Australian Government
Department of Human Services

APPLICATIONS INTEGRATION SERVICES BRANCH

Digital Services

Web Services Standards

DHS External Web Services Profile

Produced by:	Applications Integration Services Branch
Last Edited:	11/06/2019
Version:	1.08 – Final
Classification:	OFFICIAL

Table of Contents

LIST OF TABLES

Stakeholder List

Table 3: Stakeholder List

Name	Role	Team/Branch	Reason(s)
Lorraine Hollis	National Manager	Digital Services	Implementation Owner
Stephen Kirk	Director - Architecture	Digital Services	Implementation Owner

Signatories List

Table 4: Signatories List

Branch	**Digital Services**
Name	**Lorraine Hollis**
Position	**National Manager**
Signature	

1. EXECUTIVE SUMMARY

1.1. Background

This document provides a common standards-based specification to enable secure access to portfolio services that are exposed as Web services. This specification is applicable to all External facing Web service- based projects.

1.2. Purpose

The purpose of this document is to:

1. **Specify the Web services standards specification:** This specification specifies various profiles to be used in different scenarios.
2. **Specify the Web services Security Protocol:** This protocol specifies how security context containing the identify or entity accessing a portfolio Web service is propagated in order to meet authorisation and audit requirements;

1.3. Scope

The scope of the Web services Standards Specification is to describe Web services standards, design and specification of Web services interfaces using WSDL, and its realisation with SOAP messages for:

1. **Government to Government Web services**: The Web service consumer is another trusted government agency with physical network connection consuming a DHS service.
2. **External Business to Government Web services**: The Web service consumer can be any business coming over the internet or trusted physical connection consuming a DHS service.

This document does not address the following aspects:

1. **Service Design**:
2. **Network Details**:
3. **Implementation Details**: how to implement the profile using any particular programming language or toolkit

1.4. Usage

The main use of this document is for developing Service Interface Specification. The Service Interface Specification describes what the service is. That specification would adopt one or many profile(s) from this document.

2. KEY FEATURES OF SPECIFICATION

The key features of the Web Service Profile are:

1. **Standards based**: Use of industry Web services standards for messaging and security.
2. **Interoperability**: The protocol is designed to be interoperable across different technology platforms;
3. **Support for Integrated Audit**: Inclusion of user attributes in the Web Service requests to support audit requirements including the ability to correlate audit events across the systems.
4. **Extensible**: The Web Services Profile is intended for the use of external Web services being used to access in-confidence, sensitive and trusted portfolio data.

2.1. Notational Conventions

The keywords "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in RFC2119.

2.2. Namespace Prefix

This specification uses a number of namespace prefixes throughout; their associated URIs are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant. But for consistence reason it would be good to follow a consistent Namespace prefix naming standards. The following table lists various prefixes to be used.
Table 5: Namespace Prefix

Prefix	Namespace	Name
Standard Namespaces		
xsd	http://www.w3.org/2001/XMLSchema	XML Schema
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance
soapenc	http://schemas.xmlsoap.org/soap/encoding/	SOAP Encoding
soap11	http://schemas.xmlsoap.org/soap/envelope/	SOAP 1.1
soap12	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2
wsoap	http://schemas.xmlsoap.org/wsdl/soap/	WSDL SOAP Binding
wsoap12	http://schemas.xmlsoap.org/wsdl/soap12/	WSDL 1.2 Binding for SOAP 1.2
wsdl11	http://schemas.xmlsoap.org/wsdl/	WSDL 1.1
wsdl20	http://www.w3.org/ns/wsdl	WSDL 2.0 Core Language
wsi	http://ws-i.org/schemas/conformanceClaim/	WS-I
wsp	http://www.w3.org/ns/ws-policy	WS Policy 1.5
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	WS-SecurityPolicy
wsa	http://www.w3.org/2005/08/addressing	WS Addressing 1.0
wsam	http://www.w3.org/2007/05/addressing/metadata	WS Addressing 1.0 Metadata
ds	http://www.w3.org/2000/09/xmldsig#	XML Signatures
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	WS-Security Extensions
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS-Security Utilities

Prefix	Namespace	Name
xmime	http://www.w3.org/2005/05/xmime	MIME Types
Human Services Namespaces		
ce	http://ns.humanservices.gov.au/common/schema/2014/03/15/datatypes	DHS Common Services Schema
cqi	http://ns.humanservices.gov.au/common/qualifiedidentifier/schema/2014/03/15	Common qualifier identifier

2.3. Terminology / Acronyms

Name	Description
HTTP	Hyper Text Transfer Protocol
MTOM	SOAP Message Transmission Optimization Method
SAML	Security Assertions Mark-up Language
SOAP	Simple Object Access Protocol, a protocol for using XML-based messages to exchange structured data in a distributed computing environment.
SOAP message consumer	A program that receives and processes SOAP messages.
SOAP message creator	A program that creates and sends SOAP messages. SOAP message consumer
URL	Uniform Resource Locator
URN	Uniform Resource Name
UUID	Universally Unique Identifier
XML	Extensible Mark-up Language
XOP	XML-binary Optimized Packaging

2.4. Status

This document is at Final stage.

3. WEB SERVICES PROFILE OVERVIEW

'Web service' is defined by the W3C as "a software system designed to support interoperable Machine to Machine interaction over a network." Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.
Web services technology is a collection of standards that can be used to implement vendor-neutral systems. Web services provide an open-standard model for creating explicit, implementation-independent descriptions of service interfaces. Web services provide communication mechanisms that are location-transparent and interoperable.

