An Australian Government Initiative

Standard Business Reporting

SBR

STANDARD BUSINESS REPORTING

SBR CORE WEB SERVICES IMPLEMENTATION GUIDE (WIG)

V2.2d

Purpose: This document provides advice to software developers in the implementation of calls to SBR Core Services.

Date: 10 February 2016

Contact: For further information or questions, contact the SBR Program Office at SBRServiceDesk@sbr.gov.au or call 1300 488 231

DOCUMENT CHANGE CONTROL

Version number	Date of issue	Author(s)	Brief description of change
1.0	01/05/2009	Michael Leditschke	Initial Release
2.0	17/08/2009	Michael Leditschke	Update to reflect V2.0 WSDLs and 2010 Security Design
2.1	31/08/2009	Michael Leditschke	1. Add Commonwealth copyright notice and logo. 2. Correct errors in element names as compared to the WSDL schemas and add a paragraph flagging that the schemas take precedence over the WIG. Also include a reference to the specific version of the WSDLs to which this document applies. 3. Correct minor typographical errors. 4. Add four additional optional claims that should be included in STS requests, correct the description of the stalecrlminutes claim, rearrange the order of the claims in the table to list mandatory claims first, heading for device certificate in claims table should be ABR_Device not ABS_Device, refine the descriptive text in relation to optionality of claims, align examples to the claims table. 5. givennames, surname, emailaddress and credentialadministrator claims are now mandatory for user certificates. 6. Add additional information on testing – network connectivity testing, SBDM testing. 7. Add a table of end points (URLs) at which services are available to software developers. This includes both Core Services and the VANguard STS. 8. Add SOAP fault codes for use with the SBDM testing, to identify use of unsupported attachment mechanisms and to indicate the XML of the request is malformed. Update the malformed XML example to use this new fault code. 9. Correct the SOAP envelope Structure diagram to show 0 or more, not 1 or more, attachments. 10. Finalise designation text table. 11. Add a comment clarifying that the message connectivity test is subject to agency limits on the

Version number	Date of issue	Author(s)	Brief description of change
			maximum number of XBRL instances and binary attachments. 12. Add a clarification indicating how content that is dynamically inserted by agencies into error messages, but becomes part of the static text of the message, is indicated in sample message lists. 13. Complete a missing paragraph in section 4.6.1.3. 14. Update the information on error description parameters to reflect the final decisions to use named rather than indexed parameters and only support error messages in English. (These changes were inadvertently not included when the error framework was incorporated into V2.0d of the WIG from an internal SBR document) 15. Add a clarification indicating the acceptable encodings that can be used in Web Service calls. 16. Use the terms ”reporting obligations”, “business collaborations” and “report message” to align with the terminology used in the Taxonomy Architecture. 17. Add information in regard to available WS-Policy files. 18. Add a clarification indicating the value of the Message.Type.Text to be used in responses to requests where the Message.Type.Text value is not recognised by the receiving agency. 19. Add a clarification indicating the format of the URLs to be used in business document validation URIs 20. Remove references to the timestamp service, which is no longer required 21. Update the description of the STS service to cover the use of SSL rather than message level encryption. 22. Update the description of the isEditable and isVisible attributes to indicate that they should be namespace qualified, rather than being unqualified.
2.2	31/05/2010	Michael Leditschke	1. Updated use of terms such as MUST, SHOULD and MAY in this document in line with RFC 2119. In some cases, this will result in a change of wording to reflect the author’s original intent.

Version number	Date of issue	Author(s)	Brief description of change
			2. Added the following fault codes SBR.GEN.FAULT.SOFTWAREBLOCKED SBR.GEN.FAULT.SOFTWARENOTREGISTERED SBR.GEN.FAULT.INVALIDSBDM SBR.GEN.FAULT.UNKNOWNMESSAGETYPETEXT SBR.GEN.FAULT.UNKNOWNVALIDATIONURI SBR.GEN.FAULT.INVALIDTOKENSIGNATURE 23. Clarified from where the reason text should be derived in a fault returned to client software as a result of a fault being generated by an agency 24. Clarified the expected handling by Core Services of env:Sender vs env:Receiver faults generated by agencies 25. Clarified the need for clock synchronisation when using the Security Token Service 26. Documented the handling of fault “Node” element values for env:Receiver faults generated by agencies 27. Documented the error information that may be received from the Security Token Service 28. Added additional clarification in regards to message connectivity testing 29. Added additional clarification in regards to the addition of timestamps to messages 30. Added additional clarification in regards to the values to be provided as part of the SoftwareInformation, and its use as part of software self-certificaiton processes 31. Minor grammatical and spelling corrections 32. Provided guidance on a range of items associated with XBRL instance creation, including element ordering, meaning of IDs on contexts and units, provision of accuracy information and the handling of monetary units 33. Replace references to the SBR credential with

Version number	Date of issue	Author(s)	Brief description of change
			"AUSkey" 34. Clarify the interpretation of EventItem severity for XBRL payload formats that support multiple lodgements in a single request, such as PAYG summaries and TFN declarations.
2.2.d	10/01/2016	Vitaly Sidorenko	1. Updated the SBR contact email address, approving authority and copyright statement. 2. Updated the definition of existing services as authenticated and to add anonymous endpoints to them. 3. Replaced the term "SBR" to "SBR Core Services" when it is applicable to the SBR Core Services platform to distinguish it from SBR ebMS3. 4. Removed references to Knowledge Repository. 5. Updated implementation options to include sample applications and C platform. 6. Added the following soap fault codes: • SBR.GEN.FAULT.MISMATCHEDPAYLOADS • SBR.GEN.FAULT.UNSUPPORTEDPAYLOAD • SBR.GEN.FAULT.UNKNOWNPAYLOADTYPE 7. Updated the description of the following soap faults to replace "XBRL" word with the "payload" word: • SBR.GEN.FAULT.TOOMANYINSTANCES • SBR.GEN.FAULT.TOOMANYDOCUMENTS 8. Removed the soap fault SBR.GEN.FAULT.WSSECURITYNOTPERMITTED as not supported. 9. Added XML as a new supported payload type. 10. Updated description of checking for Message.Type.Text values in SBR Core Services. 11. Updated Figure 8 to reflect change in the soap Fault Reason.Text value. 12. Updated Table 24 to reflect changes in endpoints related to decommission of Test SBDM service and

Version number	Date of issue	Author(s)	Brief description of change
			introduction of anonymous endpoints. 13. Removed section 2.2.2.1. WS-Policy as not applicable. 14. Updated Table 25 to reflect current Vanguard endpoints.

DOCUMENT APPROVALS

This document was approved by:

Name	Title	Date
Neil Tothill	Senior Director Solution Architecture & Integration	10/02/2016

COPYRIGHT

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1. PURPOSE

The purpose of this document is to provide the information that will assist Software Developers in the implementation of calls to the web services offered by SBR Core Services.

1.2. AUDIENCE

The audience for this document is any organisation that will be building SBR Core web services into their products. Typically this will be software application developers.

Readers should be familiar with the following:

- SBR Program – please see **www.sbr.gov.au** for further information.
- XBRL – please see **www.xbrl.org** for further information.
- XML – please see www.xml.org for further information.
- Web Services – please see **www.ws-i.org** for further information.

1.3. CONTEXT

The SBR program offers a suite of documents and technical products to support software developers. These are illustrated in Figure 1. 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 software development kits and conformance test suites that aim to facilitate efficient implementation.

SBR Solution Overview

An overview of the SBR 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 XBRL Taxonomy and shows how the library of harmonised data elements (the “SBR AU Taxonomy”) is packaged and how the data elements are re-used across government forms (the “SBR AU Reports”). 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.

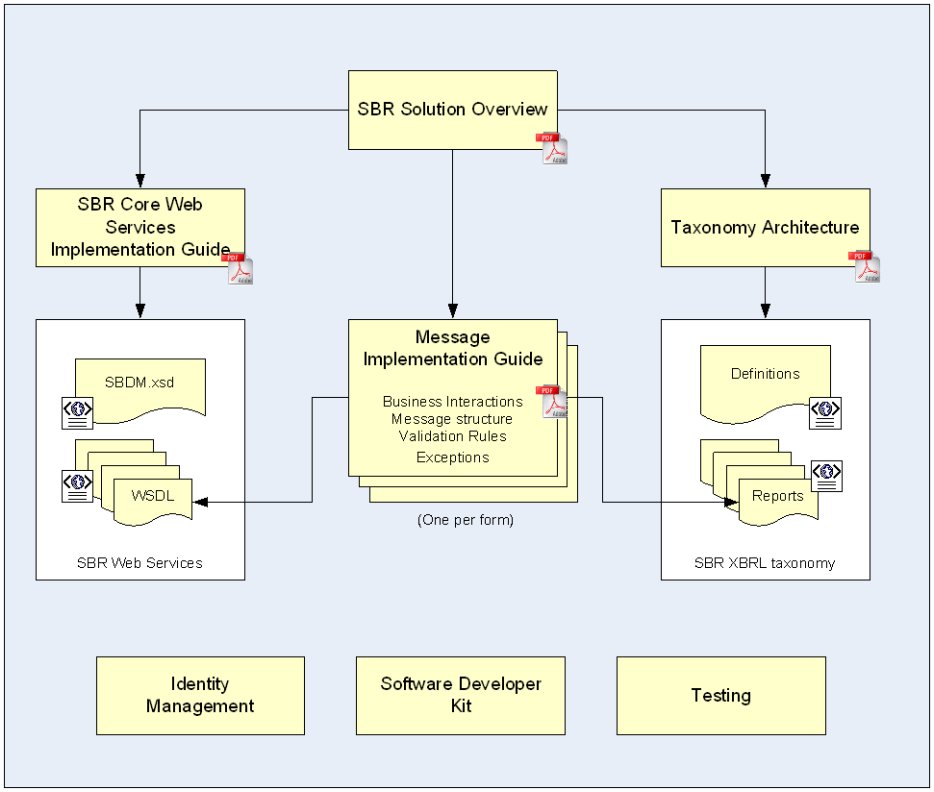

Figure 1: SBR Core Services Solution Artefacts

SBR XBRL Taxonomy

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

SBR Core Services Web Service Implementation Guide (WIG)

This document describes common technical components and services that are re-used by all business services when using SBR Core Services platform. The common services include a single gateway that exposes four web services and supports industry standard web service protocols for message exchange, a standard business document message, a security token service, and a standardised approach to handling business error conditions and transport exceptions.

Web Services

The "Web Services Description Language" (WSDL) is the W3C standard syntax for the definition of web services. A WSDL describes the service, the information exchanged, and the technical protocols used for the exchange. SBR Core Services provides "list", "pre-fill", "pre-lodge" and "lodge" web services together with a WSDL for each. This collection also includes the XML Schema for the SBR Standard Business Document Message (SBDM) which is a mandatory part of every message.

Message Implementation Guide (MIG)

There is a MIG for each report in scope for SBR. The MIG is the entry point for an implementer wishing to support a specific SBR 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

lodgements, “pre-fill” with government data, “calculate” an obligation, and “lodge” a report). The MIG provides a business collaboration model, message content, and business rules for each requesting / responding message that supports the implementation of a reporting obligation.

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

Identity Management

The SBR solution leverages the AUSkey authentication credential that will be accepted by all participating agencies. This document explains how the credential is issued and managed. It also explains how it is linked to agency business services to authorise primary credential holders or their delegates (employees or intermediaries).

Software Developer Kit (SDK)

There are some common technical components that the SBR program expects will be needed by all implementers. The SDK is a set of components created for Java, .NET and C platforms that are available for software developers to use in their products. Details of the SDK are provided on the SBR Web Site.

Testing

The SBR program will provide implementers with a suite of test services that can be used to test both the technical (web service) and 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.

1.4. TERMINOLOGY

For definition of the terminology and acronyms used within this document, please refer to the **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**. The use of the word “Mandatory” is to be read as “MUST”.

1.5. 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	NAMESPACE
env	http://www.w3.org/2003/05/soap-envelope
list	http://sbr.gov.au/list.02.service
prefill	http://sbr.gov.au/prefill.02.service
prelodge	http://sbr.gov.au/prelodge.02.service

lodge	http://sbr.gov.au/lodge.02.service
sbdm	http://sbr.gov.au/comn/sbdm.02.data
core	http://sbr.gov.au/comn/core.02.data
sbr	http://sbr.gov.au/comn/core.02.data
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

Table 1: Namespace Prefixes

2. SBR CORE SERVICES ARCHITECTURE

2.1. OVERVIEW

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

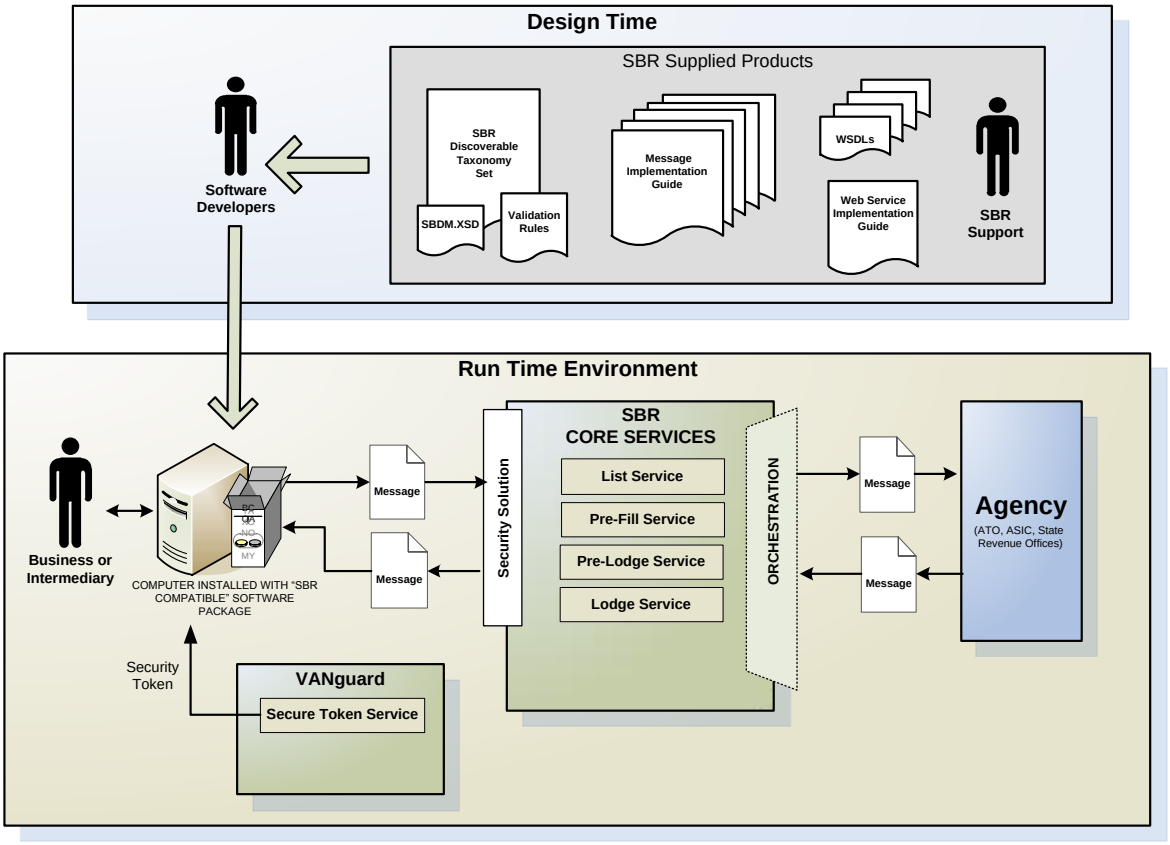

Figure 2: SBR Core Services High Level Platform Overview

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

The primary responsibility of the SBR CS is to seamlessly and securely mediate between business service requests and agencies. Whenever a business makes an SBR web service request, SBR CS will receive it and perform:

1) Authentication – ensure the message is appropriately secured and contains the necessary identity information to confirm it has come from a known entity,

2) Message structure validation,

3) Route to the designated Agency,

4) Wait for a response, and then hand the response back to the business.

2.2. WEB SERVICES

2.2.1. Services Offered

SBR Core Services exposes the following four technical services for use by business or government software. The degree to which each service is used as part of completion of a particular reporting obligation varies, with not all services necessarily being involved for every obligation.

