*.* |

# APPENDIX B: Supported pModes

SBR ebMS3 supported pModes:

1. **Two Way Sync**

| **Parameter** | **Value** | **Comment** |
| --- | --- | --- |
| **GENERAL** | | |
| PMode.Agreement | http://sbr.gov.au/agreement/Gateway/1.0/TwoWaySync/PKI |  |
| PMode.MEP | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/twoWay |  |
| PMode.MEPbinding | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/sync |  |
| **PROTOCOL** | | |
| PMode.Protocol.Address | HTTPS 1.1 |  |
| PMode.Protocol.SOAPVersion | SOAP 1.2 |  |
| **ERROR HANDLING** | | |
| PMode[1].ErrorHandling.Report.AsResponse | True |  |
| PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer | True |  |
| PMode[1].ErrorHandling.Report.DeliveryFailuresNotifyProducer | True |  |
| PMode[1].ErrorHandling.Report.MissingReceiptNotifyProducer | False | There is no receipt in two way sync scenario |
| **SECURITY** | | |
| PMode[1].Security.WSSVersion | WSS 1.1 |  |
| PMode[1].Security.X509.Sign | True | Sign all soap elements and attachments |
| PMode[1].Security.X509.Signature.HashFunction |  | SHA-256 |
| PMode[1].Security.X509.Signature.Algorithm |  | RSA |
| PMode[1].Security. X509.Encryption.Encrypt | False | All connections protected via  SSL/TLS |
| PMode[1].Security.PModeAuthorize | False |  |
| PMode[1].Security.SendReceipt | False | There is no receipt in two way sync scenario |
| **BUSINESS INFORMATION** | | |
| PMode.BusinessInfo.MPC |  | N/A for Two Way Sync Pattern |
| PMode[1].BusinessInfo.PayloadProfile.maxSize | 2 MB (pre-compression) |  |
| **COMPRESSION** | | |
| Pmode[1].PayloadService.CompressionType | Application/gzip |  |

1. **Two Way Async**

PMode[1] applies for the Push flow of the MEP

PMode [2] applies for the Pull flow of the MEP

PMode[ 1,2] applies for both Push and Pull flows

| **Parameter** | **Value** | **Comment** |
| --- | --- | --- |
| **GENERAL** | | |
| PMode.Agreement | http://sbr.gov.au/agreement/Gateway/1.0/TwoWayPushAndPull/PKI |  |
| PMode.MEP | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pushAndPull |  |
| PMode[1].MEPbinding | http://docs.oasis-open.org/ebxmlmsg/ebMS/v3.0/ns/core/200704/push |  |
| PMode[2].MEPbinding | http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pull |  |
| **PROTOCOL** | | |
| PMode.Protocol.Address | HTTPS 1.1 |  |
| PMode.Protocol.SOAPVersion | SOAP 1.2 |  |
| **ERROR HANDLING** | | |
| PMode[1].ErrorHandling.Report.AsResponse | True |  |
| PMode[2].ErrorHandling.Report.AsResponse | False |  |
| PMode[1,2].ErrorHandling.Report.ProcessErrorNotifyConsumer | False |  |
| PMode[1,2].ErrorHandling.Report.  ProcessErrorNotifyProducer | True |  |
| PMode[1,2].ErrorHandling.Report.DeliveryFailuresNotifyProducer | True |  |
| PMode[1,2].ErrorHandling.Report.MissingReceiptNotifyProducer | True |  |
| **SECURITY** | | |
| PMode[1,2].Security.WSSVersion | WSS 1.1 |  |
| PMode[1,2].Security.X509.Sign | True | Sign all soap elements and attachments |
| PMode[1,2].Security.X509.Signature.HashFunction |  | SHA-256 |
| PMode[1,2].Security.X509.Signature.Algorithm |  | RSA |
| PMode[1,2].Security. X509.Encryption.Encrypt | False | All connections protected via  SSL/TLS |
| PMode[1,2].Security.PModeAuthorize | True |  |
| PMode[1,2].Security.SendReceipt | True | Implemented in  conjunction with  PMode[1].  ReceptionAwareness |
| Pmode[1].Security.SendReceipt.ReplyPattern | Response |  |
| Pmode[1].Security.SendReceipt.NonRepudiation | True |  |
| Pmode[2].Security.SendReceipt.ReplyPattern | Call Back |  |
| Pmode[2].Security.SendReceipt.NonRepudiation | False |  |
| **RECEPTION AWARENESS & DUPLICATION DETECTION** | | |
| PMode[2].ReceptionAwareness | True |  |
| PMode[2].ReceptionAwareness.Retry | True |  |
| PMode[2].ReceptionAwareness.Retry.Parameters | As per Agency Implementation Guide |  |
| **BUSINESS INFORMATION** | | |
| PMode.BusinessInfo.MPC |  | Default MPC |
| PMode[1].BusinessInfo.PayloadProfile.maxSize | 2 MB (pre-compression) |  |
| **COMPRESSION** | | |
| Pmode[1].PayloadService.CompressionType | Application/gzip |  |