3.1. Profile

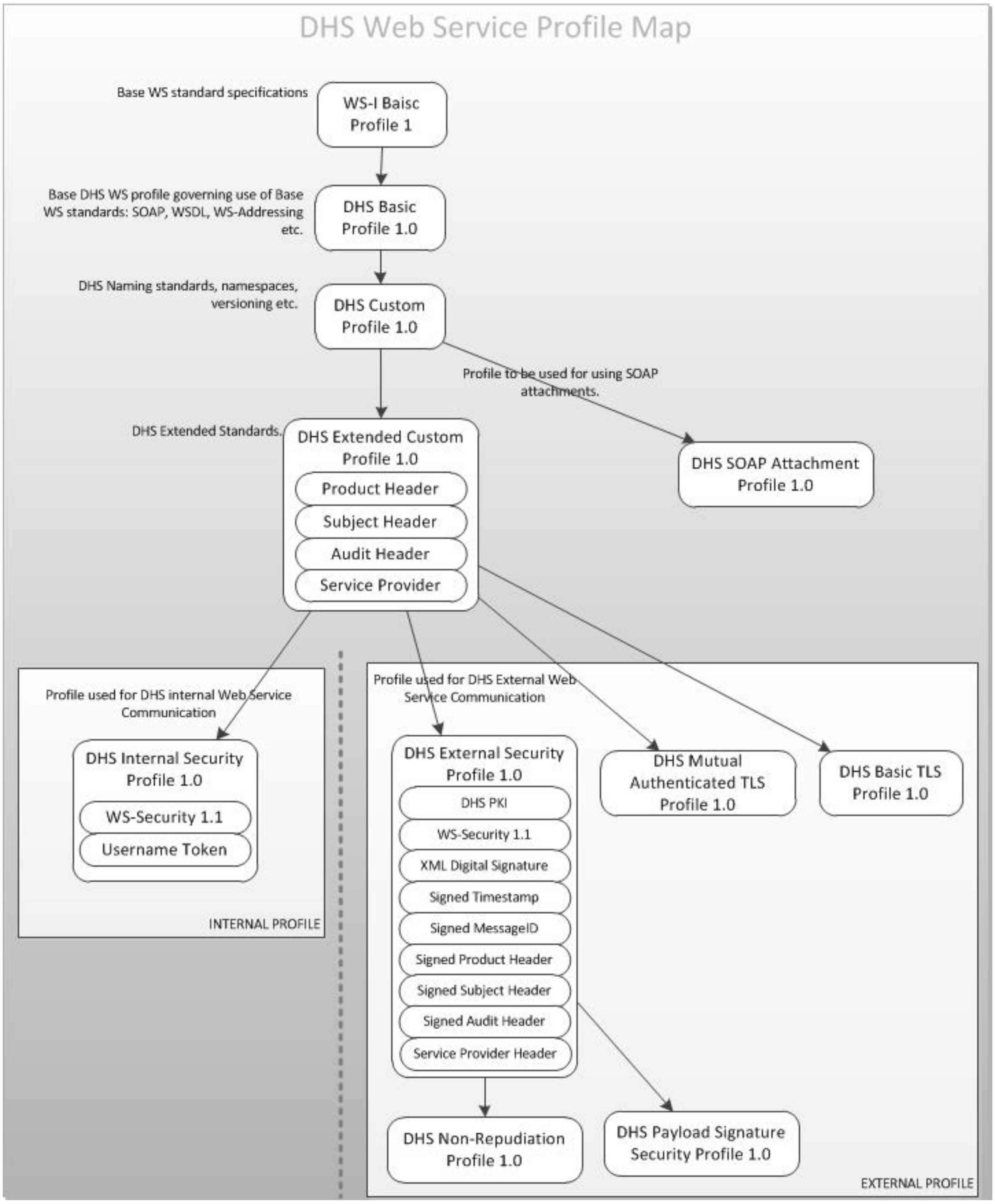
A Profile is a set of guidelines defining use of Web services specifications beyond the core protocols. These guidelines are necessary because the specifications are designed for general-purpose and they are not always enough to satisfy enterprise level requirements. Interoperability Profiles also resolve ambiguities in areas where the Web services specifications are not clear enough to ensure that all implementations process SOAP messages in the same way.

3.1.1. Actors

The criteria in this profile document are categorised according to the role of the actor.
The actors used in this document are:

Actor	Description
Service Interface Specification	Specifies the service interfaces to be implemented by both service invokers and service providers. These criteria will apply to those two actors.
Service Invoker	Service invoker initiate the service call by creating the request message and this actor also consumes the service response and the fault messages.
Service Provider	Service provider actor receives and processes the request message and creates the response and SOAP fault messages.
Service Message Creator	Service message creator actor creates SOAP messages and fault messages.
Service Message Consumer	Service Message Consumer actor consumers the SOAP messages and fault messages.

3.1.2. DHS Profile Mapping


DHS Web Service Profile Map
Base WS standard specifications
WS-I Baisc Profile 1
Base DHS WS profile governing use of Base WS standards: SOAP, WSDL, WS-Addressing etc.
DHS Basic Profile 1.0
DHS Naming standards, namespaces, versioning etc.
DHS Custom Profile 1.0
Profile to be used for using SOAP attachments.
DHS SOAP Attachment Profile 1.0
DHS Extended Standards.
DHS Extended Custom Profile 1.0
Product Header
Subject Header
Audit Header
Service Provider
Profile used for DHS internal Web Service Communication
DHS Internal Security Profile 1.0
WS-Security 1.1
Username Token
INTERNAL PROFILE
Profile used for DHS External Web Service Communication
DHS External Security Profile 1.0
DHS PKI
WS-Security 1.1
XML Digital Signature
Signed Timestamp
Signed MessageID
Signed Product Header
Signed Subject Header
Signed Audit Header
Service Provider Header
DHS Mutual Authenticated TLS Profile 1.0
DHS Basic TLS Profile 1.0
DHS Non-Repudiation Profile 1.0
DHS Payload Signature Security Profile 1.0
EXTERNAL PROFILE

3.1.3. The WS-I Basic Profile

The Web Services Interoperability Organization (WS-I) published WS-I Basic Profile 1.0, which provides guidance primarily on the interoperable use of SOAP 1.1 and WSDL 1.0 etc. In the current version of this document this will be expanded to include SOAP 1.2 and WSDL 1.1.

This specification defines multiple profiles to address different standards and best practices. It is up to the implementing project design to decide on which profile(s) to use.

3.1.4. DHS Basic Profile 1.0

The following table lists the set of standards that form the **DHS Basic Profile 1.0**.

Section Ref	Standard	Description
4.1	XML	Extensible Mark-up Language Specification
4.2	XML Namespaces	XML Namespaces
4.3	XSD (XML Schema)	XML Schema
4.4	SOAP 1.2	SOAP version 1.2
4.5	SOAP Fault	SOAP Fault for SOAP level error messages
4.9	HTTP 1.1	Hypertext Transfer Protocol version 1.0
6.2	WS-Addressing – Action	Web services Addressing Action
6.3	WS-Addressing – MessageID	Web services Addressing Message ID
6.4	WS-Addressing – RelatesTo	Web services Addressing Relates To
5.1	WSDL 1.1	Web Services Description Language Version 1.1
5.2	Separation of interface and bindings	Separation of interface and bindings
5.5	WS-Policy 1.5	Web services Policy version 1.5
5.6	WS-Policy Attachment	WS – Policy Attachments to compliment WS-Policy
5.3	Document literal encoding	Document Literal encoding for SOAP messages
5.4	Wrapped Convention	Wrapped Convention

3.1.5. DHS Custom Profile 1.0

This profile is a set of basic standards needed for every DHS Web service transaction. The **DHS Custom Profile 1.0** includes all of the standards from the **DHS Basic Profile 1.0** listed above, and additionally mandates the use of the following standards references.

