Australian Government

Australian Financial Security Authority

Online Services: Business to Government (B2G)

AFSA Web Service Gateway Guide

Version 1.1

Table of Contents

Introduction

Purpose

The Australian Financial Security Authority (AFSA) is developing a Business to Government (B2G) service channel. This channel is for parties that have information technology systems that will communicate electronically with AFSA. The purpose of this document is to provide information that will assist software developers in the implementation of software that integrates with the electronic services offered by AFSA.

Audience

The audience for this document is any organisation that will be building AFSA services into their products or business systems. Typically this will be software application developers.

Readers should be familiar with the following:

- Standard Business Reporting (SBR) Program – please see www.sbr.gov.au for further information.
- SBR Web Service Implementation Guide available via email request. Refer to http://www.sbr.gov.au/software-developers/developer-tools/web-services for details.
- Web Services – please see www.ws-i.org for further information.
- AFSA system integration information published at www.afsa.gov.au/online-services/system-integration.

Context

AFSA is standardising its web services to be SBR compliant. For the initial implementation AFSA is hosting its own Web Service Gateway (WSG). It is planned that in the future AFSA will migrate to SBR core services. Transition impacts on software developers will be minimised by the use of a software developer kit (SDK).

SBR follows a “generic” web service approach that allows the separation of the technical and business intent. The technical message places very few constraints on the business message it contains. The only constraints being that the business documents, within the business message, need only be well-formed XML and that the attachments are any binary objects. This can be contrasted with other approaches where the web service contract includes the structure of the business documents.

There are a number of supporting products to facilitate the development of systems that can integrate with AFSA.

Broadly speaking there are four groups of supporting products:

- This document, the AFSA Web Service Gateway Guide (WSGG), which documents the generic *technical service* highlighting differences with SBR and the use of the AFSA SDK. The technical service documentation describes how external software systems must communicate with AFSA. This includes the security requirements, transport protocols, error management and content container. By using the provided SDK the technical service features will already be implemented.

- Implementation guides that provide the entry point for detailed information about how to implement specific *business services*, such as NPII Search. The document that describes the *business service* is called a message implementation guide (MIG). The MIG describes the high level business context of the service, operations that are offered in each *business service* and the request / response *business messages* that are part of each operation. The operations and *business messages* have a textual description within the MIG but are authoritatively defined by XML schemas.

- Technical artefacts that directly support the software developer. This includes the XML schemas mentioned above, which define the valid content of the *business messages*. Another key technical artefact is the SDK which assists developers in creating valid *technical messages* that carry the *business message*.

- General support material and information hosted on the AFSA System Integration pages available at www.afsa.gov.au/online-services/system-integration.

The documentation types described above have dependencies on other documentation. The diagram below shows the key dependencies. This document relies heavily on parts of the SBR WIG where the implementation has minimal differences. The AFSA WSG Guide provides important context for MIGs which describe the business messages. The content of the business messages described in the MIG are defined in the operation and type XML schemas (xsd).