1. **One Way Push**

| **Parameter** | **Value** | **Comment** |
| --- | --- | --- |
| **GENERAL** | | |
| PMode.Agreement | http://sbr.gov.au/agreement/Gateway/1.0/Push/PKI |  |
| PMode.MEP | http://docs.oasis-open.org/ebxmlmsg/ebMS/v3.0/ns/core/200704/oneWay |  |
| PMode.MEPbinding | http://docs.oasis-open.org/ebxmlmsg/ebMS/v3.0/ns/core/200704/push |  |
| **PROTOCOL** | | |
| PMode.Protocol.Address | HTTPS 1.1 |  |
| PMode.Protocol.SOAPVersion | SOAP 1.2 |  |
| **ERROR HANDLING** | | |
| PMode[1].ErrorHandling.Report.AsResponse | True |  |
| PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer | True |  |
| PMode[1].ErrorHandling.Report.DeliveryFailuresNotifyProducer | True |  |
| PMode[1].ErrorHandling.Report.MissingReceiptNotifyProducer | True |  |
| **SECURITY** | | |
| PMode[1].Security.WSSVersion | WSS 1.1 |  |
| PMode[1].Security.X509.Sign | True | Sign all soap elements and attachments |
| PMode[1].Security.X509.Signature.HashFunction |  | SHA-256 |
| PMode[1].Security.X509.Signature.Algorithm |  | RSA |
| PMode[1].Security. X509.Encryption.Encrypt | False | All connections protected via  SSL/TLS |
| PMode[1].Security.PModeAuthorize | False |  |
| PMode[1].Security.SendReceipt | True | Implemented in  conjunction with  PMode[1].  ReceptionAwareness |
| Pmode[1].Security.SendReceipt.ReplyPattern | Response |  |
| Pmode[1].Security.SendReceipt.NonRepudiation | True |  |
| **RECEPTION AWARENESS & DUPLICATION DETECTION** | | |
| PMode[1].ReceptionAwareness | True |  |
| PMode[1].ReceptionAwareness.Retry | True |  |
| PMode[1].ReceptionAwareness.Retry.Parameters | Maxretries= 16,period= 3hours |  |
| PMode[1].DuplicateDetection | True |  |
| PMode[1].DuplicateDetection.Parameters | Checkwindow= 24hours |  |
| **BUSINESS INFORMATION** | | |
| PMode.BusinessInfo.MPC |  | Use Default MPC |
| PMode[1].BusinessInfo.PayloadProfile.maxSize | 1 GB (pre-compression) |  |
| **COMPRESSION** | | |
| Pmode[1].PayloadService.CompressionType | Application/gzip |  |