Section Ref	Standard	Description
8.1	General Naming Standards	Defines naming syntaxes
8.2	Naming Standards	Defines naming structures
8.3	Namespace Naming	Defines XML namespace values
8.4	Service Versioning	Defines service versioning requirements
8.5	Payload Structure	Defines request and response structures

3.1.6. DHS Extended Custom Profile 1.0

The **DHS Extended Custom Profile 1.0** includes those standards from the **DHS Custom Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
9.2	DHS Product Header	Defines the DHS vendor certification product header
9.3	DHS SubjectId Header	DHS custom header to specify user context to the service provider.
9.4	DHS Audit Header	DHS custom header to specify audit context to the service provider for logging.
9.5	DHS Service Provider Header	DHS custom header to specify service provider routing path to the pass by systems.

3.1.7. DHS External Security Profile 1.0

The **DHS External Security Profile 1.0** includes those standards from the **DHS Extended Custom Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
7.1	WS-Security 1.1	WS Security profile specification v1.1
7.4	XML Digital Signature	XML Digital Signature processing
7.5	Signed Timestamp	A signed timestamp to detect and minimise message replay
7.9	Signed WSA Headers	A signed WSA MessageID
9.1	DHS PKI	DHS Standard for certificate issuance
9.9	Signed ServiceProvider Header	A signed DHS Service Provider Header
9.6	Signed Product Header	A signed DHS Product Header
9.7	Signed SubjectID Header	A signed DHS Subject Header
9.8	Signed Audit Header	A signed Audit Header

3.1.8. DHS Internal Security Profile 1.0

The **DHS Internal Security Profile 1.0** includes those standards from the **DHS Extended Custom Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
7.1	WS-Security 1.1	WS-Security profile specification v1.1
7.7	WS-Username Token	WS-Security Username token for authentication and authorisation

3.1.9. DHS Payload Signature Security Profile 1.0

The **DHS Payload Signature Security Profile** 1.0 includes those standards from the **DHS External Security Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
7.6	Signed Payload	This profile mandates signing the SOAP Body.

3.1.10. DHS Non-Repudiation Profile 1.0

The **DHS Non-Repudiation Profile 1.0** includes those standards from the **DHS External Security Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
7.8	Non-repudiation Logging	Non-repudiation logging profile.

3.1.11. DHS Basic TLS Profile 1.0

The **DHS Basic TLS Profile 1.0** includes those standards from the **DHS Extended Custom Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
7.2	TLS	This profile mandates the use of TLS Basic authentication for transport layer security.

3.1.12. DHS Mutual Authenticated TLS Profile 1.0

The **DHS Mutual Authenticated TLS Profile 1.0** includes those standards from the **DHS Extended Custom Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
9.1	DHS PKI	DHS Standard for certificate issuance
7.2	TLS	This profile mandates the use of TLS Basic authentication for transport layer security.
7.3	TLS Mutual Authenticated	This profile mandates of the use of mutually authenticated SSL connections for client authentication and authorisation.

3.1.13. DHS SOAP Attachment Profile 1.0

The **DHS SOAP Attachment Profile 1.0** includes those standards from the **DHS Custom Profile 1.0** and additionally mandates the use of the following standards references:

Section Ref	Standard	Description
4.6	SOAP Attachments	Specifies the use of MTOM/XOP

4. CORE MESSAGING STANDARDS

4.1. XML

The Extensible Mark-up Language (XML) is a general-purpose specification for creating custom mark-up languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet. XML is a generic language that can be used to describe the content in a structured way, separated from its presentation to a specific device. XML underlies most of the specifications used for Web services.
XML structures will be used to create and share the data between the Service Provider and the service consumers via Web services for this profile.
XML is a W3C Recommendation and more information can be found at [XML2006].

4.1.1. Profile Conformance

4.1.1.1. Service Interface Specification

XML **MUST** be used to specify the message structures as well as any other xml based elements like XML Schema, WSDL etc.

4.1.1.2. SOAP Message Creators

XML **MUST** be used to create the various parts of the SOAP messages.

4.1.1.3. Service Provider

Service provider **MUST** throw a SOAP-Fault if the message received does not conform to the XML specification.

4.1.2. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:request
     xmlns:tns="http://ns.humanservices.gov.au/webservices/schema/2008/06/05/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <tns:uniqueNumber>aaaaaaaaaaaa</tns:uniqueNumber>
     .     .     .
</tns:request>
```

4.2. XML Namespaces

'XML Namespaces' is defined by the W3C as follows: "XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references."
XML Namespaces are used for providing uniquely named elements and attributes by classifying them into an XML Vocabulary. An XML instance may contain element or attribute names from more than one XML Vocabulary. By providing a unique namespace to each vocabulary the ambiguity between identically named elements or attributes can be resolved.
Using XML Namespaces is not mandatory for implementing Web services in general, but its usage will enhance artefact readability.
XML Namespaces is a W3C Recommendation. More information can be found at [XNS2006].

4.2.1. Profile Conformance

4.2.1.1. Service Interface Specification

Service Interface Specification **MUST** use XML Namespaces to uniquely name elements and attributes. The best practices for XML Namespace naming are defined in section
Namespace Service Versioning in the document.

4.2.1.2. SOAP Message Creators

SOAP elements **MUST** be uniquely named using XML Namespaces.

4.2.2. Example

Below is a XML document with appropriate XML Namespaces to provide a simple method for qualifying the element and its attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:request
     xmlns:tns="http://ns.humanservices.gov.au/webservices/schema/2008/06/05/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <tns:uniqueNumber>aaaaaaaaaaaa</tns:uniqueNumber>
     .     .     .
</tns:request>
```

4.3. XSD (XML Schema)

XSD (XML Schema Definition), is a recommendation of the World Wide Web Consortium (W3C), which specifies how to formally describe the elements in an XML document. It defines an XML document with constraints on what elements and attributes may appear, in addition to their relationship to each other, what types of data may be in them, and so on. The description can then be used to verify that each content item in a document agrees to the description of the element where the content is to be placed.
XSD will be used to describe and validate XML objects shared between service consumers and providers in this profile.
XML Schema is a W3C Recommendation, for more details visit the following reference web sites [XSD2004a, XSD2004b].

4.3.1. Profile Conformance

4.3.1.1. Service Interface Specification

Service Interface specification types **MUST** use XSD to define the structure of the messages to be transmitted via Web services.