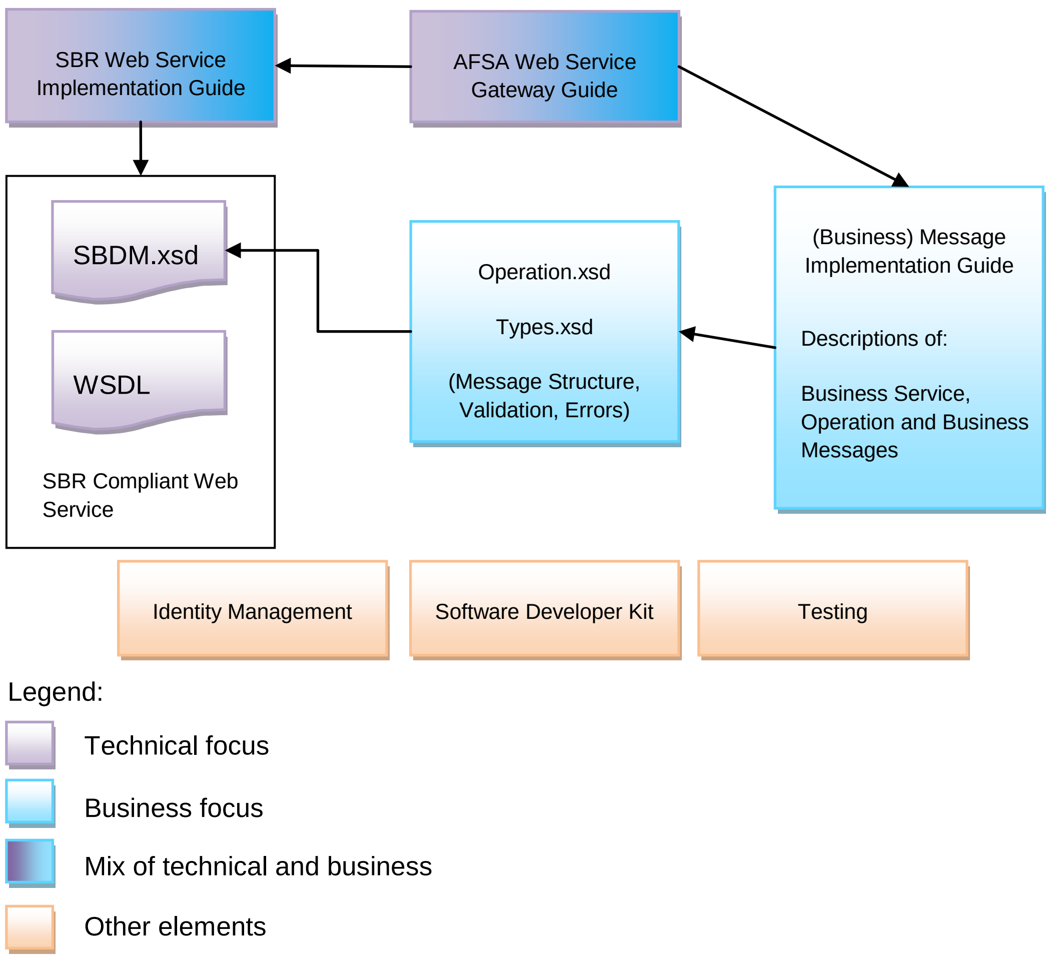

SBR Web Service Implementation Guide
AFSA Web Service Gateway Guide
SBDM.xsd
WSDL
SBR Compliant Web Service
Operation.xsd
Types.xsd
(Message Structure, Validation, Errors)
(Business) Message Implementation Guide
Descriptions of:
Business Service, Operation and Business Messages
Identity Management
Software Developer Kit
Testing
Legend:
Technical focus
Business focus
Mix of technical and business
Other elements

Figure 1 – Related Business and Technical Documents

SBR Compliant Web Services

The AFSA web service is SBR compliant and uses the same Lodge and Prelodge WSDL as SBR but a different endpoint URL. The WSDL describes the service, the technical messages exchanged, and the technical protocols used for the exchange. AFSA provides two web service WSDL for the Lodge and Prelodge web services. The technical message structure is also described in the SBR WIG and SBR specific XML schemas.

Identity Management

The AFSA web services gateway leverages the AUSkey authentication credential which is used by SBR. The SBR WIG 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). Identity management is incorporated in the AFSA SDK which hides implementation details from developers.

Software Developer Kit

There are some common technical components that AFSA expects will be needed by all software developers. The SDK is a set of components created for both Java and .NET platforms that are available for software developers to use in their products. Details of the SDK are provided in the SDK section of this document as well as on the AFSA website.

Testing

The AFSA web services gateway will provide software developers with a suite of test services that can be used to test both the technical and business services. That is the WSDL and a business service such as NPII Search. Supporting the test services is a library of test credentials, Australian Business Numbers (ABN) and test data that can be assigned to developers.

AFSA Website

AFSA’s website is an important information source for developers. Select ‘System Integration’ from the ‘Online Services’ menu to access documentation and other information about the B2G channel.

Differences To SBR Web Service Implementation

No SBR Core In Short Term

SBR Core is a whole of Government electronic gateway that routes messages to the appropriate agency. In the short term AFSA is hosting a SBR compatible web service gateway, which clients will communicate with directly. The diagram below shows the short term solution and the final solution that utilises SBR Core Services. To enable a smooth transition, with only configuration changes anticipated, an AFSA software developer kit

(SDK) is available. The functionality of the AFSA gateway is compatible with SBR core services.