SERVICE	ROLE
List Report	Allows businesses to retrieve list-based data (such as obligations to be fulfilled or summaries of previous lodgements).
Pre-fill Report	Allows businesses to retrieve information known to the agency responsible for a report
Pre-Lodge Report	Allows a business to perform a “pre-lodgement” call – this performs functions such as providing the results of complex agency calculations or checking the validity of information prior to lodgement, depending on the business scenario.
Lodge Report	Allows a business to lodge a report to an agency.

Table 2: Technical Web Services Offered By SBR Core Services

Every service provides secured (authenticated) and non-secured (anonymous) endpoints. Non-secured endpoints are allowed in Core Services on the per agency basis when an Agency decides that this is required.

Subsequent sections provide further information in regards to each service.

2.2.2. Web Service Standards

SBR CS SHALL conform to the SOAP 1.2 recommendation provided by the World Wide Web consortium, and SHALL employ a range of related recommendations generically referred to as the "WS*" suite. Table 3 below indicates the key recommendations that SHALL be employed by SBR CS. Please note that messages sent to anonymous endpoints SHOULD NOT contain SAML token as for authenticated services.

CATEGORY	APPLICABLE RECOMMENDATIONS
Transport	HTTP 1.1 HTTP over TLS (With constraints as per ACSI 33)
Messaging	RFC2392: Content-ID and Message-ID Uniform Resource Locators SOAP 1.2 MTOM 1.0
Description	WSDL 1.1 WSDL 1.1 Section 3.0 WSDL 1.1 Section 5.0 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 3: Recommendations used by SBR Core Services

1. WSDLs will contain the policy definition to enable MTOM as an example of how to add policies to the WSDLs, but this will be commented out to ensure utility of WSDLs without alteration.

   There is a high level of consistency in regards to the MTOM policy assertions across platforms.

2. Additional platform specific policies will be provided in separate files for integration as desired by developers, based on the pattern of the MTOM policy.

This allows developers to select the degree to which they wish to understand and leverage the WS-Policy support within their platform.

3. The portfolio of policies will be organically grown based on use and testing of policies on specific platforms by agencies and software developers.

   In order to facilitate re-use of artefacts across the developer community, additional examples will be made available as and when they become available to the program.

2.2.3. Common Characteristics

All web services supported via SBR Core Services platform have the following common characteristics:

- Use of a request/response style of interaction

  With the exception of communication timeouts, any request SHALL always receive a response. All requests SHALL be initiated by business software.

- Synchronous calls

  All web service calls to SBR Core Services MUST be synchronous in nature. While most responses will be received within seconds, business software SHOULD be designed to cater for delays of the order of minutes. It is thus RECOMMENDED that the process of interacting with SBR Core Services be decoupled from the user interaction with the package.

- Separation of web service intent from business intent

  The web services offered are “generic”, in that the way in which the message payload structure is expressed only constrains business documents to be provided in well-formed XML, and attachments to be any binary object. This may be contrasted with the approach where the web service contract includes the structure of the business documents.

  The advantage of this approach is that the web services infrastructure is not affected as new reporting obligations (and hence document formats) are added to the portfolio of reporting obligations covered by SBR.

- Business Documents represented in one of supported payload format (currently XBRL or XML but new types can be supported in the future)

  The business data associated with a request, and the resulting business data provided by the government in any response, MUST be represented in the same payload format used for the request payload. The XBRL business document formats are defined by the SBR AU Reports, with details of message content, optionality and rules being described in a Message Implementation Guides for corresponding business interactions (see section 2.3). Non-XBRL payloads should be compliant with appropriate schemas.

- 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 as one or more payload instance documents optionally with attachments.

- A standardised approach to the indication of failure conditions

  Failures related to the transport of SOAP messages SHOULD be indicated using SOAP Faults, while business level errors MUST be represented in a standard format within the response message (see section 4 for more details). If the request is unsuccessful, any business information, for example documents or attachments, in the response SHOULD be ignored.

- A single end-point per service regardless of agency

  As suggested by figure 2, SBR Core Services provides a single physical end point address for each service for each environment offered (production, end-to-end testing).

  A standard field within the message structure is used to determine to which agency a request MUST be forwarded. Beyond this field, business software does not require any knowledge of the mechanisms needed to communicate with the agencies involved in SBR CS. Thus new agencies MAY be added to SBR CS without the need to retest the web services infrastructure of business 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 (see section 2.3).

  In some circumstances, particularly where requests have been batched together, this will require business software to issue multiple requests for what is logically a single submission, an example being the submission of multiple Tax File Number Declarations or PAYG Summaries for a single company.

  Software developers are encouraged to consider whether multiple smaller requests can be made in real-time rather than the more traditional batch approach.

2.2.4. List Service

This service can be used by a business or reporting intermediary wishing to determine what reporting obligations they have from a particular agency, or to retrieve a summary list of previously submitted reports and their outcomes / totals for use in formulating another report – for example an end-of-year submission.

The caller provides search criteria in the payload instance contained in the request which is tailored to the particular type of reporting obligation and is defined by the agencies.

2.2.5. Prefill Service

Some reports supported by SBR allow businesses / authorised intermediaries to retrieve “pre-fill” data from the agency. Pre-fill data populates sections of the report with business specific information known by the agency, rather than the business / authorised intermediary needing to provide the information.

Other reports require pre-fill data as a pre-condition to report lodgement. An example of such a report is the “Business Activity Statement (BAS)”. A pre-condition for lodging this report to the Australian Taxation Office (ATO) is that a business / authorised intermediary would load this

report in its accounting software package with applicable pre-populated data from ATO prior to completing the report and then lodging it to ATO.

A business / authorised intermediary would use this service to retrieve single or multiple report(s) with applicable pre-populated data and optional single or multiple attachments (if applicable) for a specified reporting period from an agency.

Pre-fill data SHALL be requested for a single report period (where a reporting obligation period can also include a one-off or point-in time reporting obligation). Typically this will result in one pre-filled report instance being returned. In some scenarios there MAY be more than one reporting obligation for a reporting period – each is the same report type, but covering different non-overlapping sub-periods of the reporting period, and with different pre-filled information contained in the pre-filled report instances provided. In the case of the BAS, for example, the ATO will return a BAS report with pre-populated data applicable to the requesting business which may consist of a single or multiple report instances in XBRL format.

2.2.6. Prelodge Service

The Pre-Lodge Report service allows a business or reporting intermediary to send reporting data to an agency without completing a lodgement. This service is used in several business scenarios across the various agencies – for example:

a) Allowing a business or reporting intermediary to lodge a completed report for the purposes of executing complex agency business rules against the lodgement (in effect performing a lodgement without “committing” the result to the agencies’ back end systems);

b) Allowing a business or reporting intermediary to execute agency back-end calculations on a partially filled report.

For example, a report may be partially filled by a business or reporting intermediary to populate financial data which is pre-lodged. The agency then uses this data to calculate fields such as marginal tax rates and returns that information (in addition to the original report fields filled out and provided in the pre-lodge action) to the business or reporting intermediary, allowing them to complete the report and lodge it.

2.2.7. Lodge Service

This service allows a business or reporting intermediary to lodge a report.

Requests will be processed as soon as they are received by SBR. Requests are processed in parallel (or concurrently), so at any point in time there may be many requests in-progress.

Due to the inherent nature of parallel processing it is not guaranteed that agencies receive requests in the same order as originally received by SBR.

Agencies are expected to provide meaningful response information for report lodgements, including for example non-technical human readable status information, in order that these can be viewed by business users without change.

Given the importance of lodgement operations, there are two components in the message structure which are included solely for use within lodgement responses:-

1. Lodgement Receipt

This structure provides a lodgement receipt identifier and lodgement date and time for use by business when discussing lodgements with the agency concerned (see section 3.3.3.4)

This structure SHOULD be used to return lodgement information to the user. Any variation from this will be described in the Message Implementation Guide (see section 2.3)

2. Non-repudiation token (proof of receipt)

The ability to provide a proof of receipt token was included in the design to assist businesses in providing evidence of lodgement completion. A container for it is provided via a structure within the SOAP header of the lodgement response (see section 3.2.2).

Due to variation in the legislation under which SBR agencies operate, no token will initially be provided by SBR Core Services and thus business software SHOULD ignore this field. The container has been retained in the design to allow for future developments in this area.

2.3. MESSAGE IMPLEMENTATION GUIDES (MIG)

The web services described above provide the building blocks from which the more complex collaborations needed to fulfil an SBR reporting obligation are built. As far as possible, the web services and the business payloads they carry are loosely coupled so that additional reporting obligations MAY be added without requiring retesting of the web service infrastructure.

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

There is a MIG for each reporting obligation (e.g. Payroll Tax NSW OSR, ATO Activity Statement) and its primary purpose is to describe:

- The business interactions required to fulfil the reporting obligation which in turn drives the required web 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 web service message structure.
- Any specific use of optional fields within the web service infrastructure.

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

3. MESSAGE STRUCTURE

3.1. OVERVIEW

All SBR Core services SHALL employ a common message format shown in the diagram below.

All messages SHALL be carried over a one way HTTPS transport, and employ the SOAP 1.2 envelope structure. SOAP messages MUST employ UTF-8 or UTF-16 character encodings.

Details of the structures used within the SOAP Header and Body are described in subsequent sections. In the case of a discrepancy between this document and the WSDL schemas, the WSDL schemas SHALL take precedence and SHALL be considered normative.

This document applies to V2.1 of the SBR Core Services WSDLs. XBRL is used as a sample payload type in the figure below.

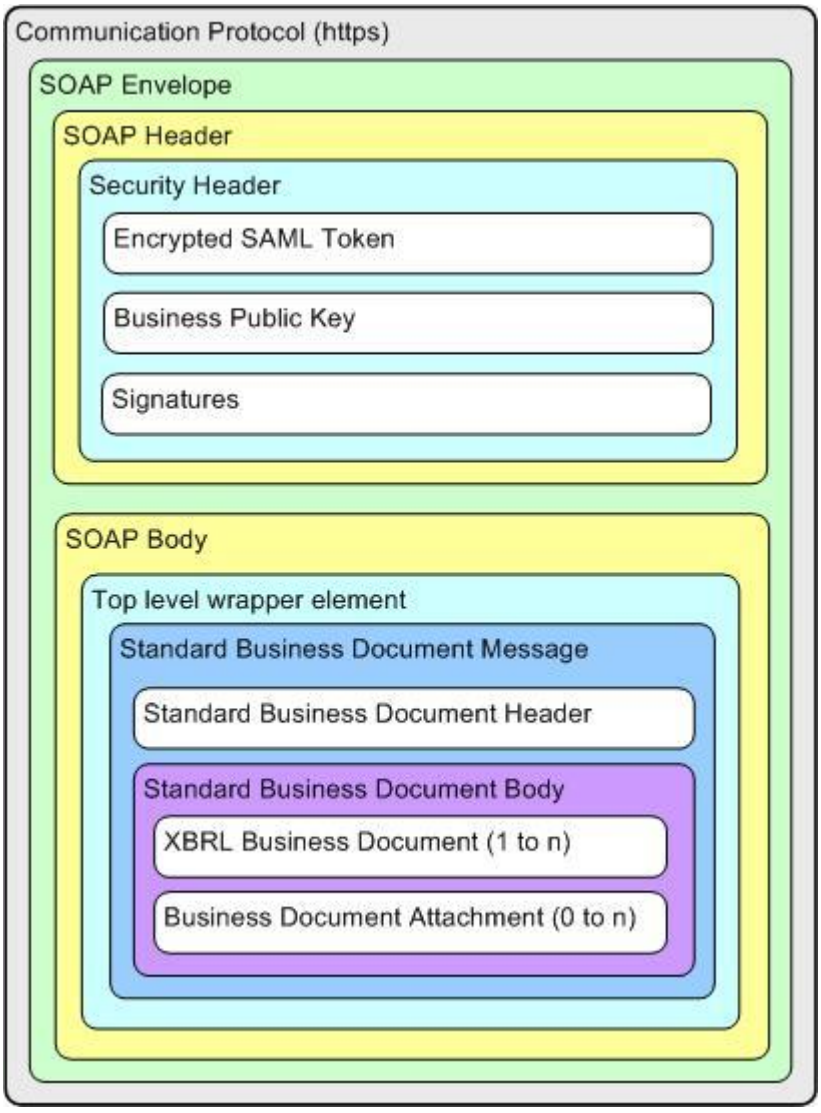

Figure 3: SBR Core Services SOAP Envelope Structure

3.2. SOAP HEADER

The SOAP header is a mandatory element for all authenticated calls to carry out the security related information so it MUST be present for List, Lodge, Prelodge and Prefill services when they are called via secured endpoint. However this element is not used for the calls to anonymous endpoints and therefore SHOULD NOT be present in anonymous messages.

3.2.1. Security Element

Because SBR Core Services 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 5 provides a detailed description of the security requirements and implementation.

This Security Element SHOULD NOT be included in the messages submitted via anonymous endpoints and it will be ignored if presented.

3.2.2. NonRepudiation Element

As described in section 2.2.7, the SOAP header of every lodge response MUST include a core:NonRepudiation element, containing a single child core:Message.NonRepudiationToken.Text element, having no content.

It has been retained in the design to allow for future capability in the area of non-repudiation, but SHOULD be ignored at present.

3.3. SOAP BODY

3.3.1. Top Level Wrapper Elements

SBR Core Services follows the recommendations of the WS-I Organisation Basic Profile 1.0. Thus each SOAP body SHALL contain a single child element, as shown in the table below.

SERVICE	REQUEST	RESPONSE
List	list:RequestList	list:ResponseList
Prefill	prefill:RequestPreFillReport	prefill:ResponsePreFillReport
Prelodge	prelodge:RequestPreLodgeReport	prelodge:ResponsePreLodgeReport
Lodge	lodge:RequestLodgeReport	lodge:ResponseLodgeReport

Table 4: SOAP Body Child Elements

3.3.2. Standard Business Document Message (SBDM)

Because of the generic nature of the web services offered by SBR Core Services, there is a high degree of commonality in the message structures used for both requests and responses across all SBR Core web services.

Thus the service top level wrapper elements employ a common content model – a single Standard Business Document Message element, or SBDM for short.

The SBDM is described by a single XML schema. Optionality is used throughout the SBDM structure to allow for the limited variability between requests and responses.

The SBDM consists of two elements - a MANDATORY header called the Standard Business Document Header (SBDH), and an OPTIONAL body called the Standard Business Document Body (SBDB).

ELEMENT	PURPOSE	OPTIONALITY
sbdm:StandardBusinessDocumentHeader	Facilitate Message Exchange	MANDATORY
sbdm:StandardBusinessDocumentBody	Facilitate Business Collaboration	OPTIONAL

Table 5: Standard Business Document Message Content Model

The header carries the information necessary to facilitate message exchange, while the body carries the business documents and attachments specific to the business collaboration. This separation allows routing and processing decisions to be made without reference to the business content.

The SBDB is OPTIONAL to allow for the situation where no business document is necessary, examples being a ping request (see section 6.3.3) where no business documents or attachments are included, and a lodgement response where the only business information required is the lodgement receipt information. If the SBDB is not provided, the BusinessDocuments element within the SBDH SHOULD NOT be provided.

If the SBDB is provided, it MUST carry at least one business document.

3.3.3. Standard Business Document Header (SBDH)

Table 6 below shows the elements that constitute the SBDH, and their use within requests and responses.

ELEMENT	PURPOSE	REQUEST	RESPONSE
sbdm:Message.Type.Text	Define the action to be taken with the information provided in the SBDB – see the Message Implementation Guide for the value required for a given interaction	MANDATORY	MANDATORY
sbdm:MessageTimestamps	Identify the date and time at which the message was generated	MANDATORY	MANDATORY
sbdm:Sender	Identify the agency who processed the request	SHOULD NOT provide	MANDATORY
sbdm:Receiver	Identify the agency to whom the request should be directed	MANDATORY	SHOULD NOT provide

sbdm:LodgementReceipt	Provide details of the results of a successful lodgement request	SHOULD NOT provide	OPTIONAL for Lodge request SHOULD NOT provide for other requests
sbdm:SoftwareInformation	Provide details of the software used to submit a request	MANDATORY	SHOULD NOT provide
sbdm:BusinessDocuments	Provide metadata regarding the business documents and attachments included in the SBDB	OPTIONAL	OPTIONAL
sbdm:MessageEvent	Provide an indication of the success or otherwise of a request, together with information on any errors that were detected	SHOULD NOT provide	MANDATORY

Table 6: Standard Business Document Header Content Model

3.3.3.1. Checking of Message.Type.Text values

It is an agency responsibility to check for Message.Type.Text values that are not recognised. Where a request is received containing a Message.Type.Text value that is not recognised, the agency will generate a response either using the same unrecognised value, or the fixed value "unknown.message.type.text". In either case, the MessageEvent SHALL contain an item of Error severity indicating the nature of the problem.