4.3.1.2. SOAP Message Creators

SOAP Message Creators **MUST** use the interface specification types derived by the XSD schemas to create messages.

4.3.1.3. SOAP Message Consumer

SOAP message consumers **MUST** reject the SOAP message if the message is not derived from the agreed XSD schema.
SOAP message consumers **MUST** respond with a SOAP-Fault if the SOAP Body, SOAP Headers (custom headers) or SOAP Faults (custom faults) are not derived from the agreed XSD schema.

4.3.2. Example

```
<wsdl:definitions
     xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
     <wsdl:types>
          <xsd:schema namespace="http://ns.humanservices.gov.au/dhs/soapmetadata/1.0" >
          ...
          </xsd:schema>
     </wsdl:types>
    <wsdl:message name="messageName">
    ...
    </wsdl:message>
    <wsdl:portType name="PortName">
    ...
    </wsdl:portType>
</wsdl:definitions>
```

4.4. SOAP 1.2

'SOAP' is defined by the W3C as “a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses”.
SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the Web services protocol stack providing a basic messaging framework upon which abstract layers can be built.
SOAP is the preferred message-level protocol for web service messages.
SOAP is a W3C Recommendation, for more details; visit the following reference web sites [SOAP2003a, SOAP2003b].

4.4.1. Profile Conformance

4.4.1.1. Service Interface Specification

Service Interface Specification **MUST** use SOAP 1.2 as the Web services protocol.

4.4.1.2. SOAP Message Creators

Web services requests **MUST** conform to SOAP 1.2 specifications.

4.4.1.3. Service Provider

Service Provider **MUST** throw a SOAP-Fault if the message received does not conform to SOAP 1.2 specification.

4.4.2. Example

```
<soapenv:Envelope
     xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     xmlns:tns="http://ns.humanservices.gov.au/webservices/schema/2008/06/05/">
```

```
      <soapenv:Header>

           ….

           </soapenv:Header>

      <soapenv:Body>

           <m:request
      xmlns:m="http://ns.humanservices.gov.au/webservices/schema/2008/06/05/">

                <tns:uniqueNumber>aaaaaaaaaaaa</tns:uniqueNumber>

                …

           </m:request>

      </soapenv:Body>

 </soapenv:Envelope>
```

4.5. SOAP Fault

SOAP Faults should be used to carry error information back to the originator of a SOAP message. For a SOAP Message to be recognised as carrying SOAP error information, it **MUST** contain a single SOAP Fault element information item as the only child element information item of the SOAP Body
The Fault element information item has:

1. A [local name] of Fault
2. Two or more child element information items in its [children] property in order as follows:
    - A mandatory Code element information item
    - A mandatory Reason element information item
    - An optional Node element information item
    - An optional Role element information item
    - An optional Detail element information item

SOAP-Fault is defined as part of SOAP 1.2 specification by W3C, for more details; visit the following reference web sites [SOAP2003a, SOAP2003b].

The standard SOAP Fault elements should be used as described below and as suggested in the profile conformance section below

Element Name	Format	Mandatory	Description
Code.Value	QName	YES	The Fault code value should follow the SOAP 1.2 specification.
Reason.Text	String	YES	A brief description of what the error relates to.
Role	URI	NO	The use of Role element is recommended and should specify the Role as described in section **11.1 SOAP Roles**
Detail	Service Message format	NO	The use of custom service message format is recommended for a detailed version of the fault.

4.5.1. SOAP Fault Details

SOAP Fault by itself is not enough for Service Provider to communicate error details to the Service Message Creator. A costume error details element **MUST** be mapped to meet that requirement. Service Message Pattern explained later in this profile **SHOULD** be used for this purpose.
There are 4 elements in the Service Message

Element Name	Format	Description
Code	Alpha Numeric	Error Code should be in the Error Code format described in the next section.
severity	Enum	The severity of the message, should be one of "ERROR / WARNING / INFORMATIONAL"
reason	Alpha Numeric	Short description of the reason for the error condition.
details	XML Node	This node can be used for further error details, such as Java Stack Trace or user session object etc.

4.5.1.1. Error Code Format:

In any distributed systems implemented using SOA, understanding the reasons for an action is very important. In a fault situation both the fault generator as well as the fault consumer should be able to understand what and why the error has been generated for. Error Code is a means of referring in what condition an error was generated. This reference **MUST** be further documented in the Service Interface Specification documents.
The Error Code **MUST** be formatted as below. It contains 4 parts

Element Name	Description	Format	Sample
Project Name	3 Char represents the project this error belongs to. E.g. MEB for MyGOV ESB.	Alpha Numeric (3)	MEB
Severity	1 Char represents the severity of the message. It **MUST** be from the list below. • E → ERROR • W → WARNING • I → INFORMATIONAL	Alpha (1)	E
Category	2 Char represents the error category. The Category should be one of 3 listed below. • BU → Business • SE → Security • IN → Infrastructure	Alpha Numeric (2)	SE
Code	4 Numeric, represents the allocated business error code for the specific scenario.	Numeric (4)	1000

4.5.2. Profile Conformance

4.5.2.1. Service Interface Specification

Service Interface Specification **MUST** define one or more fault elements and map them to custom types where possible.
Service interface Specification **SHOULD** define soap faults to use Service Message Pattern message Type.
Service interface Specification **MUST** define all soap fault codes which are not covered by the profile documents.

4.5.2.2. SOAP Message Creators

Service consumers **MUST** be able to support receiving any SOAP Faults, including faults with unknown types from the Service Provider

4.5.2.3. Service Provider

Service Provider **MUST** respond with a SOAP Fault and Fault.Code.Value set to env:Receiver if a WebService layer error occurs.
Service Provider **MUST** respond with a SOAP Fault and Fault.Code.Value set to env:Sender if the request is not valid.
Service Provider **MUST** respond with a SOAP Fault and Fault.Code.Value set to env:Receiver if the intended service processes is not performed.

Service Provider **SHOULD** map the error message to Fault.Code.Reason.Text element.
Service Provider **MUST** represent the error in service message pattern and place it inside the **Details** Node of the SOAP-Fault.
Service Provider **MUST** set the serviceMessage.Error
Service Provider **SHOULD** return a soap fault code which are either defined as part of the web service profile document or service interface specification.