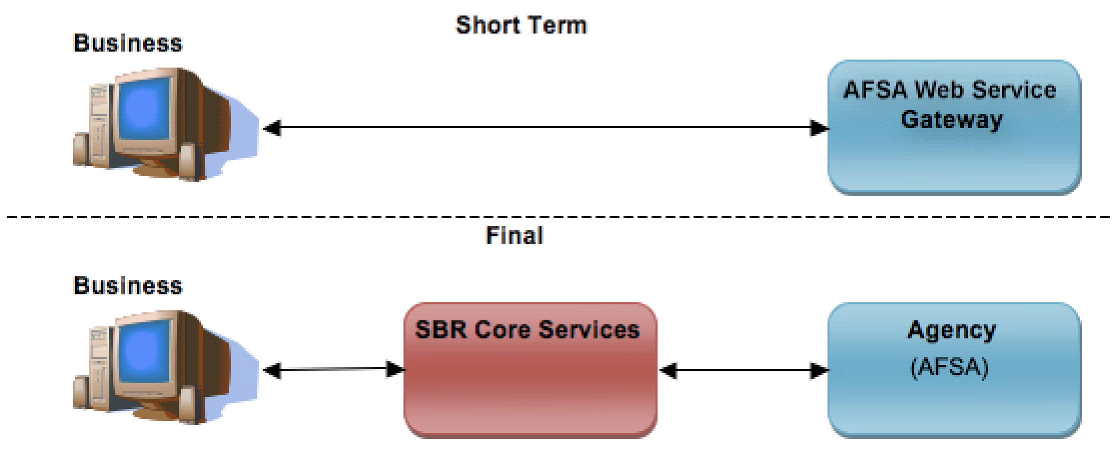


Figure 2 – Short term and long term solution

Two Service

Four technical services are offered by SBR Core Services. They are List, Pre-fill, Prelodge and Lodge. Due to the generic web service implementation of the technical service there is no real difference apart from the name. The only benefit to offering four generic web services may be to offer different service level agreements, such as performance, or different functionality for the same message. AFSA is offering the Lodge and Prelodge services. The Lodge is the most commonly used service..

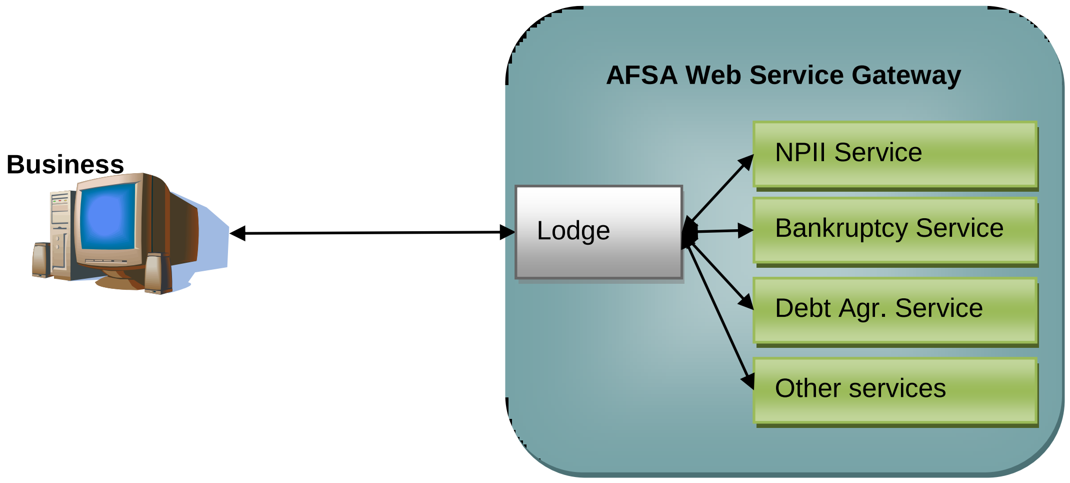


Figure 3 – Lodge technical service

If you want to check if the submission contains any errors or warnings before you call Lodge, you may call Prelodge. Prelodge will validate the request and return all errors and warnings. If the submission is ok, Prelodge will return an ok message. In all cases Prelodge will not save any data on the server, and will not return a business document in the response. Prelodge is routed in the same manner as Lodge.

AFSA Software Developer Kit

The use of the SDK is strongly encouraged as it simplifies the development process and ensures correctly formed messages. This is especially important when considering the security requirements. Whilst AFSA does provide its own SDK this has been adapted from the SBR SDK. Any software developer wishing to replace the SDK with their own code should utilise the SBR Web Implementation Guide to gain a complete understanding of the requirements. Later sections in this document will describe the SDK in more detail.

Taxonomy

The SBR Taxonomy will be used in the future to publish the definitions of the terms used within the AFSA business documents. At present the xml schemas will have annotations to provide brief definitions where necessary.

Message Structure

The business document / message structure will be standard xml documents that conform to an xml schema. There will be a schema file for each operation. The file will contain the definitions for the request and response business documents of the operation. This is slightly different to SBR, which utilises an XBRL format for the business documents. The SDK provides a simplified approach that embeds the business documents within a business message which is then embedded within the Simple Object Access Protocol (SOAP) message and sent to the AFSA web service gateway.

The SBR technical message structure caters for multiple business documents and attachments. This feature was included for batch interactions allowing multiples of the same business document or more complex interactions that require combinations of different types of business documents. The business interactions with AFSA will only require a single business document per technical message. However, multiple attachments will be allowed for the single business document. Attachments are usually supporting documents that have been scanned to enable AFSA staff to verify compliance with legislation.

The technical message is implemented as a SOAP Envelope with a SOAP Header and SOAP Body. Within the SOAP Body is the business message, see Figure 4 Message Structures below. This is implemented as an element called Standard Business Document Message (SBDM). Within the SBDM are a Standard Business Document Header (SBDH) and a Standard Business Document Body (SBDB). Software developers that use the SDK will not need to populate the technical message and will only need to focus development efforts on the creation of business documents that are carried in the business message.