The coarse checking of Message.Type.Text values is also undertaken by Core Services in order to standardise the nature of the response provided for this condition. The Message.Type.Text value from the request message is checked against the list supported by the target agency and if it is not supported then this situation is reported via a Core Services generated SOAP fault SBR.GEN.FAULT.UNKNOWNMESSAGETYPETEXT (see section 4.5.3.1).

3.3.3.2. MessageTimestamps

The generator of a message MUST include a timestamp in the message indicating the date/time at which the message was created. In addition to the date/time value, a timestamp MUST include an indication of the entity generating the timestamp. All messages thus MUST contain at least one timestamp.

If more than one timestamp is provided, they MUST be ordered according to the order that the timestamping systems see the message i.e. BusinessEntity, SBRCore, GovernmentAgency.

A particular example of this is the message connectivity response (see section 6.3), which MUST include the timestamp provided by the BusinessEntity in the request, SHOULD include a timestamp for the date/time at which the request was seen by Core Services, and MUST include the date/time at which the response was generated by the receiving agency.

Agencies MAY include BusinessEntity and Core Services timestamps on all responses, or only on responses to message connectivity requests.

A system MAY add more than one timestamp. For instance, an agency MAY include one timestamp to record the time at which a request is received, and another to record the time at which the response is returned to Core Services.

Information in regards to the generation of date/time values may be found in section 3.5.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:Message.Timestamp.Generation.Datetime	Point in time at which the timestamp was added to the message	MANDATORY
sbdm:Message.Timestamp.GenerationSource.Code	Entity adding the timestamp Business software MUST use the value "BusinessEntity" Agency software MUST use the value "GovernmentAgency"	MANDATORY

Table 7: Message Timestamp Content Model

3.3.3.3. Sender and Receiver

Because of the synchronous nature of SBR Core web service calls, and the authentication information provided with each request as part of the security model, only the receiving party MUST be included on requests to SBR Core Services. The provided designation text is used to determine the agency to which the message MUST be routed.

Because business software does not communicate directly with the agency, the sender party SHALL be included on responses as a confirmation that the correct agency did receive the request.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:IdentificationDetails.IdentifierDesignation.Text	A unique string used to identify the party under a given scheme, whose type is provided in the Identifier Name element	MANDATORY
sbdm:IdentificationDetails.IdentifierName.Text	The type of the identifier being provided in the Designation element This field MUST always contain the value	MANDATORY

	"AgencyInternetDomainName"	

Table 8: Sender and Receiver Content Model

The table below documents the standard designation text values that MUST be used for each of the agencies within SBR Core Services.

AGENCY	DESIGNATION TEXT
Australian Tax Office	ato.gov.au
Australian Securities and Investments Commission	asic.gov.au
Australian Prudential Regulation Authority	apra.gov.au
QLD Office of State Revenue	osr.qld.gov.au
NSW Office of State Revenue	osr.nsw.gov.au
VIC Office of State Revenue	sro.vic.gov.au
TAS Office of State Revenue	sro.tas.gov.au
SA Office of State Revenue	revenuesa.sa.gov.au
WA Office of State Revenue	osr.wa.gov.au
NT Office of State Revenue	tro.nt.gov.au
ACT Office of State Revenue	revenue.act.gov.au

Table 9: Designation Text Values For SBR Core Services Agencies

3.3.3.4. LodgementReceipt

This element SHALL only be included on responses to lodgement requests, and in most cases, will be used to provide the business user with information they can use to identify the lodgement in interactions with the agency outside of SBR, such as helpdesks. The usual pattern would be for the business software to store the receipt information against the message exchange for future reference by the user.

Because both the identifier and date/time are optional, reference MUST be made to the Message Implementation Guide to determine the use of these fields, and the approach more generally to receipting, employed for a given obligation.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:Lodgement.Receipt.Identifier	A unique agency identifier for the lodgement	OPTIONAL
sbdm:Lodgement.Receipt.Datetime	The date and time at which the agency recorded the lodgement as having occurred	OPTIONAL

Table 10: Lodgement Receipt Content Model

3.3.3.5. SoftwareInformation

Every request to SBR Core Services MUST include information in regards to the software used to generate it. This information is used for service access control, fault identification, usage statistics, and service migration planning.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:OrganisationNameDetails.OrganisationalName.Text	The name of the organisation responsible for the creation of the business software.	MANDATORY
sbdm:SoftwareInformation.ProductName.Text	The name of the software used to generate the request	MANDATORY
sbdm:SoftwareInformation.ProductVersion.Text	The version of the software used to generate the request	MANDATORY

Table 11: Software Information Content Model

In order that this information is current, software developers SHOULD link the information provided to the equivalent metadata within their products.

The SBR Core Services self-certification process captures the current state of the above information for the software being registered, in order to ensure that access to SBR Core Services MAY be limited to certified software.

It is recognised, however, that software versions MAY vary over time as a result of changes unrelated to support for SBR. Comparisons made at run-time between the product name and product version information provided at the point of certification and the information provided in a request thus use a partial match, in order to provide several registration options, as listed below.

1. Re-certify each time the software information changes.

In this case, the software information provided at certification completely matches that provided at runtime. This approach would be suitable where the version information changes infrequently.

2. Where the software information can be stemmed, only record the stem as part of the certification information.

   In this approach, only the starting portion of the information provided at runtime is registered. If, for instance, the version string takes the form of Major.Minor.Patch e.g. “V6.2.P5”, and re-certification of SBR Core Services support is only expected at major releases, it would be reasonable to register the string “V6.” In a similar fashion, the product name might be registered as “Acme Accounts”, while the value provided at runtime might be “Acme Accounts Basic” or “Acme Accounts Pro”.

   The implication of this is that a change of major version would necessitate re-certification.

3. Prefix the dynamic software information with a more stable identifier specifically related to the level of SBR support within the product.

   Where the form of the information is not amenable to stemming, or where SBR functionality is modularised to the extent that it can carry its own version information, this approach allows a more stable string to be provided as part of registration, with this string most likely specifically related to SBR support within the product, optionally at the obligation level.

   It is important, however, that the dynamic version information continues to be provided at run-time to support service management functions such as fault analysis and statistics. Thus, where a developer chooses to employ this approach, the fixed stable string used during certification MUST be followed by the dynamic version information, separated by a “|” character. The “|” character MUST NOT be included in the information used during registration or as part of the dynamic software information. It SHOULD be replaced with an “_” if required.

   For example, the following version information might be provided as part of registration – “SBR 1.0” or “TFN 1.4”. At run-time, assuming the dynamic version information example above, the request would contain the string “SBR 1.0|V6.2.P5” or “TFN 1.4|V6.2.P5”. Again, the same approach MAY be applied to the product name information.

   This approach offers the most stability across product releases, but requires an additional level of version management as far as SBR support is concerned, and a more complex construction process for the software information.

A developer MAY choose to adopt any of the above approaches. It is RECOMMENDED that option b) be employed.

It should also be noted that while testing of SBR Core Services functionality within a product SHOULD normally be expected to occur prior to certification, in some circumstances a developer MAY choose to update the certification information without undertaking such testing. This is particularly the case for option a), and to a lesser extent for option b).

For instance, for option a), a version change as a result of an update unrelated to SBR functionality would imply the certification of this new version, but would not necessarily require retesting of the SBR functionality. Similarly, for the example given in option b), a change in the major version number may not imply a change in SBR capabilities. The online mechanism used for registration seeks to make the process of replicating and updating registration information as simple as possible to support these situations.

Ultimately, however, the self-assessment nature of the process means that a developer MUST satisfy themselves that their product will function according to the specifications prior to certifying a product.

3.3.3.6. BusinessDocuments

Where one or more business documents are provided in the SBDB of a message, the BusinessDocuments element MUST be included in the SBDH. It MUST contain one entry for each business document in the SBDB.

Where one or more binary attachments are provided in the SBDB, one of the business document entries in the SBDH MUST have an associated entry for the attachment. An attachment MUST be accompanied by a business document. The Message Implementation Guide will identify when attachments are required as part of an obligation, and whether the obligation requires the provision of a filename or description for attachments.

Each business document and attachment in the SBDB is assigned a sequence number. Business documents and attachments MUST be separately numbered. Each numbering scheme MUST start at one and increment by 1 for each item. The number is used to correlate SBDB entries with the equivalent entry in the SBDH.

If required by an interaction, the Message Implementation Guide will document the use of the business generated and government generated identifiers.

The Message Implementation Guide also identifies the AU Reports schemas considered valid for the messages of a given obligation.

Information in regards to the generation of date/time values may be found in section 3.5.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:BusinessDocument.Sequence.Number	The unique identifier of a business document within the message	MANDATORY
sbdm:BusinessDocument.Creation.Datetime	The date/time at which the business document was created	MANDATORY
sbdm:BusinessDocument.ValidationUniformResourceIdentifier.Text	The URI that identifies the specification by which the business document may be validated. For XBRL payloads, the value of this element MUST match that of the href attribute of the schemaRef element within the XBRL business document. The value will be an absolute URL (see below) pointing to the report schema and starts with	MANDATORY

	"http://sbr.gov.au/taxonomy". For XML payloads the value of this element MUST be "xml" (case insensitive).	
sbdm:BusinessDocument.BusinessGeneratedIdentifier.Text	Specified by the MIG Where the value provided in this element is returned via an element in a response payload instance, the restrictions imposed by the definition of the taxonomy element used (e.g. shorter length) MUST be consulted prior to generation of a value for this element.	OPTIONAL
sbdm:BusinessDocument.GovernmentGeneratedIdentifier.Text	Specified by the MIG	OPTIONAL

Table 12: Business Document Content Model

SBR AU report schemas for XBRL payloads are published at the absolute URL "http://sbr.gov.au/taxonomy". Under this point, the path to a specific schema mirrors that described in the "SBR AU Naming Convention" document relative to (but not including) the top level sbr_au folder. An example URL is shown below;

http://sbr.gov.au/taxonomy/sbr_au_reports/asic/f388/f388_0001/f388.0001.lodge.request.02.00.report.xsd

Because the on-line availability of the SBR AU report schemas cannot be guaranteed, alternative mechanisms, such as caching, SHOULD be used by client software where the run-time availability of taxonomies is required.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:Message.Attachment.SequenceNumber	The unique identifier of an attachment within the message	MANDATORY
sbdm:Message.Attachment.FileName.Text	The name of the file from which the attachment was sourced. No path information should be provided.	Specified by the MIG
sbdm:Message.Attachment.Description.Text	A description of the contents of the attachment	Specified by the MIG

Table 13: Message Attachment Content Model

3.3.3.7. MessageEvent

The MessageEvent element MUST be provided on every response in order to provide a standard way in which software can determine the outcome of a request. It is covered in more detail in section 4.6.

3.3.4. Standard Business Document Body (SBDB)

Business document and attachment contents are carried in the SBDB.

If included in the SBDM, the SBDB MUST contain at least one business document.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:BusinessDocumentInstances	Container for the contents of the business documents provided in the message	MANDATORY
sbdm:AttachmentInstances	Container for the contents of the attachments provided in the message	OPTIONAL

Table 14: Standard Business Document Body Content Model

3.3.4.1. BusinessDocumentInstance

Each document MUST be accompanied by the sequence number by which it is identified within the message. The SBDM schema allows any well-formed XML as the content of a business document.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:BusinessDocument.Sequence.Number	The unique identifier of a business document within the message	MANDATORY
sbdm:BusinessDocument.Instance.Text	A container element for the XML contents of the business document	MANDATORY

Table 15: Business Document Instance Content Model

3.3.4.2. AttachmentInstance

Each attachment MUST be accompanied by the sequence number by which it is identified within the message.

Each attachment MUST also carry an xmime:contentType attribute, indicating the MIME type of the attachment, in line with the MTOM recommendation.

The Message Implementation Guide will provide details of the content types accepted as part of a given interaction.

ELEMENT	PURPOSE	OPTIONALITY
sbdm:Message.Attachment.Sequence.Number	The unique identifier of an attachment within the message	MANDATORY
sbdm:Message.Attachment.Instance.BinaryObject	The base64 encoded value of the attachment.	MANDATORY
xmime:contentType (attribute)	The MIME type of the attachment contents	MANDATORY

Table 16: Attachment Instance Content Model

3.4. SOAP FAULTS

The WSDL specification allows service definitions to include information on the format of the “Detail” element of any SOAP faults they can return. Whilst it is considered best practice for Detail information to be defined in the WSDL, given the synchronous nature of the Core Services design, it is not expected that there will be a need for information in the “Detail” field. Therefore the SOAP faults SHALL return the standard fields defined in the SOAP specification and the faults themselves SHALL NOT be defined in the WSDLs.

The one condition under which SBR Core Services MAY return a SOAP fault with a detail element is in regards to the condition of system unavailability (see section 4.5.3.2). It is expected that software developers will need to manually code for this condition, rather than relying on code automatically generated by their framework.

3.5. 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 section 3.3.3.2) 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.

3.6. 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 Core Services and between Core Services and the relevant agency, as well as processing delays at Core Services 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.

While most requests are expected to be processed within the order of 10 seconds, Core Services 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 Core Services expected as a result of the capacity of their client's internet connections and the nature of the obligations being supported.

Any timing considerations unique to a particular obligation SHALL be documented in the Message Implementation Guide.

3.7. XBRL INSTANCES

This section highlights a number of aspects common to the construction and processing of all XBRL instances for use within the SBR program.

3.7.1. Ordering of Context, Unit and Fact Elements

The XBRL 2.1 specification places no restrictions on the ordering of context, unit or top-level facts within an XBRL instance. Thus, processing of XBRL instances within SBR MUST NOT assume a particular document order for these elements.

As an example, it MUST NOT be assumed that context elements will occur before the facts which reference them.

3.7.2. Semantic Meaning of Context IDs and Unit IDs

While Message Implementation Guides provide RECOMMENDATIONS as to the values to be used for context IDs and Unit IDS, processing of XBRL instances within SBR MUST NOT assume specific ID values will be present within an instance. The association between facts and their associated context or unit MUST be performed dynamically.

As an example, in determining which facts have been provided in a particular context, processing MUST NOT assume the context has a particular ID value, and directly access those facts whose context reference uses this value. Instead, it is necessary to examine the contents of the provided context elements to ascertain the ID to use in locating the desired facts.

3.7.3. Redundant Contexts

The XBRL 2.1 specification allows the possibility of multiple redundant context elements within a single XBRL instance, that is contexts that have the same contents (entity, period, dimensions etc) but different IDs.

Message Implementation Guides define the contexts to be used in each SBR XBRL instance. Reference MUST be made to these documents to determine if redundant context are permissible for a given SBR report. To ensure interoperability, applications SHOULD avoid the generation of instances containing redundant contexts as a general principle.

3.7.4. Namespace Prefixes Used On Elements

The XBRL 2.1 specification is underpinned by the XML Namespaces specification. XBRL instances within SBR MUST thus conform to the requirements of this specification. As examples, processing of XBRL instances in SBR MUST support the use of default namespaces, and MUST NOT assume that specific namespace prefix values will be used when defining XML elements within a given SBR report.

Message Implementation Guides MAY adopt conventions for namespace prefixes for reasons of clarity or ease of update when documenting the facts expected within a given SBR report. However, conformance to these conventions MUST not be assumed when processing XBRL instances within SBR.

3.7.5. Namespace Prefixes Used in Context Element Dimension Definitions

The XBRL Dimensions 1.0 specification defines that the contents of the dimension attribute on an xbrldi:explicitMember, and the contents of the xbrldi:explicitMember itself must be QNames. The prefix on the QName is intended to be dereferenced via the in-scope namespace definitions to determine the namespace of the value provided. As such, the prefix is transitory and the value used MAY be arbitrary.