4.5.3. Example

```
<soapenv:Fault>
     <soapenv:Code>
          <soapenv:Value>env:Sender</soapenv:Value>
     </soapenv:Code>
     <soapenv:Reason>
          <soapenv:Text>The web services request timestamp in the web service message is not
valid.</soapenv:Text>
     </soapenv:Reason>
     <soapenv:Role>http://ns.humanservices.gov.au/role/gateway</soapenv:Role>
     <soapenv:Detail>
          <ce:serviceMessages
xmlns:ce="http://ns.services.my.gov.au/common/schema/2013/09/07/elements">
               <ce:highestSeverity>ERROR</ce:highestSeverity>
               <ce:serviceMessage>
                    <ce:code>badTimestamp</ce:code>
                    <ce:severity>ERROR</ce:severity>
                    <ce:reason>The web services request timestamp in the web service message
is not valid.</ce:reason>
               </ce:serviceMessage>
          </ce:serviceMessages>
     </soapenv:Detail>
</soapenv:Fault>
```

4.6. SOAP Attachments

SOAP Attachments are required when pure text-based encoding of the message is not sufficient and transfer of binary data is required. The Web services architecture also supports an Infoset encoding that allows opaque binary data to be interleaved with traditional text-based mark-up. The W3C XML-binary Optimized Packaging, or XOP, format uses multipart MIME [MIME] to allow raw binary data to be included into an XML 1.0 document without resorting to Base64 encoding. Another SOAP binding specification called SOAP Message Transmission Optimization Method (MTOM) specifies how to bind XOP data to SOAP.

4.6.1. Profile Conformance

4.6.1.1. Service Interface Specification

Service Interface Specification **MUST** use the xmime:expectedContentTypes annotation in the schema to denote the MIME type of the binary optimised element.

4.6.1.2. SOAP Message Creators

Service consumers **MUST** use MTOM/XOP for conveying optimised content.

4.6.1.3. Service Provider

Service Provider **MAY** throw a SOAP-Fault if MTOM/XOP has not been used for optimising content as attachments.

Service Provider **MAY** throw a SOAP-Fault if the xmime:expectedContentTypes does not match the actual MIME type used for optimising content as attachments.

4.6.2. Example

The example below presents a SOAP message with two images as SOAP Attachments. The important part to focus is referencing an individual attachment from within the SOAP body; whereby the Include element references an attachment contained in a MIME part outside the SOAP body.

```
--uuid:d40120f9-df21-4959-a27a-96082a3e02b8
Content-Id: <rootpart*d40120f9-df21-4959-a27a-96082a3e02b8@example.jaxws.sun.com>
Content-Type: application/xop+xml;charset=utf-8;type="text/xml"
Content-Transfer-Encoding: binary

<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
     <soapenv:Body>
     <ns2:retrieveAllDoctorInfoResponse xmlns:ns2="http://ns.humanservices.gov.au/soap-
attachments/examples/">
          <return>
          <Include xmlns="http://www.w3.org/2004/08/xop/include"
               href="cid:03986426-e6ad-45cf-83c9-c90f7ddbb615@example.jaxws.sun.com"/>
          </return>
          <return>
          <Include
               xmlns="http://www.w3.org/2004/08/xop/include"
               href="cid:995efea3-d284-4d9c-a3c7-5e59bd0486ee@example.jaxws.sun.com"/>
               </return>
          </ns2:retrieveAllDoctorInfoResponse>
     </soapenv:Body>
</soapenv:Envelope>
--uuid:d40120f9-df21-4959-a27a-96082a3e02b8
Content-Id: <03986426-e6ad-45cf-83c9-c90f7ddbb615@example.jaxws.sun.com>
Content-Type: image/png
Content-Transfer-Encoding: binary
--BINARY IMAGE--
--uuid:d40120f9-df21-4959-a27a-96082a3e02b8
Content-Id: <995efea3-d284-4d9c-a3c7-5e59bd0486ee@example.jaxws.sun.com>
Content-Type: image/png
Content-Transfer-Encoding: binary
--BINARY IMAGE--
--uuid:d40120f9-df21-4959-a27a-96082a3e02b8
```

4.7. SOAP Action

The SOAP Action is a HTTP header defined by the SOAP specification, and it indicates the intent of the SOAP HTTP request. Its value is completely arbitrary, but it is intended to tell the HTTP server what the SOAP message wants to do before the HTTP server decodes the XML.
The function of the SOAP Action has been replaced with WS-Addressing Action SOAP Header. WS-Addressing Action is preferred over SOAP Action for indicating the Web services operation being invoked because SOAP Action unlike WS-Addressing action uses the underlying transport protocol (HTTP) header, which makes the SOAP message closely couple with the transport mechanism. Whereas with WS-Addressing Action the underlying transport protocol can change without any impact to the message structure.

4.7.1. Profile Conformance

4.7.1.1. Service Interface Specification

Service Interface Specification **MUST NOT** define SOAP Action values for any operations in the service.

4.7.1.2. Service Provider

Service Provider **MUST** ignore any SOAP Action values found in the SOAP message.

4.8. Transport Independence

SOAP is defined independently of the underlying messaging transport mechanism in use. It allows the use of many alternative transports for message exchange and allows both synchronous and asynchronous message transfer and processing.
Despite this general transport independence, most of today's Web services communicate using HTTP because this is one of the primary bindings included within the SOAP specification.

4.8.1. Profile Conformance

4.8.1.1. Service Interface Specification

External Service Interface Specification **MUST** use HTTP Version 1.1 as the transport mechanism for Web services.

4.9. HTTP 1.1

HTTP (Hypertext Transfer Protocol) is a communications protocol for the transfer of information on the intranet and the Internet. HTTP is a request/response standard between a client and a server. Where a client is the requestor, the server is the resource provider.
Web Services can be delivered over multiple transport level protocols and one of them is HTTP. The HTTP transport protocol has been identified as the preferred protocol for Web Services communications.
HTTP is a W3C & IETF Recommendation, more information can be found at [RFC2616].