1. **One Way Pull**

| **Parameter** | **Value** | **Comment** |
| --- | --- | --- |
| **GENERAL** | | |
| PMode.Agreement | http://sbr.gov.au/agreement/Gateway/1.0/SelectivePull/PKI |  |
| PMode.MEP | http://docs.oasis-open.org/ebxmlmsg/ebMS/v3.0/ns/core/200704/oneWay |  |
| PMode.MEPbinding | http://docs.oasis-open.org/ebxmlmsg/ebMS/v3.0/ns/core/200704/push |  |
| **PROTOCOL** | | |
| PMode.Protocol.Address | HTTPS 1.1 |  |
| PMode.Protocol.SOAPVersion | SOAP 1.2 |  |
| **ERROR HANDLING** | | |
| PMode[1].ErrorHandling.Report.AsResponse | False |  |
| PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer | True |  |
| PMode[1].ErrorHandling.Report.DeliveryFailuresNotifyProducer | True |  |
| PMode[1].ErrorHandling.Report.MissingReceiptNotifyProducer | True |  |
| **SECURITY** | | |
| PMode[1].Security.WSSVersion | WSS 1.1 |  |
| PMode[1].Security.X509.Sign | True | Sign all soap elements and attachments |
| PMode[1].Security.X509.Signature.HashFunction |  | SHA-256 |
| PMode[1].Security.X509.Signature.Algorithm |  | RSA |
| PMode[1].Security. X509.Encryption.Encrypt | False | All connections protected via  SSL/TLS |
| PMode[1].Security.PModeAuthorize | True |  |
| PMode[1].Security.SendReceipt | False | Implemented in  conjunction with  PMode[1].  ReceptionAwareness |
| Pmode[1].Security.SendReceipt.  ReplyPattern | N/A |  |
| Pmode[1].Security.SendReceipt.  NonRepudiation | N/A |  |
| **RECEPTION AWARENESS & DUPLICATION DETECTION** | | |
| PMode[1].ReceptionAwareness | True |  |
| PMode[1].ReceptionAwareness.Retry | True |  |
| PMode[1].ReceptionAwareness.Retry.Parameters | As per Agency Implementation Guide |  |
| PMode[1].DuplicateDetection | True |  |
| PMode[1].DuplicateDetection.Parameters | Checkwindow=24hours |  |
| **BUSINESS INFORMATION** | | |
| PMode.BusinessInfo.MPC |  | Default MPC |
| PMode[1].BusinessInfo.PayloadProfile.maxSize | 1 GB (pre-compression) |  |
| **COMPRESSION** | | |
| Pmode[1].PayloadService.CompressionType | Application/gzip |  |