Message Implementation Guides MAY further constrain the prefixes used in the situations above to be explicit values. Reference MUST be made to these documents to determine if specific prefix values MUST be used for a given SBR report.

3.7.6. Monetary Units

The XBRL 2.1 specification requires that the QNames used in unit definitions for monetary values MUST use ISO4217 currency designations for the local part, and MUST use a namespace of “http://www.xbrl.org/2003/iso4217”. Unit definitions for monetary currencies in XBRL instances within SBR MUST conform to these requirements. In particular, amounts representing Australian dollars MUST be associated with a unit definition that uses a currency designation of “AUD”.

The error code SBR.GEN.GEN.24 (see section 4.6.1.1) is provided to indicate the condition where the provided currency unit does not conform to the specification requirements. Error code SBR.GEN.GEN.22 is provided to indicate that the currency unit for a fact is valid according to ISO 4217, but was not the unit expected by the receiving agency.

Unless otherwise stated in the Message Implementation Guide, all monetary amounts in XBRL instance must be expressed in Australian dollars.

3.7.7. Measurement Accuracy

The XBRL 2.1 specification requires that each numeric item (apart from those whose value is a fraction) carry either a precision or decimals attribute allowing the creator of an XBRL instance to provide a statement of the accuracy of the provided value.

Unless otherwise stated in the relevant Message Implementation Guide,

when producing XBRL instances within SBR

1. non-financial numeric values, such as counts, SHOULD be provided with a value of ”0” for the decimals attribute.

2. financial amounts accurate to the dollar SHOULD be provided with a value of “0” for the decimals attribute.

3. financial amounts accurate to the cent SHOULD be provided with a value of “2” for the decimals attribute.

when consuming XBRL instances within SBR

1. any digits considered to be insignificant SHOULD be replaced with zeros.

3.7.8. XBRL Element Attributes

Some SBR reports (e.g. ATO Activity Statement) have a single supporting XBRL taxonomy but have context dependent rules about which elements need to be populated. In the Activity Statement example, there are structures in place for reporting a number of different tax obligations such as GST, FBT, PAYG, etc. But not every business is required to report on every obligation in every period. An element that is MANDATORY for one business MAY not be required for another.

To assist in creating a valid lodge report, the SBR prefill service is often used to return an empty "template" for a lodge. For example, the ATO Activity Statement as.0001.prefill.response message SHALL contain the data elements that are required to be populated to complete a valid lodgement (with some data elements pre-populated by the ATO).

There are three optional xml attributes that MAY be present on a prefill template and these are used to guide the expected behaviour of the software package:

1. **isEditable** – is a boolean flag (xsd boolean data type) to indicate whether the content of an element SHOULD be edited by the user of a web service.

2. **isVisible** - is a boolean flag (xsd boolean data type) to indicate whether the content of an element SHOULD be displayed to a user.

3. **xsi:nil** – is a Boolean flag (xsd Boolean data type) to indicate that a MANDATORY field MAY have a nil value.

The "isEditable" and "isVisible" attributes MUST be namespace qualified, and MUST use the fixed value "http://sbr.gov.au/fdtn/sbr.02.00.tech" as the namespace URI. The xsi:nil attribute is a standard attribute defined by the XML schema specification http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

The use of these attributes is specific to the interaction employing them. The Message Implementation Guide thus MUST be consulted to determine whether the attributes will be present on responses and what defaults (if any) MUST be assumed.

3.8. XML INSTANCES

XML instances MUST comply with the relevant XML schema and schemas in turn MUST comply with xml schema specification http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

4. ERROR MANAGEMENT

4.1. OVERVIEW

This chapter details the approach to be taken to the handling of errors and exception conditions associated with the submission of requests to agencies via the SBR Core Services channel. It divides the conditions that MUST be addressed into four broad areas;

1. user errors
2. client software errors
3. transport exceptions and
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 Core Services to assist in this process.

On the other hand, this document provided detailed information on the reporting and management of errors in the latter two areas. In a nutshell, transport exceptions SHALL be handled using the framework provided by the SOAP 1.2 recommendation, while business events SHALL be handled via structures within the Standard Business Document Header structure.

4.2. CONTEXT

Figure 4 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 Core Services. The diagram also shows that there are a number of human driven setup activities that business users will need to undertake prior to using the electronic channel provided by SBR Core Services. Client software will need to accommodate the possibility that users have not undertaken these steps prior to attempting to submit reports via SBR Core Services.

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

1. User errors (shown in green)

- Triangle 1 – the user has not obtained an SBR credential
- Triangle 2 – the user has not configured their internet connection correctly
- Triangle 13 – the user has not undertaken the appropriate registration or authorisation procedures required by the agency
- Triangles 14, 15 – the information content (provided by the user) of the payloads is not correct

2. Client software problems (shown in yellow)

- Triangle 3 – there are defects in the client software
- Triangles 6, 7, 8, 11, 12 – the messages generated by the client software do not conform to the required SBR Core Services standards

- Triangles 14, 15 – the payload is not correctly formed (XBRL type is used as an example)

3. Unavailability of components of SBR Core Services (shown in blue)

- Triangle 5, 10, 19, 20 – connectivity issues within the systems comprising SBR, or component unavailability due to scheduled maintenance
- Triangle 16 – agency processing systems are not available

4. Errors internal to SBR Core Services systems (shown in red)

- Triangle 4, 9 and 18 – there are defects in SBR Core Services systems
- Triangle 11, 12 – problems are detected in the modifications made by core to messages provided by business
- Triangle 17 – Internal problems within the agency processing systems

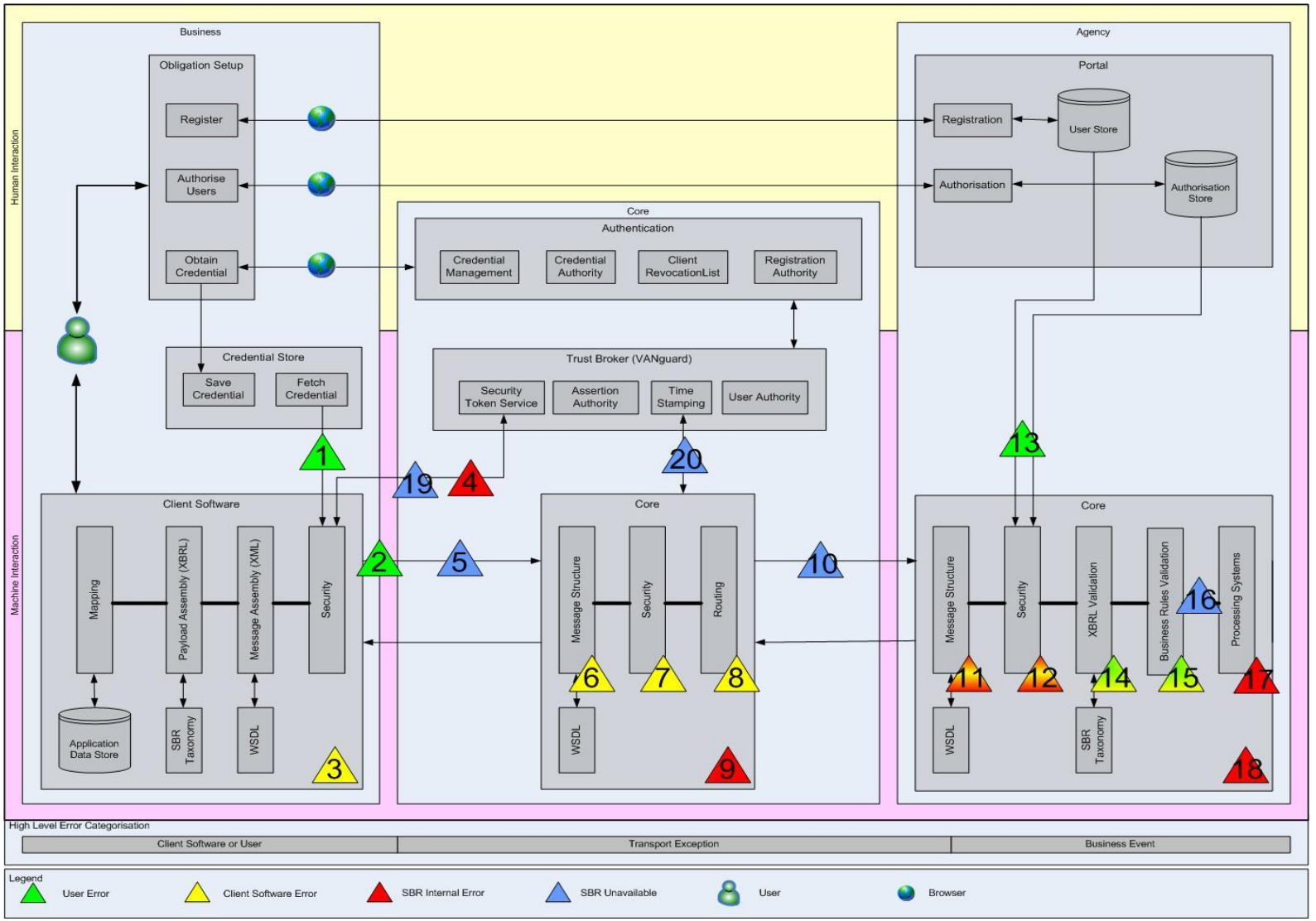

Figure 4: Sources of Errors In SBR Core Services platform

4.3. HIGH LEVEL CATEGORISATION OF ERROR CONDITIONS

This document asserts that the various errors conditions described in section 4.2 can be placed into four high level areas, as shown at the bottom of figure 4.

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, the associated documentation will indicate the nature of the possible errors. This category is outside the scope of this document. It should be noted that, in general, errors associated with invalid user credentials will be identified as part of the use of the SBR provided APIs for Core Services.

The remaining conditions result from the client software interacting with SBR Core Services, 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 SOAP 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 SOAP messages, all remaining error conditions will be as a result of business activities. 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.

Business events SHALL be communicated via the "MessageEvent" structure within the Standard Business Document Header. A MessageEvent MUST contain one or more "MessageEventItem"s, with each item representing a single business event. It is thus possible to communicate multiple events within a single business response.

4.4. PRINCIPLES

The following principles have guided the development of the error approach outlined in this document.

4.4.1. P.01 Provide certainty as to the action to be taken in regard to an error

Reference	P.01
Principle	Provide certainty as to the action to be taken in regard to an error
Rationale/Motivation	1. Minimise the number of conditions for which software developers must code 2. Improve consistency of error handling across software products
Implications	1. If there is no difference in the action expected from business or their software as a result of two different error conditions, the errors need to be grouped/categorised to make this clear.

4.4.2. P.02 Provide sufficient detail to allow appropriate action to be taken

Reference	P.02
Principle	Provide sufficient detail to allow appropriate action to be taken
Rationale/Motivation	1. The action to be taken by business as a result of an error should be unambiguous 2. Improve the user experience by minimising resubmissions due to misunderstanding of error conditions
Implications	1. Short and detailed descriptions of errors must be provided by agencies 2. Error conditions must be sufficiently fine-grained to allow differentiation between actions to be taken 3. Where appropriate, include agency specific error information in the error detail for follow up with agencies

4.4.3. P.03 Provide consistency in the errors to be handled

Reference	P.03
Principle	Provide consistency in the errors to be handled
Rationale/Motivation	1. Provide consistency of error reporting across various obligations 2. Minimise implementation burden on software developers
Implications	1. Agencies need to harmonise the error information they produce 2. Agencies need to map, where possible, their individual error codes to a harmonised SBR set

4.5. TRANSPORT EXCEPTIONS

4.5.1. SOAP Processing Model

For the Core Services platform the SBR program has adopted the W3C SOAP 1.2 recommendation, together with an HTTP based transport, as the basis for the on-the-wire format of messages to be exchanged between client software and agencies. This implies the solution SHALL be consistent with the SOAP processing model, and SHALL leverage the SOAP fault mechanism as the primary way in which transport related exceptions will be communicated.

It is recommended that the reader familiarise themselves with the SOAP specification prior to reading the remainder of this document.

The top portion of figure 5 below shows how the SOAP 1.2 processing model maps to the SBR solution, the solution being as shown in the bottom half of figure 4. The lower half of figure 5 provides examples of the various conditions that will result in the returning of a SOAP fault to the client software.

The software generating requests to government takes the role of the initial sender, while the agency processing system takes the role of ultimate receiver. Core services acts as a forwarding intermediary. At both the business and agency, a gateway may be optionally employed to mediate between the internal architecture and that used by SBR Core Services. Any such gateway takes the role of a forwarding intermediary, and logically maps to the intermediate processing stages shown in figure 4.

Because SBR Core Services uses the Request/Response Message Exchange Pattern (MEP) for all exchanges between business and SBR Core Services, a successful exchange will see a valid SOAP message generated by the initial sender, forwarded to the relevant agency via the chain of SOAP intermediaries, actioned by the agency processing system, and a valid response message generated and returned to the initial sender via the same chain of SOAP intermediaries.

The SOAP recommendation specifies that, by their nature, SOAP intermediaries do not play the role of ultimate receiver, and thus SHOULD NOT process the body of a SOAP message. Any errors MUST be flagged via the generation of SOAP faults. In general, SBR Core Services complies with this, in particular in relation to the generation of responses.

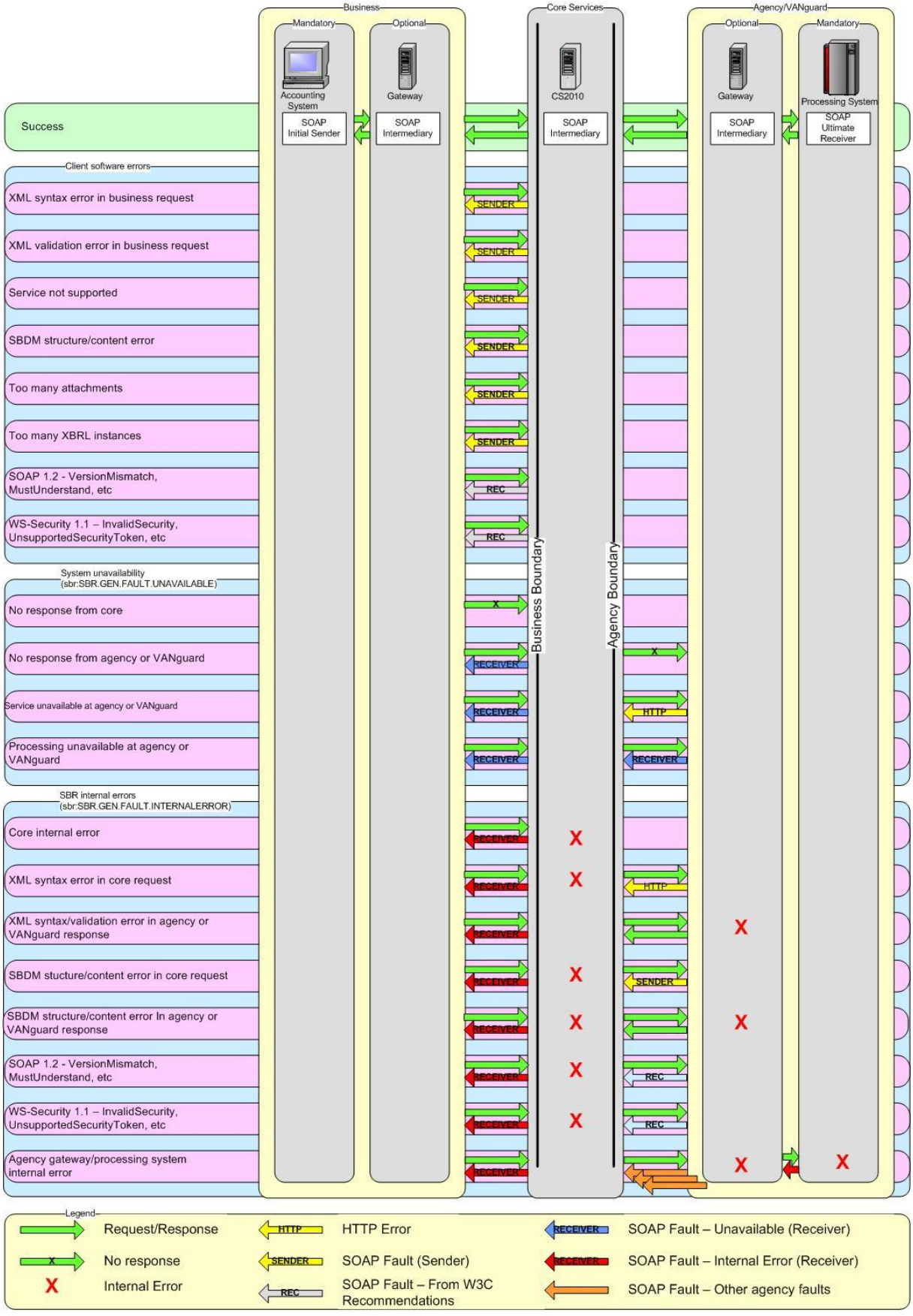

Figure 5: SOAP Fault Processing

There MAY, however, be a blurring of responsibilities between gateway and processing systems. A case in point is the error condition in which the taxonomy(ies) on which the payload(s) of the message are based are not supported by the receiving agency. Based on a strict interpretation of the SOAP recommendation, if detected by the agency gateway, this error condition SHOULD be indicated via a SOAP fault. However, in order to allow a strict separation of the generic messaging infrastructure and issues surrounding the versioning of taxonomies, this condition is considered to be detected by the processing system, and hence SHOULD be indicated via a business event rather than a SOAP fault.