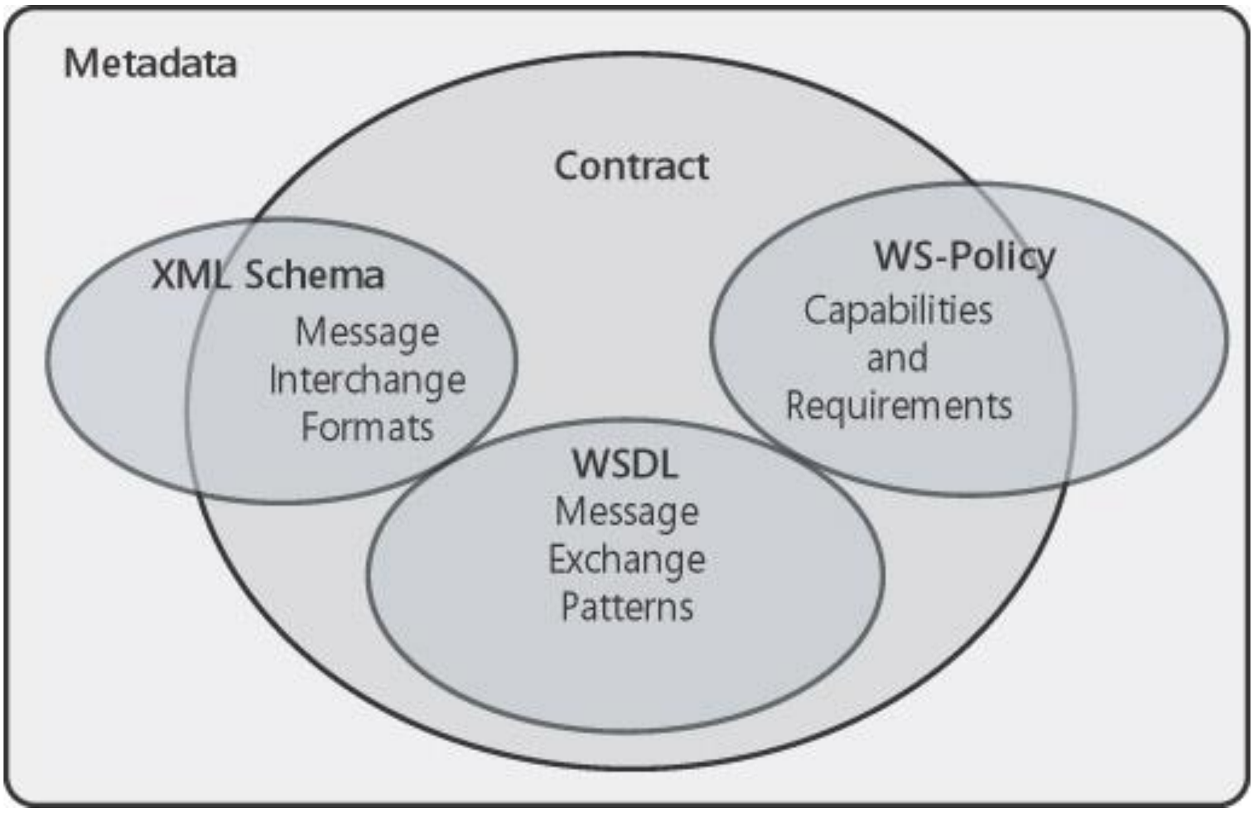
4.9.1. Profile Conformance

4.9.1.1. Service Provider

HTTP Version 1.1 **MUST** be supported by all Service Provider. Other transport standards such as JMS, SMTP etc, **MAY** be used for Web services delivery if required.

5. METADATA STANDARDS

All Web service interactions are performed by exchanging SOAP messages as described in the previous section. To provide for a robust development and operational environment, services are described using machine-readable metadata. Metadata provides system infrastructure that can be used in a variety of ways. Metadata supports automation and tooling, enables automated service discovery, promotes automation of communication between services, and increases interoperability between the services.
The 3 most important purposes of metadata are:
Message Interchange Format: To describe the message interchange formats the service can support (XML Schema)
Message Exchange Patterns: To describe the valid message exchange patterns of a service (WSDL)
Capabilities & Requirements: To describe the capabilities and requirements of a service


Metadata
Contract
XML Schema
Message Interchange Formats
WS-Policy
Capabilities and Requirements
WSDL
Message Exchange Patterns

5.1. WSDL 1.1

WSDL 'Web Services Description Language' is defined by the W3C as "WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.2, HTTP GET/POST, and MIME".
WSDL is a W3C Recommendation, for more details visit the following reference web sites [WSDL2001].

5.1.1. Profile Conformance

5.1.1.1. Service Interface Specification

Service Interface Specification **MUST** describe every service and its elements using WSDL 1.1.

5.1.2. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
     <wsdl:types>
          <xs:schema targetNamespace="http://new.webservice.namespace"/>
     </wsdl:types>
     <wsdl:message name="NewMessageRequest">
     </wsdl:message>
     <wsdl:portType name="NewPortType">
          <wsdl:operation name="NewOperation">
          </wsdl:operation>
     </wsdl:portType>
     <wsdl:binding name="NewBinding" type="tns:NewPortType">
          <wsdl:operation name="NewOperation">
          </wsdl:operation>
     </wsdl:binding>
     <wsdl:service name="NewService">
          <wsdl:port name="NewPort" binding="tns:NewBinding">
          </wsdl:port>
     </wsdl:service>
</wsdl:definitions>
```

5.2. Separation of interface and bindings

A WSDL document consists of two parts: a reusable **abstract** part and a **concrete** part. The abstract part of WSDL describes (type, message, portType elements) the operational behaviour of Web services by recounting the messages that go in and out from services. The concrete part of WSDL (binding, service elements) allows you to describe how and where to access a service implementation.

5.2.1. Profile Conformance

5.2.1.1. Service Interface Specification

Service Interface Specification **MUST** physically separate service descriptions into two parts: a reusable abstract part with all the type, message and portType elements and concrete part with binding and service elements.
Service interface specification **MUST** use WSDL:import functionality to include the abstract part inside the concrete part.
Service Interface specification **SHOULD** use the word **interface** in the abstract wsdl namespace and the word **binding** in the concrete wsdl namespace.

5.2.2. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://ns.humanservices.gov.au/dhsoperation/interfacewsdl/2008/06/05/">
     <wsdl:types>
          <xs:schema targetNamespace="http://new.webservice.namespace"/>
     </wsdl:types>
     <wsdl:message name="NewMessageRequest">
     </wsdl:message>
     <wsdl:portType name="NewPortType">
          <wsdl:operation name="NewOperation">
          </wsdl:operation>
     </wsdl:portType>
</wsdl:definitions>
<wsdl:definitions
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace=http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/>
<wsdl:import
     namespace=
"http://ns.humanservices.gov.au/dhsoperation/interfacewsdl/2008/06/05/"
     location="DHSWebServiceInterfaces.wsdl"/>
     <wsdl:binding name="NewBinding" type="tns:NewPortType">
          <wsdl:operation name="NewOperation">
          </wsdl:operation>
     </wsdl:binding>
     <wsdl:service name="NewService">
          <wsdl:port name="NewPort" binding="tns:NewBinding">
          </wsdl:port>
     </wsdl:service>
</wsdl:definitions>
```

5.3. Document literal encoding

5.3.1. Profile Conformance

Web services have standardised on the use of document / literal model and use of RPC is quickly becoming obsolete. This profile mandates the use of the document / literal encoding as the default and the only type of encoding.
The document style is loosely coupled making it useful for cross organisation operations. In document / literal style, the structure of the SOAP body and the types used are defined by an XML Schema. The literal style is compliant with the widely implemented WS-I Basic profile 1.1 and it complements well with the XML schema structures.