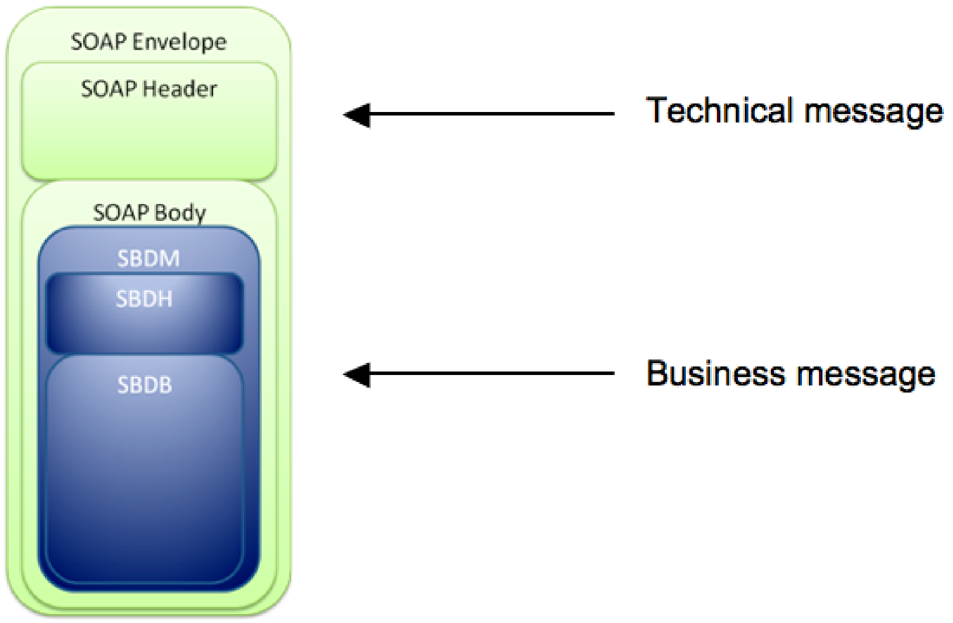


Figure 4 – Message Structure

The MIG describes a business service comprised of operations. Each operation has a request and response message. The request and response messages in the MIG correspond to the business document. The diagram below shows the correlation between the contents of a MIG and the business document and attachments within the business message. The diagram below is indicative only. Request and Response messages will generally be a business document that may have attachments.

The business document in SBR is formatted in XBRL which requires processing by an XBRL toolset. The business documents used by AFSA are formatted in XML which allows a standard XML toolset to be utilised.

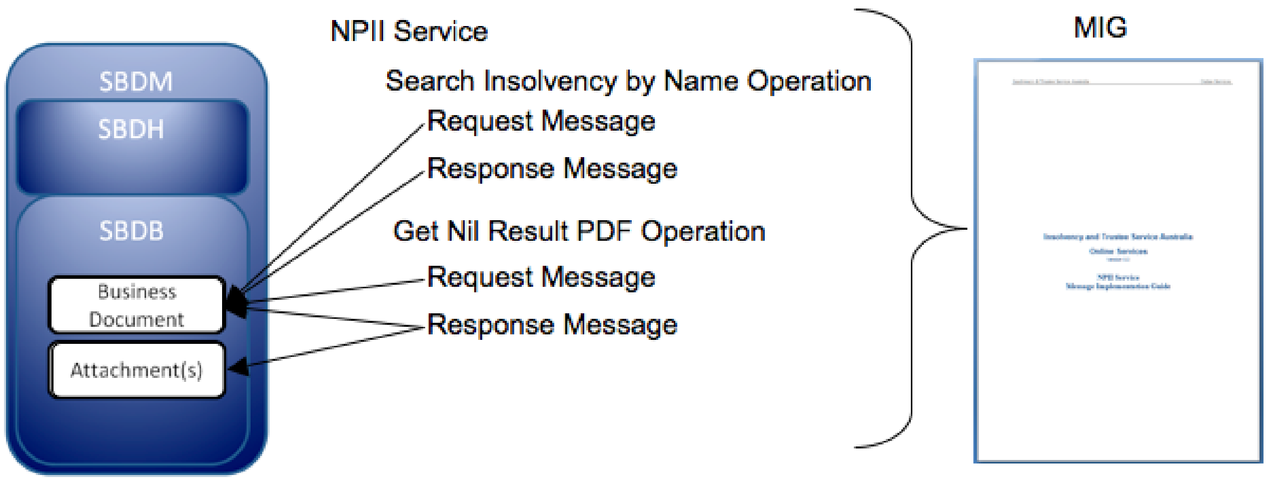


Figure 5 – Correlation between Business Document and MIG

The management of the taxonomy of data element definitions from a whole of government perspective is a major feature of SBR. AFSA will be leveraging the SBR Taxonomy, as mentioned in Taxonomy, to map to existing government terms and to publish additional terms needed by AFSA.