4.5.2. Use of SOAP Fault fields

The SOAP 1.2 recommendation defines a number of standard fields within a SOAP Fault.

4.5.2.1. Code Element

SBR Core Services SHALL conform to the SOAP 1.2 recommendation and use only the standard values it defines for the Code element. As shown in figure 5, most faults will use a code of “env:Sender” or “env:Receiver”.

As described in the recommendation, a code of “env:Sender” carries with it an expectation that the sender will change/correct the request, whereas “env:Receiver” implies a downstream problem with no change needed to the request. Thus figure 5 shows some apparent duplication of error conditions (e.g. SBDM structure/content error), the differentiating factor being the source of the problem i.e. client software or SBR Core Services component.

4.5.2.2. Subcode Element

SBR Core Services SHALL use subcodes in order to provide the detail necessary as per P.02. More than one level of subcode MAY be necessary in order to establish the high level action required in each case as per P.01, while providing the necessary detail as per P.02.

Subcodes defined by SBR Core Services SHALL use the namespace “http://sbr.gov.au/comn/core.02.data”. Subsequent sections use a prefix of “sbr” to correspond to this namespace, in line with section 1.5.

These subcodes SHALL follow the naming convention defined in section 4.6.1.1, and, for faults from Core Services, SHALL use the Jurisdiction.Agency.Function value of “SBR.GEN.FAULT”. In line with the approach to fault codes employed by SOAP related specifications, the identifiers used SHALL be short mnemonics of the condition represented. An example is shown below for the case where too many payload instances have been included in a request;

sbr:SBR.GEN.FAULT.TOOMANYINSTANCES

SBR Core Services is employing the WS-Security 1.1 and SAML Token Profile 1.1 recommendations from OASIS. The subcodes itemised in these recommendations in sections 12 and 3.6 respectively SHALL be used to report security related exceptions. These subcodes do not provide a great deal of granularity, however, typically to minimise the information available to a party undertaking a cryptographic attack. Thus more detailed subcodes SHALL be provided in testing environments provided by SBR Core Services in order to assist the rapid resolution of issues that arise during execution of test scenarios.

4.5.2.3. Reason Element

The reason element SHALL describe the specifics of the particular error condition, and SHALL thus reflect the finest granularity of subcode provided in the fault.

4.5.2.4. Node Element

In line with the SOAP 1.2 recommendation, the following URIs SHALL be employed in SBR Core Services.

NODE	URI
Core Services	http://sbr.gov.au/comn/node/core
Agency	http://sbr.gov.au/comn/node/{agency designation text} e.g. http://sbr.gov.au/comn/node/ato/gov/au
VANguard	http://sbr.gov.au/comn/node/vanguard

Table 17: SOAP Fault Node element URI values

It should be noted that when including the agency designation text (see 3.3.3.3) into the element value, "." should be replaced with "/" as per the example above. Faults generated by agencies SHOULD include a Node element.

4.5.2.5. Role Element

Role elements SHOULD not be included in SBR Core Services SOAP faults.

4.5.2.6. Detail Element

It is not intended that SOAP faults provided to client software in the production environment carry any additional information over and above the codes and reason elements already described. Thus they SHALL be provided without a Detail element.

For SOAP faults generated by agencies, it is intended that Core Services will log any contents of Detail elements, but SHALL NOT include this information in the SOAP fault provided to client software – see 4.5.3.4. This SHALL prevent the leakage of sensitive production information.

The passing of the Detail element through to client software SHALL, however, be configurable in Core Services. Thus it SHALL be possible in testing environments provided by SBR Core Services to allow this information to be passed through, in order to assist the rapid resolution of issues that arise during execution of test scenarios.

4.5.3. Exception Conditions

Figure 5 documents a range of conditions that may occur while exchanging messages with SBR Core Services. In line with principle P.01, however, they are grouped into three categories – client software

errors, SBR Core Services unavailability and SBR Core Services internal errors. At the highest level, these categories identify the distinct actions needed to be taken on receipt of a SOAP fault.

4.5.3.1. Client software errors

All errors in this category result from a defect within the software used by business causing the generation of invalid requests. It is intended that the SBR Core Services 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 Core Services, with the remainder being detected by agencies.

In line with the SOAP 1.2 recommendation, errors in this category SHOULD mostly be reported with a SOAP fault. Such faults SHALL use a code value of “env:Sender”. The only exceptions to this are conditions called out by the W3C recommendation itself such as “env:VersionMismatch”. It SHOULD be noted that Core Services will not be supporting V1.1 of the SOAP recommendation.

Where an agency generates a SOAP fault as a result of a client software error, a code value of “env:Sender” SHALL be used. A fault from an agency with a code value of “env:Sender” SHALL pass unaltered through Core Services and be provided as is to the client software. Thus the same client logic will be able to handle such faults, regardless of whether they are generated by Core Services or an agency. It should be noted that the value of the Node element MAY be used to provide an indication as to the source of the fault.

It is expected that if errors in this category occur during the operation of a software package, the business user would be expected to contact their software provider for rectification advice.

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

SUBCODE	REASON	DESCRIPTION/COMMENT
sbr:SBR.GEN.FAULT.MALFORMEDXML	The request was not well formed XML.	The request is not well formed XML, as documented in the XML specification.
sbr:SBR.GEN.FAULT.INVALIDXML	The request does not validate against the service XML Schema	The request does not validate against the XML Schema for the service, which is defined as part of the WSDL for the service.
sbr:SBR.GEN.FAULT.TOOMANYINSTANCES	Payload instance limit exceeded	The request contains more payload instances than are allowed by the agency.
sbr:SBR.GEN.FAULT.TOOMANYATTACHMENTS	Attachment limit exceeded	The request contains more attachments than are allowed by the agency.
sbr:SBR.GEN.FAULT.TOOMANYDOCUMENTS	Payload instance limit and attachment limit exceeded	Both the agency limit on payload instances and on attachments have been exceeded.

sbr:SBR.GEN.FAULT.TOOBIG	Request size limit exceeded	SBR Core Services limit on maximum request size is set to cater for all obligation requirements for the program.
sbr:SBR.GEN.FAULT.ATTACHMENTERROR	The request message attachment could not be processed.	The request message contained an attachment that could not be processed. Note: Soap with Attachments is not supported by SBR Core Services.
sbr: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:SBR.GEN.FAULT.SOFTWAREBLOCKED	The software used to generate this request has been blocked from submitting to SBR Core Services	SBR Core Services has the capability to block requests from particular client software. This fault is generated in the situation where this capability is enabled.
sbr:SBR.GEN.FAULT.SOFTWARENOTREGISTERED	The software used to generate this request has not been registered with SBR Core Services	SBR Core Services 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:SBR.GEN.FAULT.INVALIDSBDM	The structure of the request does not conform to the requirements documented in the SBR Core Web Services Implementation Guide	This document defines a number of rules regarding request structure, over and above those enforced by the WSDL schemas. This fault is generated when a violation occurs of these rules, an example being the absence of an element that is MANDATORY on request messages but not on response messages.
sbr:SBR.GEN.FAULT.UNKNOWNMESSAGETYPETEXT	The message type text of the request is not known by the receiving agency	The Message Implementation Guide relevant to the obligation being implemented by the message exchange provides details of the expected message type text values.
sbr:SBR.GEN.FAULT.UNKNOWNVALIDATIONURI	One or more of the business documents in the request uses a	The Validation URIs included in requests must obey the format rules specified in this document, as well as

	validation URI that is not known by the receiving agency	matching the values documented in the Message Implementation Guide relevant to the obligation being implemented by the message exchange. This error is applicable only to XBRL payloads.
sbr:SBR.GEN.FAULT.MISMATCHEDPAYLOADS	Mismatched payload types were provided	First Validation URI element is used to recognise the payload format and then it cross-checked against other Validation URI elements for consistency. All payloads in the message should be of the same type. The error is raised when business documents in the request message have payloads of different types (e.g. XBRL and XML)
sbr:SBR.GEN.FAULT.UNSUPPORTEDPAYLOAD	Unsupported payload type was provided	The error is raised if the value of the first Validation URI element is not matched against the list of supported payload types. See rules applicable to the Validation URI element in XBRL payloads above.
sbr:SBR.GEN.FAULT.UNKNOWNPAYLOADTYPE	The payload type of the request is not known to be supported for the message type text used	The error is raised when the payload type of the request message is one of the supported (e.g. XBRL, XML) but target agency doesn't support it for the specified Message.Type.Text.
wsse:SecurityTokenUnavailable	No security token was provided	A VANguard supplied security token must be included in the request.
wsse:FailedCheck	The provided signatures or encryption were invalid	This error will occur as a result if - the security token does not decrypt successfully[1] - the session signature was not valid[1] - the SBDM signature was not valid[1]
wsse:InvalidSecurityToken	An invalid security token was provided	This error covers a number of circumstances related to the security token :- - The signature of the provided token was invalid - Core Services is not the audience - the token has expired - the token did not include a session

		key[1] - the token did not include the required set of mandatory claims[1] - the session key could not be decrypted[1] - the certificate used to sign the SBDM was not the same as that used to obtain the security token[1] A current VANguard supplied security token must be included unaltered in the request. The certificate presented to obtain this token should be used to sign the SBDM.

[1] In test, this condition will be indicated via an additional subcode - see the table of security subcodes below

Table 18: SOAP Fault Subcodes for client software errors

In testing environments, the following subcodes will also be provided, where relevant, in addition to the WS-Security subcode.

SUBCODE	REASON	DESCRIPTION/COMMENT
sbr:SBR.GEN.FAULT.CANTDECRYPTTOKEN	The security token could not be decrypted	See Reason
sbr:SBR.GEN.FAULT.SESSIONKEYMISSING	No session key was provided	See Reason
sbr:SBR.GEN.FAULT.CANTDECRYPTSESSIONKEY	The session key could not be decrypted	See Reason
sbr:SBR.GEN.FAULT.INVALIDSESSIONSIGNATURE	The session based signature is invalid	See Reason
sbr:SBR.GEN.FAULT.INVALIDSBDMSIGNATURE	The SBDM signature is invalid	See Reason
sbr:SBR.GEN.FAULT.INVALIDTOKENSIGNATURE	The security token signature is invalid.	Provided the token provided by VANguard is passed through unaltered to SBR Core Services, this error should not occur under normal circumstances.
sbr:SBR.GEN.FAULT.MISSINGCLAIMS	The security token did not include all of the mandatory information	The token provided by VANguard did not include all of the mandatory claims expected by SBR.
sbr:SBR.GEN.FAULT.CERTIFICATEMISMATCH	The security token certificate does not match the SBDM signature	The same certificate must be used to obtain the security token and to sign the SBDM

Table 19: SOAP Fault Security Subcodes (Test Environments Only)

The figures below show examples of the faults that will be generated as a result of client software errors. Line wraps within text are for presentational purposes only.

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
        <env:Subcode>
              <env:Value>sbr:SBR.GEN.FAULT.MALFORMEDXML</env:Value>
        </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en"> The request was not well formed XML</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 6: SOAP Fault indicating XML is not well formed

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.INVALIDXML</env:Value>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">The request does not validate against the service
XML schema</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 7: SOAP Fault indicating XML schema validation failure

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.TOOMANYINSTANCES</env:Value>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">Payload instance limit exceeded</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 8: SOAP Fault indicating too many payload instances

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>wsse:SecurityTokenUnavailable</env:Value>
```

```
      </env:Subcode>
   </env:Code>
   <env:Reason>
      <env:Text xml:lang="en">No security token was provided</env:Text>
   </env:Reason>
   <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 9: SOAP Fault indicating a missing security token

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>wsse:InvalidSecurityToken</env:Value>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">An invalid security token was provided</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 10: SOAP Fault indicating an invalid security token (production)

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>wsse:InvalidSecurityToken</env:Value>
        <env:Subcode>
          <env:Value>sbr:SBR.GEN.FAULT.CERTIFICATEMISMATCH</env:Value>
        </env:Subcode>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">The security token certificate does not match the
SBDM signature</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 11: SOAP Fault indicating an invalid security token (test)

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>wsse:FailedCheck</env:Value>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">The provided signatures or encryption were
invalid</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 12: SOAP Fault indicating an invalid digital signature (production)

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>wsse:FailedCheck</env:Value>
        <env:Subcode>
               <env:Value>sbr:SBR.GEN.FAULT.INVALIDSBDMSIGNATURE</env:Value>
        </env:Subcode>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">The SBDM signature is invalid</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 13: SOAP Fault indicating an invalid digital signature (test)

```
<env:Fault >
  <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.UNKNOWNSERVICE</env:Value>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">Unknown agency or service</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 14: SOAP Fault indicating unknown agency or service not supported by an agency

4.5.3.2. SBR Core Services unavailability

A reality of the SBR Core Services platform, 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 reason text SHOULD contain the date and time (including timezone) after which the service is expected to be available again. In addition, in order to allow automatic requeuing of the request, the fault detail MAY contain the equivalent information in a machine consumable format (see example below). The presence of this information SHOULD be checked for, and where possible, used to requeue the request. The human readable and machine consumable times MAY not align exactly, in order to allow the spreading of requests over time after the service resumes.

Regardless of the approach taken by the software, 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.

With the exception of the condition where Core Services does not respond at all to requests, all conditions resulting in unavailability of SBR Core Services SHALL be reported to client software via a SOAP fault with a code value of "env:Receiver" and a subcode of "sbr:SBR.GEN.FAULT.UNAVAILABLE". The way in which a timeout condition on a request to Core Services will be flagged is dependent on the platform used by the client software.

Where a SOAP fault is generated, the reason message SHALL reflect the specific condition that resulted in the unavailability of the service. An equivalent subcode SHALL also be provided, the possible values being provided below.

SOAP faults generated by agencies that carry a subcode of sbr:SBR.GEN.FAULT.UNAVAILABLE SHALL pass unaltered through Core Services and be provided as is to the client software.

SUBCODE	REASON	DESCRIPTION/COMMENT
sbr:SBR.GEN.FAULT.CANTCONNECTTOAGENCY	A connection could not be established to an agency	SBR Core Services was not able to initiate a connection to the agency.
sbr:SBR.GEN.FAULT.HTTPERRORFROMAGENCY	The agency gateway could not process the request.	The HTTP headers included in the agency response indicated an error. This is often caused by the service being unavailable at the agency gateway.
sbr:SBR.GEN.FAULT.AGENCYNOTRESPONDING	The connection with the agency timed out	SBR Core Services was able to connect to the agency, but did not receive a response to the submitted request.
sbr:SBR.GEN.FAULT.NOAGENCYPROCESSING	The agency processing system is unavailable	The agency gateway is accepting requests, but the backend processing system is not currently available.

Table 20: SOAP Fault Subcodes for SBR Core Services unavailability

The figures below show examples of the faults that will be generated as a result of unavailability of SBR Core Services components.