5.3.1.1. Service Interface Specification

Service Interface Specification **MUST** use “Document / Literal” encoding style.

5.3.2. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
     <wsdl:types>
          <xs:schema targetNamespace="http://new.webservice.namespace"/>
     </wsdl:types>
     <wsdl:binding name="NewBinding" type="tns:NewPortType">
     <wsdl:operation name="NewOperation" style="document">
            <wsdl:input>
     <soap:header message="mwui:NEHTAHeader" part="NEHTAHeaderPart" use="literal"/>
                <soap:body parts="DHSRequestPart" use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body parts="DHSResponsePart" use="literal"/>
            </wsdl:output>
            <wsdl:fault name="BadData">
                <soap:body parts="fault" use="literal"/>
            </wsdl:fault>
          </wsdl:operation>
     </wsdl:binding>
     <wsdl:service name="NewService">
          <wsdl:port name="NewPort" binding="tns:NewBinding">
          </wsdl:port>
     </wsdl:service>
</wsdl:definitions>
```

5.4. Wrapped Convention

The wrapped convention is a constraint on how the SOAP body is structured. The basic characteristics of the document/literal wrapped pattern are:
The input message has a single part
The part is a reference to an XML Schema element
The request element has the same name as the operation
The element's complex type has no attributes
The response element has the same name as the operation with 'Response' appended to it

5.4.1. Profile Conformance

5.4.1.1. Service Interface Specification

Service Interface Specification **MUST** follow the wrapped convention for structuring the body of the SOAP message.

5.4.2. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
    <wsdl:types>
        <xsd:schema targetNamespace="http://new.webservice.namespace">
       <xsd:element name="dhsoperation">
          <xsd:complexType>...</xsd:complexType>
         </xsd:element>
       <xsd:element name="dhsoperationResponse">
          <xsd:complexType>...</xsd:complexType>
         </xsd:element>
     </xsd:schema>
    </wsdl:types>
    <wsdl:message name="dhsOperationInputMessage">
     <wsdl:part name="parameters" element="tns:dhsOperation"/>
    </wsdl:message>
    <wsdl:message name="dhsOperationOutputMessage">
     <wsdl:part name="parameters" element="tns:dhsOperationResponse"/>
    </wsdl:message>
    <wsdl:portType name="">
     <wsdl:operation name="dhsOperation">
         <wsdl:input message="tns:dhsOperationInputMessage"/>
         <wsdl:output message="tns:dhsOperationOutputMessage"/>
     </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

5.5. WS-Policy 1.5

'WS-Policy' is defined by the W3C as follow: “The Web Services Policy provides a general purpose model and corresponding syntax to describe the policies of entities in a Web services-based system”.
Web Services Policy 1.5 defines a framework and a model for expressing domain-specific policies which provides a way to describe Web services behaviour, but it does not define policy statements to use. Other standards like WS-Security and WS-Addressing will specify those policy statements.
WS-Policy does not cover discovery of policy, policy scopes and subjects, or their respective attachment mechanisms. WS-Policy Attachment specification covers that topic.
WS-Policy is a W3C Recommendation, more information can be found at [WSPL2007, WSPA2007].

5.5.1. Profile Conformance

5.5.1.1. Service Interface Specification

Service Interface Specification **MUST** describe the policies for Web services in the WSDL using WS-Policy 1.5.
Service Interface Specification **MUST** apply the policies in binding section of the WSDL.

5.5.2. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
    <wsdl:types>
       …
     </xsd:schema>
    </wsdl:types>
    ...
     <wsdl:binding …  >
         <wsp:PolicyReference URI="#dualLayerSecurityPolicy" />
         ...
     </wsdl:binding>
</wsdl:definitions>
```

5.6. WS-Policy Attachment

WS-Policy Attachment is defined by the W3C as "Web Services Policy Attachment defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing metadata about the subject or the policies may be defined independently and associated through an external binding to the subject".
WS-Policy Attachment specification works as a bridging gap between the existing Web services technologies and WS-Policy specification, this specification describes the use of these general-purpose mechanisms with WSDL and UDDI. Specifically, this specification defines the following:

1. How to reference policies from WSDL definitions
2. How to associate policies with specific instances of WSDL services
3. How to associate policies with UDDI entities

WS-Policy Attachment is a W3C Recommendation, more information can be found at [WSPA2007].

5.6.1. Profile Conformance

5.6.1.1. Service Interface Specification

Service Interface Specification **MUST** describe the policies for Web services as WS-Policy Attachments and bind them with the WSDL.

5.6.2. Example

```
<wsp:Policy
     xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
     xmlns:wsrr="http://wsrr/policy"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
     <wsp:PolicyAttachment
          xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
          xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <wsp:AppliesTo>
               <wsp:URI xmlns:wsp="http://www.w3.org/2006/07/ws-policy">
     http://ns.humanservices.gov.au/dhsOperation/bindingwsdl/2008/06/05/#wsdl11.service(dhsOperationService)
               </wsp:URI>
          </wsp:AppliesTo>
          <wsp:PolicyReference
               DigestAlgorithm="http://schemas.xmlsoap.org/ws/2004/09/policy/Sha1Exc"
          URI="http://ns.humanservices.gov.au/policies/2008/06/18/security/includetimestamp"/>
     </wsp:PolicyAttachment>
</wsp:Policy>
```

6. WS-ADDRESSING

WS-Addressing enables messaging systems to support message transmission through networks in a transport-neutral manner. It eliminates the need of transport specific data by adding that information to the XML message itself. This way the message can carry along its own dispatch metadata in a standardised SOAP header. WS-Addressing Metadata specification specifies how to define WS-Addressing properties, such as Action, MessageID in WSDL and how to specify the use of addressing in a WSDL using the WS-Policy framework.

'WS-Addressing ' is defined by the W3C as "A specification which provides transport-neutral mechanisms to address Web services and messages. Web Services Addressing 1.0 - Core defines a set of abstract properties and an XML Infoset representation thereof to reference Web services and to facilitate end-to-end addressing of endpoints in messages".

WS-Addressing is a W3C Recommendation, more information can be found at [WSA2006, WSAM2007].