The business document message structures are defined within an XML schema file per operation, contained within a folder per service. Common types are grouped into a single ‘type’ schema and located within the relevant service folder. The schemas are available as part of the AFSA SDK or from the AFSA website.

Error Management

The error management will be the same as SBR. There are some differences as some errors are never raised because there is no communication between SBR core and AFSA. A majority of the errors will be the same.

The same principles, structures and protocols as defined in section 4 of the SBR WIG will be used by AFSA. SBR divides the conditions that can cause errors or exceptions into four broad areas:

1. User errors
2. Client software errors
3. Transport exceptions; and
4. Business events.

The detection and remediation of user or client software errors is the responsibility of the client software provider and therefore will not be considered in this document.

Transport exceptions are associated with the physical transport of messages from client software to the AFSA web service gateway. They cover any problem related to ensuring the technical messages are passed from the client software to the AFSA web service gateway and back again. The adoption of the SOAP 1.2 processing model implies that the SOAP fault mechanism is the primary way in which transport exceptions will be communicated. The SDK will isolate software developers from the implementation details; however the response to the exception will need to be considered. Any developer not using the SDK is referred to the SBR WIG for more details.

The fourth area of errors is from business events. They are called business events because in addition to indicating errors they also cater for information and warning conditions. In the absence of any technical exceptions the only likely source of error is from the business information that is being exchanged. There is a message event structure within the SBDH that contains one or more message event items. Each message event item represents a single business event. The ability to include many event items within a single message event makes it possible to return multiple errors that may exist within the business document.

A message event item is categorised by a severity code with the options being 'information', 'warning' or 'error'. The message event, containing the message event items, has a summary severity code that duplicates the most severe category of all the message event items. This approach allows efficient processing logic, especially for successful responses that do not have any errors and will include only one message event item with a severity categorisation code of 'information'. A response business message may not require a business document or attachment and would therefore not include the SBDB. The structures described in this paragraph are shown in Figure 6 Message Event Structure. Each message event item also has an error code which uniquely identifies any condition that has occurred during processing. This includes the successful processing of a request.

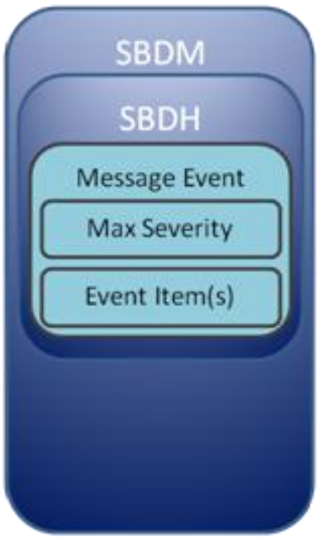
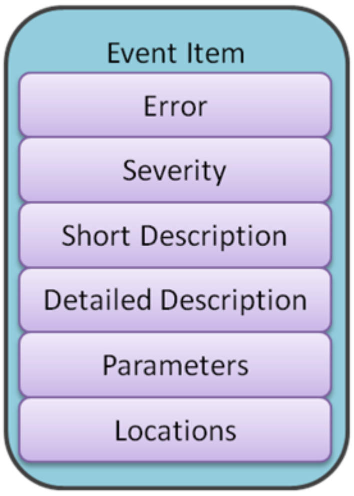


Figure 1- Message Event Structure

There are additional elements within the message item to describe in more detail the errors that have occurred. SBR has provision for both short and detailed descriptions; however AFSA will only be using the short description. Short descriptions can include dynamic content through the use of parameters. An instance of a short description will have the parameters replaced with the actual content.

e.g. "{ABN} is invalid" would appear as "123456789 is invalid".

EventItem.Parameters will always be empty. The source of the error in the original business document can be located by an XPath expression included in the locations section of the message event item.



The format for a message event item error code is as follows: {Jurisdiction}.{Agency}.{Function}.{Id}.

Figure 2 - Message Event Item Structure

All error codes unique to AFSA will be prefixed with CMN.ITSA whilst all other error codes will be prefixed with SBR.GEN.

SBR Errors

- {Jurisdiction}.{Agency} will be SBR.GEN the follow functions apply.
- {Function} will be GEN or FAULT.
- {Id} is a function specific identifier.