```
<env:Fault >
  <env:Code>
     <env:Value>env:Receiver</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.UNAVAILABLE</env:Value>
        <env:Subcode>
               <env:Value>sbr:SBR.GEN.FAULT.AGENCYNOTRESPONDING</env:Value>
        </env:Subcode>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">The connection to the agency timed out</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 15: SOAP Fault indicating agency is unavailable

```
<env:Fault >
  <env:Code>
     <env:Value>env:Receiver</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.UNAVAILABLE</env:Value>
        <env:Subcode>
               <env:Value>sbr:SBR.GEN.FAULT.NOAGENCYPROCESSING</env:Value>
        </env:Subcode>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">We are unable to process your request at this
time. Please try again after 30-06-2009 09:00:00 EST</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/asic/gov/au</env:Node>
  <env:Detail>
     <sbr:FaultDetail>
        <sbr:AvailableAfter>2009-06-29T23:00:00Z</sbr:AvailableAter>
     </sbr:FaultDetail>
  </env:Detail>
</env:Fault>
```

Figure 16: SOAP Fault indicating agency processing system is unavailable

The figure above shows an example of a fault indicating unavailability generated by an agency, as indicated by the value of the Node element. Such a fault would have passed unaltered through Core Services.

4.5.3.3. SBR Core Services internal errors

As with the client software, core 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 Core Services exception handling regime allow for their possibility at runtime.

As figure 5 shows, there is a broad range of errors that may occur internal to SBR Core Services. They have been grouped under a single subcode, however, because the actions

needed to be taken by business are the same – requeue 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.

All error conditions in this category SHALL be reported to client software via a SOAP fault with a code value of “env:Receiver” and a subcode of “sbr:SBR.GEN.FAULT.INTERNALERROR”.

The reason message SHALL reflect the specific condition that resulted in the internal error. An equivalent subcode SHALL also be provided as per the table below.

SUBCODE	REASON	DESCRIPTION/COMMENT
sbr:SBR.GEN.FAULT.MALFORMEDXMLINCORE	Malformed XML encountered during Core Services processing	Malformed XML was generated within Core Services during message processing.
sbr:SBR.GEN.FAULT.GENERALERRORINCORE	An unhandled error occurred within SBR Core Services	This error is generated whenever an unhandled error is detected in Core Services
Sbr:SBR.GEN.FAULT.CONFIGURATIONERRORINCORE	Configuration Error within SBR Core Services.	An error occurred while accessing Core configuration data
Sbr:SBR.GEN.FAULT.IDENTIFIERERRORINCORE	SBR Core Services was unable to retrieve a unique identifier for this request	SBR Core Services was unable to retrieve the unique message identifier for the request
sbr:SBR.GEN.FAULT.MALFORMEDXMLFROMAGENCY	The response was not well formed XML	The agency XML is not well formed according to the XML specification.
sbr:SBR.GEN.FAULT.INVALIDXMLFROMAGENCY	The response does not validate against the service XML Schema	The agency response does not validate against the XML Schema for the service, which is defined as part of the WSDL for the service
Sbr:SBR.GEN.FAULT.FAULTFROMAGENCY	Reason text in fault will be copied from the fault received from the agency as per section 4.5.3.4	SBR Core Services received a SOAP fault from the agency, which was not related to system availability. (see section 4.5.3.4 for further detail)
sbr:GEN.FAULT.AGENCYATTACHMENTERROR	The agency response message attachment could not be processed.	The agency response message contained an attachment that could not be processed.
wsse:FailedCheck	The agency provided signature was invalid	This error will occur as a result the agency signature not being valid.

Table 21: SOAP Fault Subcodes for SBR Core Services Internal Errors

The figure below shows an example of the faults that SHALL be generated as a result of internal errors in SBR Core Services components.

```
<env:Fault >
  <env:Code>
     <env:Value>env:Receiver</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.INTERNALERROR</env:Value>
        <env:Subcode>
               <env:Value>sbr:SBR.GEN.FAULT.INVALIDXMLFROMAGENCY</env:Value>
        </env:Subcode>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">An error was detected in the response from the
agency</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/core</env:Node>
</env:Fault>
```

Figure 17: SOAP Fault indicating invalid XML from an agency

4.5.3.4. Agency internal errors

Agencies are employing a variety of frameworks to implement their gateways and processing systems. The nature of these frameworks means that, at times, it will be impractical for agencies to control the exact format of the SOAP faults generated. These faults may also expose implementation details that agencies do not wish to reveal to external parties.

Thus, in line with P.03, this document proposes that core services undertake the task of filtering and unifying these faults. This is reflected in Figure 5 in the last exception - “Agency gateway/processing system internal error”. The subcode(s) of the agency fault will be transcribed into a subcode or subcodes of the fault provided to the client software. Similarly, the Node element value and Reason value in the agency fault will be provided as the equivalent values in the fault provided to the client software. This information is to assist the user in communications with the agency, and is not intended to be interpretable by the client software.

Any information provided in the “Detail” element of the agency generated fault will be logged by core services. Its inclusion in the fault sent to the client software is discussed in section 4.5.2.6.

An example of the fault (after processing by Core Services) representing an internal error within an agency is shown below. Note that the agency specific subcode is for example purposes only.

It should be noted that the above treatment SHALL only be applied to faults from agencies that have a code of “env:Receiver”. Faults from agencies with a code of “env:Sender” will be passed unaltered through Core Services (see 4.5.3.1).

```
<env:Fault >
  <env:Code>
     <env:Value>env:Receiver</env:Value>
     <env:Subcode>
        <env:Value>sbr:SBR.GEN.FAULT.INTERNALERROR</env:Value>
        <env:Subcode>
           <env:Value>sbr:SBR.GEN.FAULT.FAULTFROMAGENCY</env:Value>
           <env:Subcode>
              <env:Value>sbr:ATO.GEN.FAULT.STACKTRACE</env:Value>
           </env:Subcode>
        </env:Subcode>
     </env:Subcode>
  </env:Code>
  <env:Reason>
     <env:Text xml:lang="en">An error was detected within the ATO</env:Text>
  </env:Reason>
  <env:Node>http://sbr.gov.au/comn/node/ato/gov/au</env:Node>
</env:Fault>
```

Figure 18: SOAP Fault indicating an agency internal error

4.6. MESSAGE EVENTS

In order that every message exchange has an explicit indication of its result, every response to a service request MUST include one MessageEvent as part of the Standard Business Document Header. An event MUST include at least one MessageEventItem. Note that items MAY not necessarily be ordered by severity within a MessageEvent.

4.6.1. Use of event item fields

4.6.1.1. 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 unharmonised, 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 support of principle P.03.

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.

4.6.1.2. Severity Code

Items are categorised by severity, the options being Information, Warning or Error. To facilitate efficient processing logic, the most severe category of item in a MessageEvent MUST be duplicated as part of the MessageEvent structure in the MaximumSeverity field.

A MessageEvent that does not include any items with a severity of Error is considered to indicate the successful initiation of the request. (In many instances, the completion of the request will also have occurred prior to the generation of a response, but in some cases, there may be follow up processing subsequent to the return of the response).

In the common situation of successful requests, the MessageEvent MAY contain a single item with a severity of Information and a MaximumSeverity of Information. The example below shows the minimum information that would need to be provided in this situation.

```
<MessageEvent>
  <Message.Event.MaximumSeverity.Code>Information</Message.Event.MaximumSeverity.Code>
  <MessageEventItems>
     <MessageEventItem>
     <Message.Event.Item.Error.Code>SBR.GEN.GEN.OK</Message.Event.Item.Error.Code>
     <Message.Event.Item.Severity.Code>Information</Message.Event.Item.Severity.Code>
     </MessageEventItem>
  </MessageEventItems>
</MessageEvent>
```

Figure 19: Minimal MessageEvent indicating success of the request

Conversely, in order to indicate failure of a request, the response MUST include a MessageEvent with at least one item with a severity of Error. This will result in a value of Error in the MaximumSeverity field. Where multiple logical lodgements are contained within a single request message (for example PAYG Summaries or TFN Declarations), a severity of Error on the response message MUST be interpreted as meaning that all logical lodgements within the request have failed, even if the EventItem information only identifies problems with some of the lodgements.

4.6.1.3. 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 “<” 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   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.

4.6.1.3.1. Short Description

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.

4.6.1.3.2. Detailed Description

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.

This document does not propose a limit on the length of the content of this field, recognising that, in line with P.02, preference SHOULD be given to clarity of user understanding over efficiency of message size.

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.

4.6.1.3.3. 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.

4.6.1.4. 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.

If no locations are provided, 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 payload 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.

4.6.1.4.1. Sequence Number

Each location MUST include a sequence number, which indicates to which payload document in the incoming request the event item applies. The sequence number SHALL have the same value as the sequence number for the payload document described in section 3.3.3.6.

4.6.1.4.2. Location Path Text

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

The XPath expression MUST be interpreted relative to the Business.Document.Instance.Text container element for a given payload, and will assume the namespace mappings active within this payload. Where possible, XPath expressions SHOULD uniquely identify elements using their XBRL contextRef attribute (applicable to XRL payloads).

The figure below shows an example of an event, employing parameters to provide dynamic content and indicating the field in error in the input XBRL document via a location.

```
<MessageEvent>
  <Message.Event.MaximumSeverity.Code>Error</Message.Event.MaximumSeverity.Code>
  <MessageEventItems>
    <MessageEventItem>
      <Message.Event.Item.Error.Code>SBR.GEN.GEN.INVALIDABN</Message.Event.Item.Error.Code>
      <Message.Event.Item.Severity.Code>Error</Message.Event.Item.Severity.Code>
      <Message.Event.Item.Short.Description>ABN {abn} is not
valid</Message.Event.Item.Short.Description>
      <Message.Event.Item.Detailed.Description>Please check the ABN to confirm that the
number is correct. See &lt;a href="{url}"&gt;www.ato.gov.au&lt;/a&gt; for more
information</Message.Event.Item.Detailed.Description>
      <Parameters>
        <Parameter>
  <Message.Event.Item.Parameter.Identifier>url</Message.Event.Item.Parameter.Identifier>
  <Message.Event.Item.Parameter.Text>http://www.ato.gov.au/path/to/information</Message.Event.
Item.Parameter.Text>
        </Parameter>
        <Parameter>
  <Message.Event.Item.Parameter.Identifier>abn</Message.Event.Item.Parameter.Identifier>
  <Message.Event.Item.Parameter.Text>12345678901</Message.Event.Item.Parameter.Text>
        </Parameter>
      </Parameters>
      <Locations>
        <Location>
          <BusinessDocument.Sequence.Number>1</BusinessDocument.Sequence.Number>
          <Message.Event.Item.Location.Path.Text>/xbrli:xbrl/...
</Message.Event.Item.Location.Path.Text>
        </Location>
      </Locations>
    </MessageEventItem>
  </MessageEventItems>
</MessageEvent>
```

Figure 20: MessageEvent indicating an ABN in the input was invalid

4.6.2. Providing Codes and Descriptions To Software Developers

SBR intends to supply software developers with lists of response messages concerning business events. Each list, in the form of an XML document based on the Message Event structure (as described in section 4.6.2.3 below), SHALL contain a representative sample of the response messages that may be generated by an SBR participating agency. There is also a list of messages identified as common across the SBR Program.

The lists may not be exhaustive, but SHOULD be representative. For each code, where applicable, the list SHALL include the fields described in section 4.6.1.

In some cases, agencies are employing processing systems that add dynamic content into messages, but the nature of these systems makes it impossible to include it in the way described in section 4.6.1.3.3. For these systems, the dynamic content will appear in responses as part of the static text of the message. In order to indicate such situations in sample response messages, the parameter name may be surrounded by square, rather than curly, brackets. For example, “[abn]” in a sample message would be replaced in actual responses by the static text representing the value of the abn.

4.6.2.1. Collection of Agency Code Lists and Code Usage

Each agency SHALL provide a message list to record their codes and associated details. Any rules documented within agency Message Implementation Guides (MIG) SHALL include the code that will be produced in the event that the rule fails. Agency lists thus MUST include all such codes as a minimum.

4.6.2.2. SBR Common Response Messages

To enable consistency with response messages the SBR program has defined a number of common response messages.

4.6.2.2.1. Use of Parameters

Parameters provide a way for agreement to be reached on common codes and description formats, while allowing a degree of agency variability. For instance, parameters might be used to incorporate a URL specific to each agency in an otherwise standard message, such as

“You must be authorised to perform this request. Please refer to {desc} for further information”

In this case, each agency would provide, at run-time, the relevant values for the URL and descriptive text of the hyperlink. Nevertheless, only a single code would be needed for what would otherwise require multiple, agency specific codes.

4.6.2.3. Code List Format for Software Developers

The format of an agency message list consists of a container element “AgencyCodeList” element, with attributes of the agency designation text (as provided as part of the Party element in the Standard Business Document Header) and a version identifier. This root element contains a single MessageEventItems child, the structure of this element being the same as that used to

include items in an SBR Core Services response message. The use of the “parameter” element is modified such that the content of the element provides further information as necessary about the parameter, rather than the parameter value itself. The version identifier uses the major.minor convention. The major version is aligned with the major version of the WSDLs, while the minor version is increased whenever a new version of the code list is produced for an agency.

4.7. ERROR CODING EXAMPLE

The code below captures the general logic expected of a business application when processing a message response. It is intended as informational, and should not be taken as normative.

```
Try
     Send request and get response
     if (MaximumSeverity = Error)
          For each (EventItem: i)
               Process based on descriptions and locations
     else
          // Success – check for warnings
          If (MaximumSeverity == Warning)
               For each (EventItem: i)
                    Process based on descriptions and locations
          Else
               Ensure at least one item with Information severity
          Process SBDM
Catch (exception e)
     If (e.timeout)
          Queue request for resubmission with increasing delay between attempts
          Notify user – no further action required
     else
          Switch (Fault Code)
               Case Receiver:
                    Switch case subcode
                         Case sbr:Unavailable
                              Queue request for resubmission with increasing delay
between attempts, using "AvailableAfter" information where present
                              Notify user – no further action required

                         Case sbr:InternalError
                              Queue request for resubmission with increasing delay
between attempts
                              Notify user
                              Possible specific action based on subcode,
                         e.g. log error, generate error report
               Case Sender:
               Case default:
                    Report problem to software vendor capturing the subcode
               and reason
```

5. SECURITY

This section will only describe the security aspects associated with the 'message on the wire' related to authenticated services and is not applicable to anonymous service. 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.

5.1. OVERVIEW

The main security interactions are shown below.

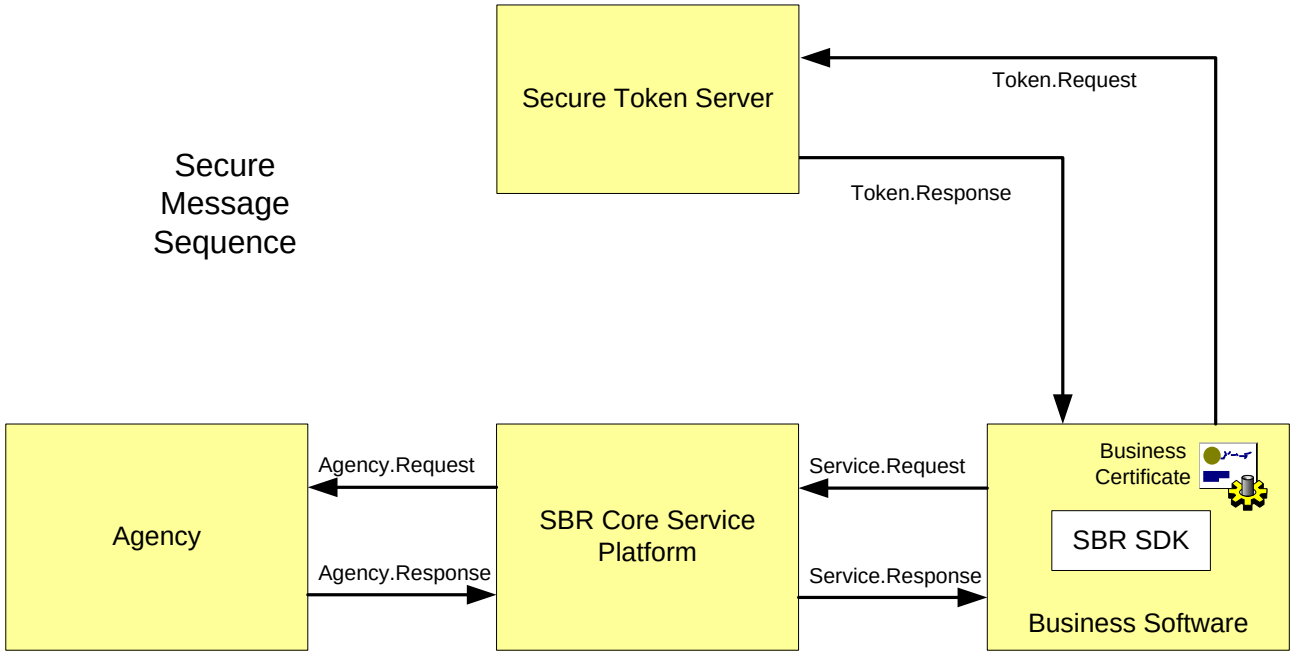

Figure 21: Security Interactions

- The business software presents their business certificate to the Security Token Service (STS), is authenticated, and receives a session key and an encrypted token.
- The business software uses their certificate to sign the Standard Business Document Message within each request, and incorporates this, together with the encrypted token from the STS into the WS-Security header.
- The SBDM signature, the STS token and the entire SOAP body are signed using the session key obtained from the STS. The complete SOAP message is then sent to the Core Service platform.
- The SBR Core Services platform decrypts the token and checks various aspects of the security information provided.
- The agency identifies and authorises the business using data in the decrypted token, processes the business message and returns a response.
- The business software receives the business response via the SBR Core Services platform. Depending on the agency, the SBDM MAY be signed by the responding agency.