# APPENDIX C: Previous Version History

| Version | Revision Date | Summary of Change |
| --- | --- | --- |
| 1.6.1 | 17/11/2017 | **Section 2.6 Common Characteristics**   * The following change has been made in this section   From: “All response messages going out from SBR ebMS3 will be signed using the relevant agency’s key.”  To: “Only one-way pull response messages going out from SBR ebMS3 as a result of Bulk and Batch Pull request will be signed using the relevant agency’s key. All other responses will remain unsigned.”  Section **5.5 Business Event**   * Minor updates to refer to relevant agency documentation for information related to Business Event processing.   **Section 5.6 Message Events section and 5.6.2 Severity Code section**   * Added details for partially rejected requests. |
| 1.6 | 27/03/2017 | **Section 5.4 Transport Exceptions** 5.4.1.3 Overall Event Message Block to Report Transport Exceptions  * Removed information related to SystemError  5.4.2.3 Platform Internal errors  * Removed section related to SystemError   **Section 5.6 Message Events**  5.6.2 Severity Code   * Removed section related to SystemError Section |
| 1.5 | 01/03/2017 | **Section 2.6 Common Characteristics**   * Message Compression – updated to include compression changes for Bulk/Batch and Collect messages going out of SBR ebMS3 |
| 1.4 | 23/07/2015 | **Section 2.3 Service Attributes**  2.3.2 Request Message Types   * Batch – updated to list point 2 with additional information of three sub-types. * 2. Bulk re-numbered to 3. Bulk   2.3.3 Response Time Service Levels   * Intermediate – updated to list point 2. * ‘2. Delayed’ renumbered to ‘3. Delayed’ with a call out to a note below (^). * ‘\* ‘ note updated to include reference to system load in the second sentence. * ‘^’ note added – clarifying batch and bulk behaviours.   **Section 2.4 Supported Service Invocation Types**  Table updated to include a new row for Bulk Async Intermediate.  2.4.2 Single- Async - chatty   * Point (e) updated with *error code EBMS:0006 and description “EmptyMessagePartitionChannel”* reference.   2.4.3 Batch – Async – Intermediate, Batch – Async – Delayed, Bulk – Async – Intermediate and Bulk - Async - Delayed   * Reference to Bulk Async Intermediate added   2.4.3.1 Batch and Bulk Request Message Rules   * Reference to Bulk Async Intermediate added to second dot point * Third sub point under third dot point has been deleted (referenced response times and SLAs).   2.4.3.2 Invocation of Services for Batch and Bulk Request Message Processing   * Point (ii) updated with *error code EBMS:0006 and description “EmptyMessagePartitionChannel”* reference.   2.4.4 Collect   * Point (e) updated with *error code EBMS:0006 and description “EmptyMessagePartitionChannel”* reference.   2.4.5 Polling Interval   * Updated to reference agency Implementation Guides * Sections 2.4.5.1 to 2.4.5.4 inclusive deleted.   **Section 3 Message Structure**  3.3 UserMessage SBR ebMS3 Profile   * Table headings updated from ‘Single Request 2 Way Sync 1 Way Push’ to ‘Single Request 2 Way Sync Push (Single/Collect)’ * Table headings updated from ‘Response 2 Way Sync 1 Way Push’ to ‘Response 2 Way Sync 1 Way Pull Pull (Single/Collect)’.   3.3.1 eb:UserMessage/eb:MessageInfo   * ebMessageId description updated to include reference that the value cannot exceed 255 characters.   3.4.2.2 eb:SignalMessage/eb:Error   * Clarification of the @category name in table 25:eb:Error Structure.   **Section 4 Message Packaging**  4.2 Single Pull Request   * Reference to message id element criteria added.   4.6 Batch/Bulk Pull Request (Selective Pull)   * Reference to MIME part deleted along with Figure 10 – Batch/Bulk Request Message Packaging.   **Section 5 Error Management**  5.4.2.1 Client Side Error Location Path Text   * Reference to incorrect exchange profile added along with error code table.   5.4.2.2 Platform Unavailability   * Reference to re-queing removed from the second paragraph. * Figure 15 updated with the correct http://sbr.gov.au reference.   5.4.2.3 Platform Internal Errors   * Reference to correlation id deleted in the second row of the error code table under description/comment. * Figures 17 and 18 updated with the correct http://sbr.gov.au reference.   5.6.5.1 Sequence number changed to 5.6.5.1 Location Instance Identifier  5.6.5.2 Location Path Text   * Reference to sequence number changed to location instance identifier.   **Section 7 Testing**  7.1.1 Service End Points   * Table updated to include EVTE and PROD Collect-async end points.   7.2 Network Connectivity Testing   * Reference to ‘403 Forbidden’ changed to ‘404 Not Found’ in the first dot point. * Reference to ‘404 Not Found’ removed in the second dot point. |
| 1.3 | 18/06/2015 | - Updated to align the Security section with platform implementation  - Addressed feedback from SWD’s  - Removed redundant properties from ebMS3 header element eb:PartProperties  - Updated the details for eb:SignalMessage message structure |
| 1.2.1 | 01/12/2014 | Updates:   * Updated to reflect the support for payload types in addition to XBRL |
| 1.2 | 17/11/2014 | Updates:   * Specification of Collect message type functionality * Corrections and additions relating to use of Logical Record terminology. * Miscellaneous other corrections |
| 1.1 | 30/06/2014 | * Major Update : Completed Section 7 (Testing) * Major Update : Added p-Modes in Appendix B * Minor Updates to the following :   + Section 8 (Supporting Files) Completed   + Namespace Information provided   + Section 5 (including Fig11)   + Business Error Code final list     - Changes to table 28,     - Changes to figure 14,   + Property Name Change in Section 3.3.5.2 |
| 1.0 | 20/06/2014 | Document base-lined: suitable for use. |
| 0.8 | 13/06/2014 | SBR1 renamed to SBR Core Services; SBR2 renamed to SBR ebMS3. Document re-formatted for ease of reading. |
| 0.7 | 04/04/2014 | * Major changes to Section 5 – Error Management. * Minor Changes based on 2nd round of review. * Other Minor cosmetic changes. |
| 0.6 | 28/1/2013 | Major changes based on review feedback. |
| 0.5 | 20/12/2013 | Incorporated comments from internal review. |
| 0.4 | 29/11/2013 | Incorporated comments from internal review. |
| 0.3 | 29/11/2013 | Incorporated comments from internal review. |
| 0.2 | 15/11/2013 | Addressed comments from internal review. |
| 0.1 | 06/11/2013 | Initial Draft. |

1. Initial releases of this document are aimed at software application developers transitioning from ELS to SBR [↑](#footnote-ref-1)