Some values for SBR.GEN.GEN are:

Id	Message
SBR.GEN.GEN.OK	Ok. Message was processed successfully
SBR.GEN.GEN.1	ABN {abn} is not valid
SBR.GEN.GEN.2	You are not authorised to lodge this report
SBR.GEN.GEN.3	Intermediaries not accepted for this service
SBR.GEN.GEN.4	Invalid message type text {messageTypeText}
SBR.GEN.GEN.13	ABN not registered with {agency}
etc	

Some values for SBR.GEN.FAULT are:

Id	Message
SBR.GEN.FAULT.MALFORMEDXML	Malformed content from client. The request is not well formed XML, as documented in the XML specification
SBR.GEN.FAULT.INVALIDXML	The request does not validate against the XML Schema for the service
SBR.GEN.FAULT.TOOMANYINSTANCES	XBRL instance limit exceeded. The request contains more XBRL instances than are allowed by the agency. Check the specification and remove excess XBRL instances to a separate message before resending.
SBR.GEN.FAULT.TOOMANYATTACHMENTS	Attachment limit exceeded. The request contains more attachments than are allowed by the agency. Check the specification and resend only with the allowed number of attachments.
etc	

For a complete list of codes refer to the file sbrgovaucodes202.xml contained in the AFSA SDK or downloadable from the SBR website at http://www.sbr.gov.au/__data/assets/file/0014/2453/20100422-sbr-generic-response-messages.zip.

AFSA Errors

- {Jurisdiction}.{Agency} will be CMN.ITSA the follow functions apply.
- {Function} will be the service, being NPII/DA/etc.
- {Id} is a function specific identifier.

See the relevant XSD for a full list of errors.

Id	Message
CMN.ITSA.NPII. SEARCHRESULTLIMITMAXED	The number of search results is greater than the limit

A successful invocation of the AFSA web service gateway would most likely have a single message event item with a code of SBR.GEN.GEN.OK and a severity code of ‘information’.

Warnings

Sometimes a constraint is optional and is not returned as an error, but as a warning. For example, if a Debt Agreement Proposal is submitted with a debtor that is 100 years old, a warning is generated.

To support returning warnings that a user can correct or ignore, a feature has been included in application submission operations. By using the Prelodge endpoint the application can be validated by AFSA. Any errors or warnings will be returned. The application is not saved in our system. If there are no errors and if the warnings are deemed acceptable, the application may be lodged using the Lodge endpoint.

The warnings are returned in the same fashion as errors, except the severity code is ‘Warning’. This means the usual Response body will not be returned. i.e. The responses SBDB will contain no business documents.

Security

The security aspects will be the same as SBR. All security related information is carried in the security elements within the SOAP header. This is in accordance with the Web Service Security 1.1 recommendations. Software developers will need to obtain test AUSkeys from AFSA for the development and test process. When the software has been tested, the AUSkey for the business using the software will need to be setup in accordance with the AUSkey installation instructions. Software developers and clients should review the Australian Business Register (ABR) website for information on how to obtain and install AUSkeys for their organisation.

Software developers are expected to use the AFSA SDK which will alleviate the need for implementing the security requirements. There are some configuration requirements when using the SDK, such as the organisation name and location of the AUSKey, but these are minimal. If any developer does not wish to use the SDK they will need to refer to the SBR WIG for implementation details.

There will be errors related to security which may be initiated from the Secure Token Service (STS) or integration with the AFSA WSG. These errors will be raised as exceptions within the SDK and must be catered for by the software developer.

AFSA Software Developer Kit (SDK)

Overview

The SDK is provided as the AFSA Reference Client, which is a fully functioning example of how to execute AFSA operations. The AFSA Reference Client is made up of a number of artefacts that are explained below. It is provided in the programming languages of .Net C# and Java.