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).

The SBR Core Services interaction (Service.Request and 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).

5.2. IMPLEMENTATION OPTIONS

The SBR program provides software developers with two implementation choices for the security framework related to SBR Core Services:

- A "reference client" and sample applications which are part of the SBR Core Services SDK (Software Developer Kit) provided by SBR for the corresponding platform. The reference client is available in Java or .NET versions but sample applications are available for all three platforms including C. Software developers SHOULD refer to the reference client and sample application documentation for details on how to use them.

- Software developers who prefer to build their own implementation without dependencies on the SBR Core Services reference client 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 core service platforms.

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

5.3. SECURITY TOKEN SERVICE (STS)

In a typical WS-Trust scenario, a "relying party" (e.g. an SBR Core Service) 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 (Core Services). A client that wishes to invoke a service offered by a relying party will normally:

- Request the security policy from the relying party – which is returned as a set of "claims".
- Authenticate to the STS using a valid credential and provide the set of claims it requires to the STS.
- The STS will provide and sign a set of assertions that validate the client identity.
- 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.
- 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 Core Services 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 SBR Core Services SDK.

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

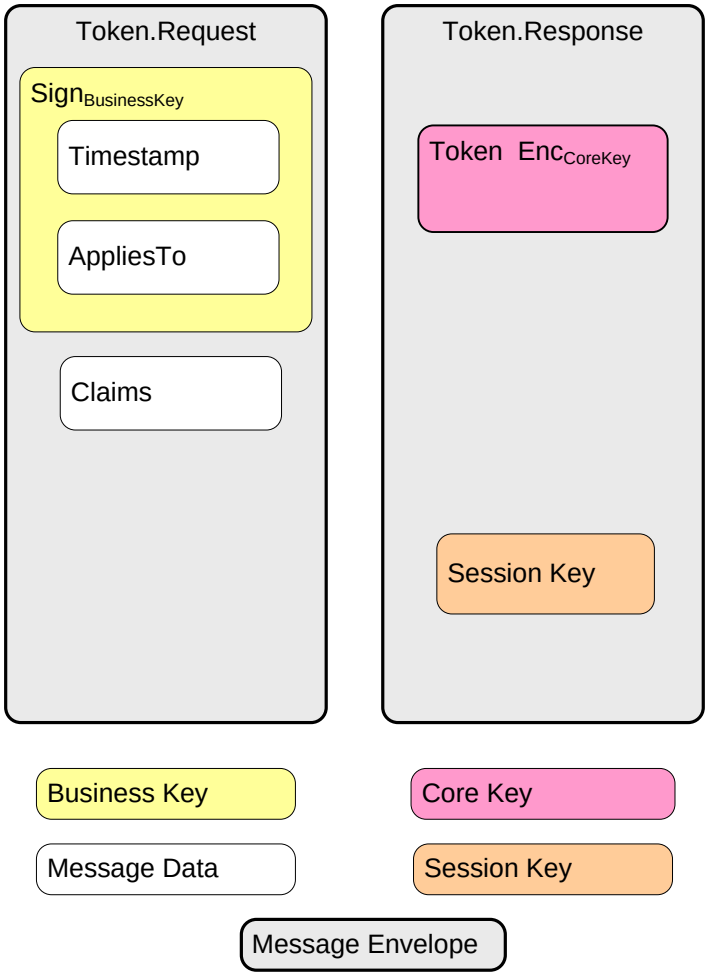

Figure 22: STS Request and Response Envelopes

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 Core Services 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 Core Services.

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 5.3.1.

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	Optional	Optional

	this many minutes.		
http://vanguard.ebusiness.gov.au/2008/06/identity/claims/subjectdn	User X.509 distinguished name	Optional	Optional
http://vanguard.ebusiness.gov.au/2008/06/identity/claims/issuerdn	Issuer (ABR) X.509 distinguished name	Optional	Optional
http://vanguard.ebusiness.gov.au/2008/06/identity/claims/notafterdate	Certificate validity expiry date time	Optional	Optional
http://vanguard.ebusiness.gov.au/2008/06/identity/claims/certificateserialnumber	Certificate serial number	Optional	Optional

Table 22: 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.

5.3.1. 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.sbr.gov.au/services</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 23: 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 Core Services (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 lifetime for an SBR Core Services session key is set to 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.

5.3.2. 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.sbr.gov.au/services</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+BkTCBbFojAEJaYIvu9NIqF
Z8THUZVIbhjBge6miNrsx+kRz70+QJKv6F9WmCmH+rQgWNi1T1MjEP9xIcVLcZDzvFBPEJlvK13KLV5Hoimxp/8F8=</e:Cip
herValue>
                                    </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+djWRs
8orObzXDclQ==</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 24: Token.Response

The token response includes the following elements:

a. The <trust:RequestSecurityTokenResponseCollection> is the envelope element that contains all the token data. It contains one <trust:RequestSecurityTokenResponse>.

b. The <trust:KeySize> element defines the key size of the symmetric session key. The session key is used to sign envelopes sent to SBR Core Services. This is for information only.

c. 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 Core Services.

d. The <wsp:AppliesTo> element is the identifier for the service end point (in this case the SBR Core Services platform) that is the "relying party" in the WS-Trust interaction. This is for information only.

e. The <trust:RequestedSecurityToken> element contains the **<EncryptedAssertion>** structure. The client should insert the entire <EncryptedAssertion> structure into the SOAP header of the SBR Core Services 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 Core Services to decrypt the claims.

f. The <trust:RequestedProofToken> element contains the symmetric key that is used by the client to sign interactions with SBR Core Services until it expires and must be renewed.

g. The <trust:RequestedAttachedReference> element contains the unique identifier for the symmetric session key (the <trust:RequestedProofToken>). This is not currently used.

h. 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.

5.3.3. 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, 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.

5.3.3.1. BusinessContext Element

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

5.3.3.2. 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 subcode tree consistent with the approach used by Core Services faults, but is also available as part of the information in the BusinessContext element.

5.3.3.3. 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 Core Services cannot proceed until the source of the error has been rectified.

5.3.3.4. 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.

5.3.3.5. 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	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	wst:RequestFailed	E2003	The relying party specified in the AppliesTo element is not recognized.
		E1001[2] E1003[2] E1004[2]	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[1]	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 23: STS Fault Codes

[1]The treatment of errors of this type SHOULD be the same as SBR Core Services Unavailable - see section 4.5.3.2.

[2]The treatment of errors of this type SHOULD be the same as SBR Core Services Internal Errors - see section 4.5.3.3.

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 25: 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.

5.4. SECURE MESSAGING

SBR Core 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 business payload 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 receipt information in the lodgement response or optional agency digital signatures on the response message.

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

The conceptual security structure of any SBR Core Services request and response is shown in the figure below using XBRL as a sample payload type.

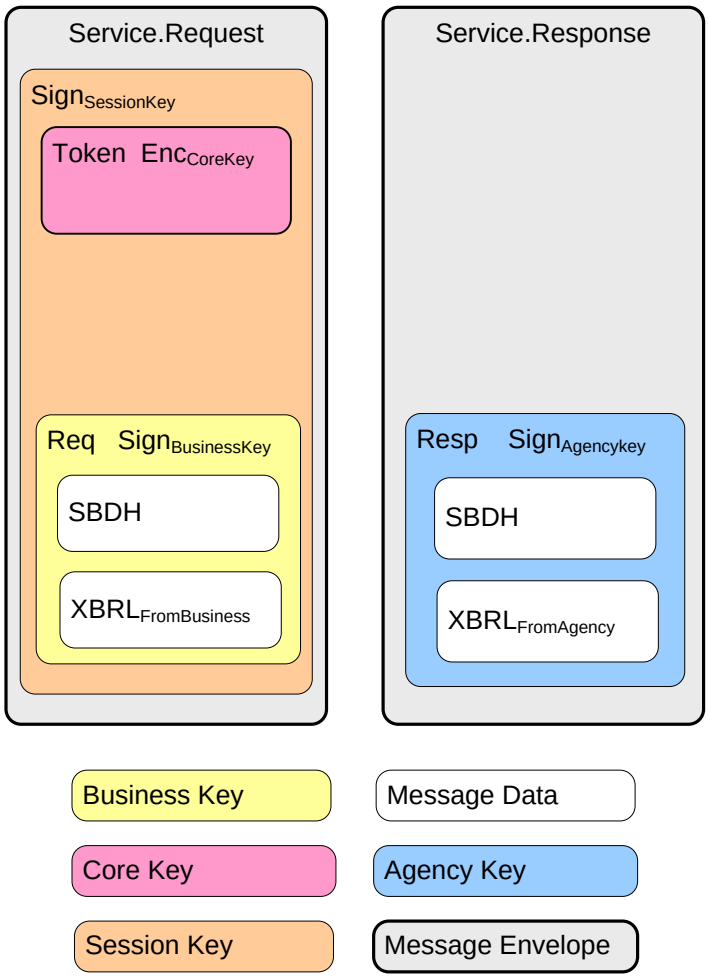

Figure 26: SBR Core Services Secure Messaging

The Service.Request message envelope includes:

- In the SOAP header;
  - The encrypted SAML token exactly as received from the STS.
  - A digital signature of the SBDM (SBDH plus payload) using the business digital certificate.
  - A digital signature of the SOAP Body plus the SAML token using the session key received from the STS.
- In the SOAP Body, the SBDM (Standard Business Document Message) which contains;
  - The SBDH (Standard Business Document Header)
  - Any number of business documents with payloads in supported format
  - Any number of binary attachments.

The Service.Response message envelope includes:

- In the SOAP header;
  - An optional digital signature of the response message SBDM (in the SOAP body) using the Agency key.
- In the SOAP Body the SBDM (Standard Business Document Message) which contains;
  - The SBDH (Standard Business Document Header)
  - Any number of business documents with payloads in supported format
  - Any number of binary attachments.

The provision of the WSDLs for all SBR Core web services together with sample request and response envelopes is discussed in section 7.

5.5. SIGNATURE STRUCTURES

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

The SOAP header contains one <wsse:Security> structure. This structure contains four main sub components:

- A <saml2:EncryptedAssertion> that carries identity information for the agency. This information is provided by the STS and forwarded unchanged to the agency.

- A <wsse:BinarySecurityToken> that carries the business certificate. This is used by the agency to validate the document signature.

- A <ds:Signature id=”signed_sbdm_id”> element that carries the signature of the SBDM and the business certificate, and is signed with the business certificate. This signature is for non-repudiation of origin.

- A <ds:Signature> element that carries the enveloping signature of the SOAP Body, the SAML Token, and the business document signature. This signature links the business document to the SAML identity token.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
        xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
```

```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header>
        <wsse:Security
                soap:MustUnderstand="1"
                xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <saml2:EncryptedAssertion
                    wsu:Id="assertion_id"
                    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">
                <xenc:EncryptedData
                        xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
                        xmlns:a="http://www.w3.org/2005/08/addressing"
                        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
                        xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
                        xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                        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> TdTH/IV7gccbRW2PWou/Cch7P8g8aOqjy35W9nyreTwE=</e:CipherValue>
                            </e:CipherData>
                        </e:EncryptedKey>
                    </KeyInfo>
                    <xenc:CipherData>
                        <xenc:CipherValue> / +euKyWoJmES +ghWa/hnSkMPvHQTn6B0sSSTAVJu5c=</xenc:CipherValue>
                    </xenc:CipherData>
                </xenc:EncryptedData>
            </saml2:EncryptedAssertion>
            <wsse:BinarySecurityToken
                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="BinarySecurityToken-78bd05bf-80de-4bbf-9084-8fb3ee78f423">
                    MIIEHDCCAwSgAwIBAgICCkMwDQYJKoZIhvcNAQEFBQAwgYUxCzAJB
            </wsse:BinarySecurityToken>
            <ds:Signature Id="signed_sbdm_id" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo Id="signedInfo-eac65786-48f7-499b-a688-b1328de4ef9a">
                    <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="#sbdm_id">
                        <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>kvKBax6HB1cVffej7WggalQ8DM4=</ds:DigestValue>
                    </ds:Reference>
                    <ds:Reference URI="#BinarySecurityToken-78bd05bf-80de-4bbf-9084-8fb3ee78f423">
                        <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>R3KAVHwhs7kRpwRG6WrPnOEomCo=</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue Id="signatureValue-46341d8d-e8b4-4e83-85e8-47dc6539fe77">
                    iawKJTf1/elSjy3aX0mhJ+A0ROQp2mgkeogwD5tsT/TPIcNGgmJOCw7cx6YmDDJDBzwMVbrVkCyY
```

```
                        mDwJW1a/T7hxW+/WbaA8t0rLE0IjPuxUDzCzUJdZx/KJy5KGt8GVZ8n6k0BSB8hCeF444Ircggxv
                        05NaSZ8DbopHUuQ6hNk=</ds:SignatureValue>
                    <ds:KeyInfo Id="KeyInfo-f4a3af05-89be-444d-8d91-36c66f23f222">
                        <wsse:SecurityTokenReference>
                            <wsse:Reference
                                URI="#BinarySecurityToken-78bd05bf-80de-4bbf-9084-8fb3ee78f423"
                                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
                        </wsse:SecurityTokenReference>
                    </ds:KeyInfo>
                </ds:Signature>
                <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:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
                        <ds:Reference URI="#signed_sbdm_id">
                            <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>v0dxw6ZPMBV/Pjs6dXUrJb5KMVo=</ds:DigestValue>
                        </ds:Reference>
                        <ds:Reference URI="#soapbody_id">
                            <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>9Yrzsq8dVHJItSzvyuOLK6pUxWY=</ds:DigestValue>
                        </ds:Reference>
                        <ds:Reference URI="#assertion_id">
                            <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>/MmViMAR3/2gmJcXh6JQivjn86o=</ds:DigestValue>
                        </ds:Reference>
                    </ds:SignedInfo>
                    <ds:SignatureValue>aNuW8eO5gWKAMDDP+45iP4XkMNY=</ds:SignatureValue>
                </ds:Signature>
            </wsse:Security>
        </soap:Header>
        <soap:Body wsu:Id="soapbody_id">
            <sbr:RequestPreLodgeReport
                    xmlns="http://sbr.gov.au/comn/sbdm.02.data"
                    xmlns:sbr="http://sbr.gov.au/comn/prelodge.02.service"
                    xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
                <StandardBusinessDocumentMessage wsu:Id="sbdm_id">
                    <StandardBusinessDocumentHeader>
                        <Message.Type.Text>message.ping</Message.Type.Text>
                        <MessageTimestamps>
                            <MessageTimestamp>
                                <Message.Timestamp.Generation.Datetime>2009-03-25T13:53:48.234Z
                                    </Message.Timestamp.Generation.Datetime>
                                <Message.Timestamp.GenerationSource.Code>BusinessEntity
                                    </Message.Timestamp.GenerationSource.Code>
                            </MessageTimestamp>
                        </MessageTimestamps>
                        <Receiver>
                            <IdentificationDetails.IdentifierDesignation.Text>sro.vic.gov.au
                                </IdentificationDetails.IdentifierDesignation.Text>
                            <IdentificationDetails.IdentifierName.Text>AgencyInternetDomainName
                                </IdentificationDetails.IdentifierName.Text>
                            <Party.Type.Code>GovernmentAgency</Party.Type.Code>
                        </Receiver>
                        <SoftwareInformation>
                            <OrganisationNameDetails.OrganisationalName.Text>My Accounting Pty Ltd
                                </OrganisationNameDetails.OrganisationalName.Text>
                            <SoftwareInformation.ProductName.Text>Success!</SoftwareInformation.ProductName.Text>
                            <SoftwareInformation.ProductVersion.Text>12.34.0.56</SoftwareInformation.ProductVersion.Text>
                        </SoftwareInformation>
                        <BusinessDocuments>
                            <BusinessDocument>
                                <BusinessDocument.Sequence.Number>1</BusinessDocument.Sequence.Number>
                                <BusinessDocument.Creation.Datetime>2009-03-25T13:53:46</BusinessDocument.Creation.Datetime>
                                <BusinessDocument.ValidationUniformResourceIdentifier.Text>
                                        http://sbr.gov.au/rprt/asic/t33/t33.declare.lodge.200903.report
                                </BusinessDocument.ValidationUniformResourceIdentifier.Text>
```