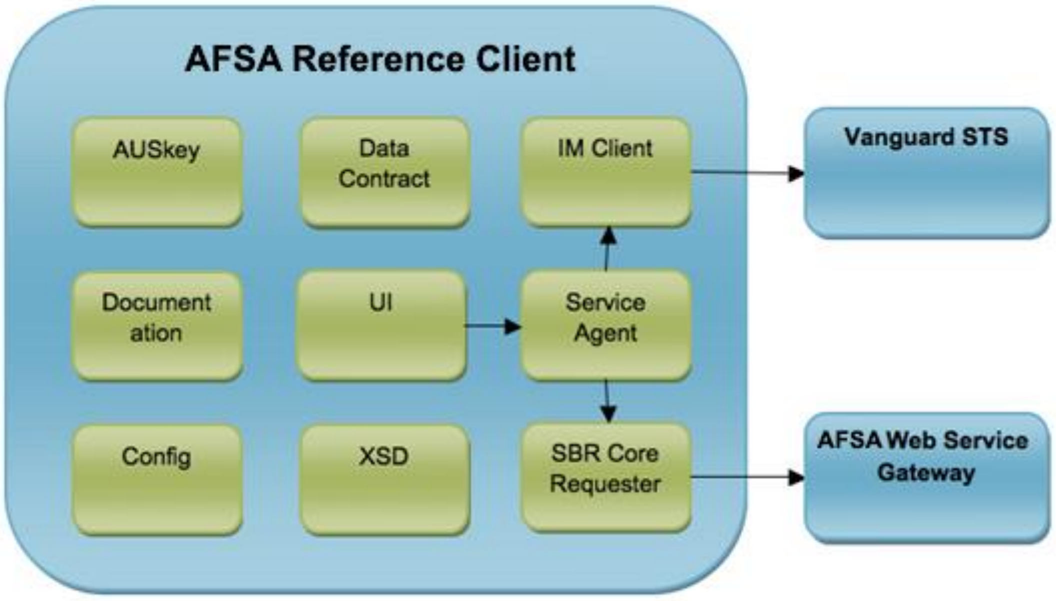


Figure 8 – AFSA Reference Client

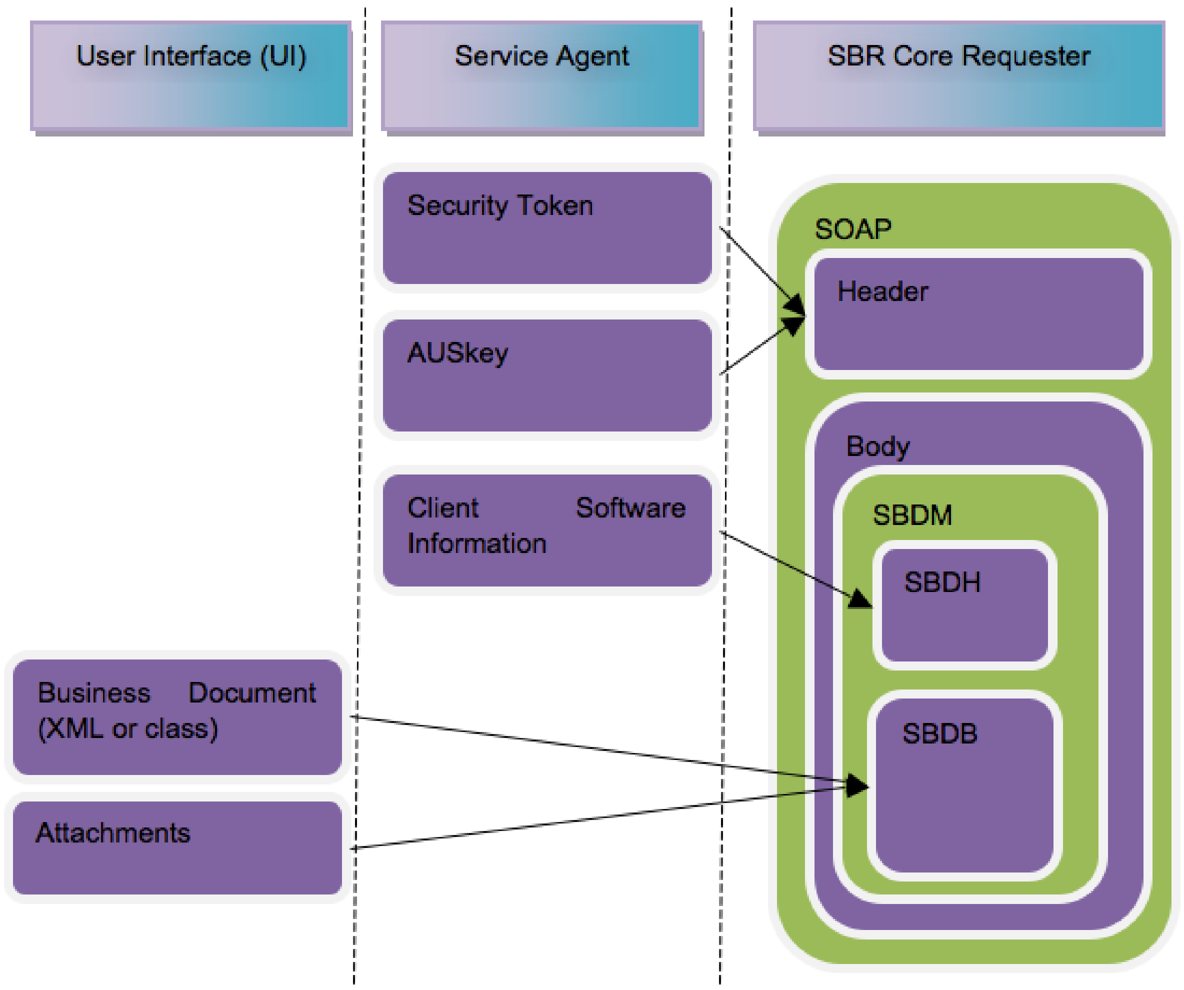
Name	Description
AUSkey	AUSkey is a common authentication solution for business-to-government online services. It is passed to the Vanguard Secure Token Service (STS) to obtain a Security Token. The security token is then used to communicate with the AFSA Gateway. During testing AFSA will provide a test AUSkey to clients. For production, clients must register for an AUSkey at www.abr.gov.au/auskey. *Artefact: Provided separately*
Documentation	Detailed documentation regarding the AFSA Reference Client is provided within the SDK.

	Artefact: SDK\ITSA Reference Client.pdf
UI	This is a sample User Interface (UI) and the starting point when running the Reference Client. It allows the creation of requests that are then executed by the Service Agent. *Artefact: SDK\ITSA Reference Client\UI*
Config	The Configuration (Config) contains values such as: • AFSA Gateway URL • IM Client URL and Claims *Artefact: SDK\ITSA Reference Client\UI\App.config*
XSD	The request and response messages for each operation are defined using XSD. The XSD has been structured to have: • A file per operation named [Service]\[Operation Name].[Version].xsd containing the message definition, and any contained complex types. • A file per service named [Service]\Type.[Version].xsd containing a definition for each field, including its description. • One file containing common types, such as Address, Countries, Phone Numbers. A more human readable version of the XSD is provided with the XSD in the file named Messages.html *Artefact: SDK\ITSA Reference Client\Message\XSD*
Data Contract	.Net and Java classes have been generated from the XSD. The generated classes can be used programmatically and then serialised into XML. *Artefact: SDK\ITSA Reference Client\Message\*
Service Agent	Service Agent is the code that takes a request xml/class, gets the security token and executes the operation. It is expected this will be copied into the client system code and adapted as required. *Artefact: SDK\ITSA Reference Client\ServiceAgent\*
IM Client	IM Client is a library provided and maintained by Vanguard. It contains a number of features, of which we are using the get security token feature. In production it defaults to enable a certificate renewal feature that will automatically request a new business certificate when the existing certificate is close to expiring. *Artefact: SDK\IM Client*

SBR Core Requester	SBR Core Requester is a library provided and maintained by SBR. It contains the necessary code to construct and execute an SBR operation. *Artefact: SDK\SBR Core Requester*
Vanguard STS	Vanguard STS is a web service hosted by Vanguard. The Service Agent requests a Security Token from the STS that is then passed onto the AFSA Gateway. IM Client calls the web service making it unnecessary to directly interact with Vanguard STS.
AFSA Gateway	AFSA Gateway is the web service hosted by AFSA that executes the various operations. There will be two environments available, an integration environment used for testing, and the production environment.

Constructing a Request

The following diagram shows how the message request is constructed. Please see 2.5 Message Structure for information on the various parts of the message.



1. The UI constructs the XML document, and selects any attachments and passes them to the Service Agent.
2. The Service Agent will retrieve a Security Token from the Vanguard STS (or use a cached one), retrieve the client's software information (Company Name, Product Name, Product Version), retrieve the AUSkey and pass them along with the XML document and attachments to the SBR Core Requester.
3. SBR Core Requester will construct the SBDM, package it into a SOAP message and execute the operation.

Testing

An integration testing environment will be provided to support the testing of client software against the B2G interface published by AFSA. Clients will need to register and accept conditions of use, upon which they will be issued with a test AUSkey and other artefacts to support testing.

This environment is currently being provisioned and more detail will be provided as test interfaces become available.

Versioning

The versioning strategy aims to support the changing of messages in a way that minimises the impact on our clients.

XSD Structure

The versioning strategy is reflected in how the XSD files and namespaces are constructed.

- A file and namespace per operation
  - This will include the request and response messages, as well as most complex types used by the operation.
  - The filename will include a major version number, which aligns with the version number in the namespace.
  - A minor version number will be added to the schema elements version attribute.
  - E.g. File NpiiService\GetInsolvency.1.xsd with namespace "http://itsa.gov.au/NpiiService/GetInsolvency.1" and with schema element "<xs:schema id="GetInsolvency" version="1.0"".
  - This will reference the service type file.
- A file and namespace per service
  - This is for all simple types, and some shared complex types.
  - It will contain the change history as a comment at the top of the file for any change to operations within the service.
  - The filename will include a major version number, which aligns with the version number in the namespace.
  - A minor version number will be added to the schema elements version attribute.

    - E.g. NpiiService\ NpiiServiceTypes.1.xsd with namespace http://itsa.gov.au/NpiiService.1
- One common file and namespace
    - This is for things like address.
    - The filename will include a major version number, which aligns with the version number in the namespace.
    - A minor version number will be added to the schema elements version attribute.
    - Common.xsd with namespace http://itsa.gov.au.1

Change Methods

There are 2 change methods that will be used, a minor and a major. In both cases the change will be documented in the change history, which is a comment section at the top of each service type file.

Minor: A minor change is a non breaking change and will result in the minor version number of the affected files being incremented.

Major: A major change is a breaking change and will result in a new version of an operation and its file/namespace, and possibly service type file, being created. This will result in two versions of a message being available. Where possible, both versions will continue to be supported. The switch over timeframe for clients will vary depending on the nature of the change. It is preferred that at most 2 versions will be supported at one time, however this will be driven by business needs.

Development Phase

During a messages development phase the message may be changed in a breaking way but using the minor change method. For example, a new mandatory field may be added to a request. This will be deployed to the testing environment and all clients will need to be updated in expectation of this change. The change will be communicated in advance to allow time for clients to prepare to support it.