```
                        <BusinessDocument.BusinessGeneratedIdentifier.Text>??23465789-WERTZ-5%
                        </BusinessDocument.BusinessGeneratedIdentifier.Text>
                        <Attachments>
                            <Attachment>
                                <Message.Attachment.Sequence.Number>1</Message.Attachment.Sequence.Number>
                                <Message.Attachment.FileName.Text>200809 Annual Report.pdf
                                    </Message.Attachment.FileName.Text>
                                <Message.Attachment.Description.Text>Annual report for the 2008/09 year
                                    </Message.Attachment.Description.Text>
                            </Attachment>
                        </Attachments>
                    </BusinessDocument>
                </BusinessDocuments>
            </StandardBusinessDocumentHeader>
            <StandardBusinessDocumentBody>
                <BusinessDocumentInstances>
                    <BusinessDocumentInstance>
                        <BusinessDocument.Sequence.Number>1</BusinessDocument.Sequence.Number>
                        <BusinessDocument.Instance.Text>
                            <xbrli:xbrl xmlns:xbrli="http://www.xbrl.org/2003/instance">
                                .... insert remainder of XBRL document here
                            </xbrli:xbrl>
                        </BusinessDocument.Instance.Text>
                    </BusinessDocumentInstance>
                </BusinessDocumentInstances>
                <AttachmentInstances>
                    <AttachmentInstance>
                        <Message.Attachment.Sequence.Number>1</Message.Attachment.Sequence.Number>
                        <Message.Attachment.Instance.BinaryObject xmime:contentType="application/pdf">
                                UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi
                        </Message.Attachment.Instance.BinaryObject>
                    </AttachmentInstance>
                </AttachmentInstances>
            </StandardBusinessDocumentBody>
        </StandardBusinessDocumentMessage>
    </sbr:RequestPreLodgeReport>
</soap:Body>
</soap:Envelope>
```

Figure 27: Sample Security Header

5.5.1. 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 the agency. The symmetric key is itself encrypted using the public key of the agency. The <e: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 agency must use the corresponding private key to decrypt the symmetric key. The agency can then use the decrypted symmetric key to decrypt the identity assertions.

- 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 Service.Request security header without change.

5.5.2. 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="BinarySecurityToken-78bd05bf-80de-4bbf-9084-8fb3ee78f423". The agency will use the public key contained in this certificate to validate the business document signature.

5.5.3. Document Signature <ds:Signature id="signed_sbdm_id">

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

- The <ds:SignedInfo element 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 SBDM (<ds:Reference URI="#sbdm_id">) and the business certificate (<ds:Reference URI="#BinarySecurityToken-78bd05bf-80de-4bbf-9084-8fb3ee78f423">).

- The <ds:SignatureValue element contains the actual digital signature.

- 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="#BinarySecurityToken-78bd05bf-80de-4bbf-9084-8fb3ee78f423" element.

5.5.4. Envelope Signature <ds:Signature>

This structure contains the envelope signature created using the symmetric key provided by the STS. There are two sub-structures:

- The <ds:SignedInfo element 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 envelope signature, the signed parts are the business document signature (<ds:Reference URI="#signed_sbdm_id">), the SOAP Body (<ds:Reference URI="#soapbody_id">), and the encrypted SAML Token (<ds:Reference URI="#assertion_id">).

- The <ds:SignatureValue element contains the actual digital signature.

There is no <ds:KeyInfo> element because this signature is created using the STS session key. The encrypted session key is provided for the agency within the encrypted assertions.

6. TESTING

6.1. OVERVIEW

SBR Core Services offers several options to assist software developers in the testing of their products, allowing a software developer to “step up” from basic tests to more sophisticated tests. The options are

1. Network connectivity testing
2. Message connectivity testing
3. Report testing.

Further explanation of these options is provided in subsequent sections.

6.1.1. Service End Points

The following table documents the various end points at which the services described in section 2.2.1 are available.

There are two environments to which software developers have access, one for testing and one for production submission of reports to agencies. The external vendor testing environment (EVTE) offers end points supporting different levels of testing as shown below. Network and message connectivity tests may also be performed in the production environment as part of any diagnostic functions within a software package.

ENVIRONMENT	TESTING	SERVICE	END POINT
Software Developer to Agency Test	• Network connectivity • Message connectivity • Report testing for anonymous interactions	List	https://test.sbr.gov.au/services/nowssecurity/list.02.service
		PreFill	https://test.sbr.gov.au/services/nowssecurity/prefill.02.service
		PreLodge	https://test.sbr.gov.au/services/nowssecurity/prelodge.02.service
		Lodge	https://test.sbr.gov.au/services/nowssecurity/lodge.02.service
	• Network connectivity • Message connectivity • Report testing for authenticated interactions	List	https://test.sbr.gov.au/services/list.02.service
		PreFill	https://test.sbr.gov.au/services/prefill.02.service
		PreLodge	https://test.sbr.gov.au/services/prelodge.02.service
		Lodge	https://test.sbr.gov.au/services/lodge.02.service
SBR Production	• Network connectivity • Message connectivity	List	https://sbr.gov.au/services/nowssecurity/list.02.service
		PreFill	https://sbr.gov.au/services/nowssecurity/prefill.02.service
		PreLodge	https://sbr.gov.au/services/nowssecurity/prelodge.02.service
		Lodge	https://sbr.gov.au/services/nowssecurity/lodge.02.service
	• Network connectivity • Message connectivity	List	https://sbr.gov.au/services/list.02.service
		PreFill	https://sbr.gov.au/services/prefill.02.service
		PreLodge	https://sbr.gov.au/services/prelodge.02.service
		Lodge	https://sbr.gov.au/services/lodge.02.service

Table 24: Service End Points Provided by Core Services

Test and production end points for the Security Token Service are also provided at the locations documented in the table below.

ENVIRONMENT	SERVICE	END POINT
Test	STS	https://thirdparty.authentication.business.gov.au/R3.0/vanguard/S007v1.2/service.svc
Production	STS	https://authentication.business.gov.au/R3.0/vanguard/S007v1.2/service.svc

Table 25: Service End Points Provided by VANguard

6.2. NETWORK CONNECTIVITY TESTING

SBR Core Services conforms to the industry convention of returning the WSDL for a given service if an HTTP GET request is performed on the service URL with the string "?wsdl" appended to it. This applies to any of the end points documented in the above table. The Java version of the WSDL will be returned (see chapter 7).

For example, requesting the URL "https://sbr.gov.au/services/lodge.02.service?wsdl" in a web browser will confirm that network connectivity is present between the network on which the browser is operating and the production Lodge service.

6.3. MESSAGE CONNECTIVITY TESTING

6.3.1. Overview

Having generated syntactically valid SBDM structures, the next phase in development is to add the SBR Core Services security solution to messages, and confirm that it is working correctly. To assist in the testing of the security implementation, SBR agencies offer a simple message connectivity test, called "message.ping", which is designed as an end-to-end connectivity test from business software, through core services, to agencies and back. message.ping is not a separate web service; it is implemented as a particular message type that is supported by each of the four SBR web services (list, prefill, prelodge, lodge). message.ping is implemented in SBR test and production systems.

- In the SBR Core Services test environment, software developers may use the "message.ping" to test that they have correctly implemented all SBR Core Services messaging and security protocols, leaving only the task of creating valid XBRL reports.
- In the SBR Core Services production environment, business users can use "message.ping" as a diagnostic tool in the event of difficulties with business reports. Software developers are encouraged to include this connectivity test as a diagnostic capability within their product release.

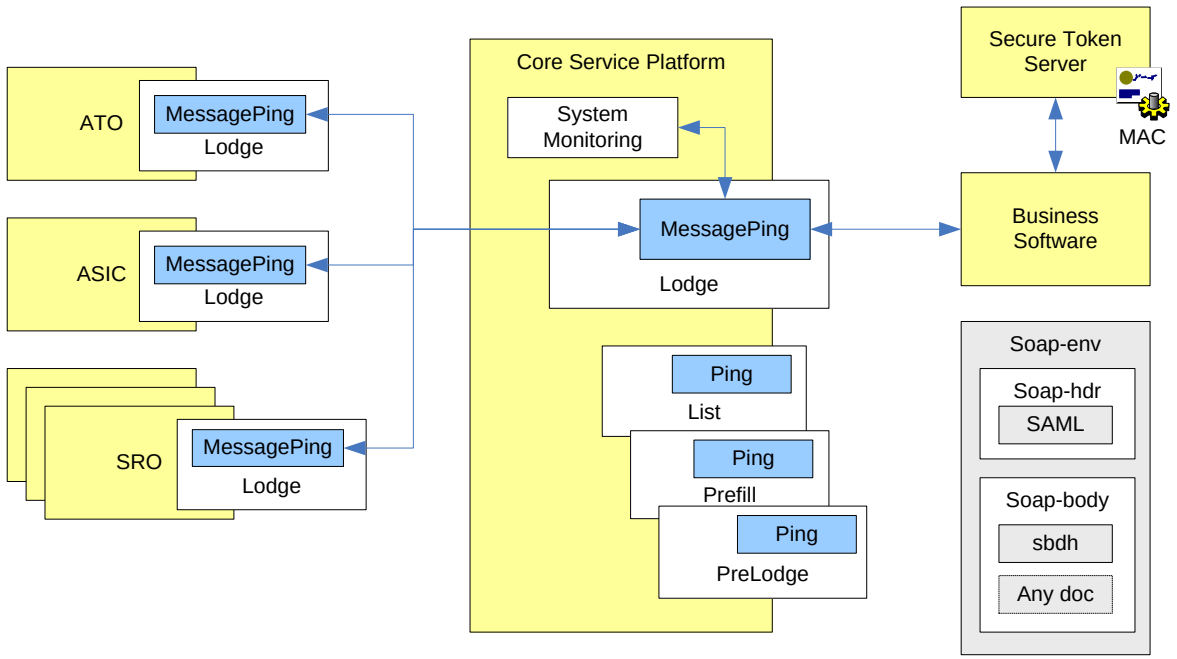

Figure 28: Message Ping

6.3.2. Scenario

The "message.ping" re-uses the existing four core services. Therefore it must also be authenticated like any other interaction. The client must call the Security Token Service (STS) for a SAML token and construct a security header in accordance with the security specification in this document.

A successful message.ping will result in the return of a “message.pong” that is a copy of the message.ping with the addition of timestamps in the SBDH.

Any transport level condition will result in the return of a SOAP fault with error codes as defined in the error handling section of this document. If successful, a MessageEvent element as discussed in section 4.6 will be returned.

The "message.ping" MUST comply with the existing SBDM schema (which allows any number of business documents and binary attachments). Therefore a client MAY send zero or more XBRL instances or binary documents along with their message.ping. Agencies SHOULD copy this content (if it exists) back in the message.pong response but SHOULD NOT validate or process it in any way.

It should be noted that Core Services checks that the number of business documents and binary attachments on a request is within the limits implied by the Message Implementation Guides for the agency/service combination. Thus, for instance, a message.ping with an attachment to an agency service that does not accept attachments will result in a SBR.GEN.FAULT.TOOMANYATTACHMENTS SOAP fault being returned.

6.3.3. message.ping

The message.ping envelope contains an instance of a Standard Business Document Message. All fields in the SBDM should be populated in accordance with the standard request message guidelines described previously. The table below provides specific details on how to populate a request message for the message.ping service.

message.ping can be sent to any SBR participating agency by populating the SBDH Receiver Identification Details appropriately, as per section 3.3.3.3.

message.ping can be sent to any SBR Core Web Service (list, prefill, prelodge, lodge) by invoking the service in accordance with the WSDL for the corresponding service assuming that the target agency supports it.

SBDM ELEMENT	ELEMENT VALUE
Message.Type.Text	“message.ping”

Table 26: Specific SBDM Values For message.ping

Provision of a sample message.ping envelope is provided in section 7.

6.3.4. message.pong

The message.pong envelope is returned by the target agency in response to a successful message.ping. The message.pong is essentially an echo of the message.ping with additional timestamp information in the SBDH. The message.pong MUST follow the standard message response structure defined previously. The table below provides details on how fields specific to message.pong will be populated by the Agency.

SBDM ELEMENT	ELEMENT VALUE
Message.Type.Text	“message.pong”
MessageTimestamp (where GenerationSource.Code = Business Entity)	Date/time as provided on ping request
MessageTimestamp (where GenerationSource.Code = SBRCore)	Date/time request was seen by Core Services (OPTIONAL)
MessageTimestamp (where GenerationSource.Code = GovernmentAgency)	Date/time response envelope was created
BusinessDocuments	As provided in the request.
SBDB	As provided in the request

Table 27: Specific SBDM Values For message.pong

Provision of a sample message.pong envelopes is provided in section 7.

6.4. REPORT TESTING

Having determined that SOAP messages can be successfully generated, secured and sent to SBR Core Services, and that the resulting response can be interpreted, full testing of the desired reports and business obligations can commence.

SBR Core Services offers a range of artefacts to assist this stage of testing including conformance suites, test credentials, test cases and test data.

7. SUPPORTING FILES

7.1. OVERVIEW

The collection of SBR Core Services Web Service Definition Language (WSDL) files, schemas and message samples are provided in zip files on the SBR website. The WSDL files are provided for reference purposes and are only required by developers who choose to build their own clients to interact with SBR Core Services instead of using the SBR SDK.

The WSDL files describing the services provided by Core Services are provided in two variants in separate zip files. The first variant is targeted at .NET version 3.0 (or higher) that uses the Windows Communications Framework (WCF), and the second variant is for Java version 1.5 (or higher) based platforms. If not using the .NET platform, it is suggested that developers use the Java version of the WSDLs. Each of the four services is described by their own WSDL file which is applicable to authenticated and anonymous endpoints for the service. Note that it is not possible to generate .NET clients for the SBR Core Services from the WSDL dynamically fetched using the "?wsld" convention, proxy classes for .NET must be generated from the provided .NET specific WSDL files.

Zip files are also provided containing message samples, as well as message samples for interactions with the Security Token Service.

It should be noted that the examples showing the inclusion of attachments will appear different on-the-wire because of the use of Message Transmission Optimisation Mechanism (MTOM).

The WSDL associated with the STS service is included in the STS message samples zip as a guide, with it being recommended that the actual WSDL be dynamically fetched from VANguard using the "?wsdl" convention.

8. PLATFORM SPECIFICS

8.1. OVERVIEW

The following information is provided for developers who choose to build their own clients to interact with the SBR Core Services instead of using the SBR Core Services SDK.

8.2. .NET

.NET offers two data serialisers, the XML serialiser which has been offered since .NET 2.0, and the Data Contracts Serialiser, which has been offered since the introduction of WCF.

It is recommended, if intending to interact with ASIC, that the XML Serialiser be used in order to ensure the correct operation of MTOM, particularly in relation to the provision of the xmime:contentType attribute.

If not planning to implement interactions that require binary attachments, the SBR Core web service definitions will work with either serialiser. If using the Data Contracts serialiser, however, it is recommended that the svcutil utility be used to generate client code. The /importXmlTypes switch also needs to be used, in order that appropriate code be generated to handle the structures used to hold business document instances.