WS-Addressing introduces a set of message headers to allow messages to be directed to service endpoints and to provide the information necessary to support a rich bidirectional and asynchronous interaction. WS-Addressing describes how message headers are used and processed.

According to the specification only two message headers are required on every message, WS-Addressing – To and Action. Everything else is optional from the specification point of view, but for this profile the use of MessageID and RelatesTo are mandatory and ReplyTo is mandatory for asynchronous services.

The WS-Addressing key elements addressed in this profile are:

1. WS-Addressing – To: A WS-Addressing SOAP-Header which contains the URI address of the target endpoint.
2. WS-Addressing – Action: A WS-Addressing SOAP-Header which contains a URI identifying the intent or semantics of the message.
3. WS-Addressing – MessageID: A WS-Addressing SOAP-Header which contains a URI value that uniquely identifies the message that carries it.
4. WS-Addressing – RelatesTo: A WS-Addressing SOAP-Header which contains a URI value that must be the MessageID of a previously exchanged message.
5. WS-Addressing – ReplyTo: A WS-Addressing SOAP-Header which contains a URI to send the reply messages.

6.1. WS-Addressing – To

The WS-Addressing – To SOAP-Header block in the message information header provides the value for the destination property. This header block contains a single URI that is typically used by the ultimate recipient to dispatch the message for processing. If this element is not present then the value of the destination property is assumed to be WS-Addressing anonymous address which is "http://www.w3.org/2005/08/addressing/anonymous".

6.1.1. Profile Conformance

6.1.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the WS-Addressing – To header as a required element for the SOAP operation.

6.1.1.2. SOAP Message Creators

Service Message Creator **MUST** include the WS-Addressing – To Header in SOAP request Message.

6.1.1.3. Service Provider

Service Provider **SHOULD** respond with a SOAP-Fault indicating WS-Addressing – To policy fault if the SOAP Request violated the above mentioned WS-Addressing – To SOAP Message Creator rules.

Service Provider **MUST** ignore the value of WS-Addressing – To.

6.1.2. Example

```
<soapenv:Envelope
     xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsa="http://www.w3.org/2005/08/addressing">
     <soapenv:Header>
          <wsa:To>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpoint</wsa:To>
          …
     </soapenv:Header>
     <soapenv:Body>
     …
     </soapenv:Body>
</soapenv:Envelope>
```

6.2. WS-Addressing – Action

The Action header block in the message information header is used to indicate the expected processing of a message. This header block contains a single URI that is typically used by the ultimate recipient to dispatch the message for processing.

6.2.1. Profile Conformance

6.2.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the WS-Addressing – Action values as a required element for the SOAP messages of all operations.

6.2.1.2. SOAP Message Creators

Service Message Creator **MUST** include the WS-Addressing – Action Header value for every SOAP request Message. Service Message Creator **MUST** set the WS-Addressing – Action value to a valid URI which can be used by the ultimate service provider to dispatch the message for processing.

6.2.1.3. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the SOAP request message does not contain a valid WS-Addressing – Action value.

6.2.2. Example

```
<wsdl:definitions
     xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
     xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
     <wsdl:types>…</wsdl:types>
    <wsdl:portType name="DhsOperationPortType">
        <wsdl:operation name="dhsOperation">
            <wsdl:input name="dhsOperationPortTypeRequest"
```

```
                message="tns:DhsOperationPortTypeRequest"
wsam:Action="http://ns.humanservices.gov.au/testservice/bindingwsdl/2008/06/05/request"/>

            <wsdl:output name="dhsOperationPortTypeResponse"

                message="tns:DhsOperationPortTypeResponse"
wsam:Action="http://ns.humanservices.gov.au/testservice/bindingwsdl/2008/06/05/response"/>

            <wsdl:fault name="BadData"

                message="tns:StandardError"
wsam:Action="http://ns.humanservices.gov.au/webservices/dhsoperation/fault/baddata"/>

        </wsdl:operation>

    </wsdl:portType>

</wsdl:definitions>
```

```
<soapenv:Envelope

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

     xmlns:wsa="http://www.w3.org/2005/08/addressing">

     <soapenv:Header>

          <wsa:To>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpoint</wsa:To>

          <wsa:Action>

               http://ns.humanservices.gov.au/testservice/bindingwsdl/2008/06/05/request

          </wsa:Action>

     </soapenv:Header>

     <soapenv:Body> … </soapenv:Body>

</soapenv:Envelope>
```

6.3. WS-Addressing – MessageID

The WS-Addressing MessageID and RelatesTo header blocks are used to identify and correlate individual messages. The MessageID and RelatesTo headers use simple URIs to uniquely identify messages— typically these URIs are transient UUIDs.

6.3.1. Profile Conformance

6.3.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the WS-Addressing – MessageID values as a required element for the SOAP messages of all operations.

6.3.1.2. SOAP Message Creators

Service Message Creator **MUST** include the WS-Addressing – MessageID Header value in every SOAP Message (requests, responses, faults).
Service Message Creator **MUST** use a value that is a globally unique id as the MessageID value.
Service Message Creator **MUST** use a UUID formatted URN as specified by [RFC4122] as the MessageID value.

6.3.1.3. SOAP Message Consumers

SOAP Message Consumers **MUST** reject a SOAP Message without a valid MessageID Header value.
SOAP Message Consumers **MAY** reject a SOAP Message with a MessageID value not confirming to the UUID formatted URN as specified by [RFC4122].
SOAP Message Consumers **SHOULD** reject a SOAP Message with duplicate MessageID value.
Where SOAP Message Consumers rejects a SOAP Message due to problems with the MessageID Header value they **MUST** respond with a SOAP-Fault indicating that the MessageID was invalid.

6.3.2. Example

```
<soapenv:Envelope
      xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
      xmlns:wsa="http://www.w3.org/2005/08/addressing"       >
      <soapenv:Header>
      <wsa:To>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpoint</wsa:
To>
      <wsa:Action>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpointAc
tion </wsa:Action>
            <wsa:MessageID>urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</wsa:MessageID>
      </soapenv:Header>
      <soapenv:Body>…</soapenv:Body>
</soapenv:Envelope>
```

6.4. WS-Addressing – RelatesTo

The WS-Addressing – RelatesTo header contains a URI value that must be the MessageID of a previously exchanged message. It specifies that there is a relationship between the current message and the message that is being identified. The “RelationshipType” attribute in this header specifies the type of correlation between the two messages being referenced.

6.4.1. Profile Conformance

6.4.1.1. SOAP Message Creators

Service Message Creator **MUST** include the WS-Addressing – RelatesTo Header value in every SOAP response Message. Service Message Creator **MUST** map the wsa:MessageID value from the SOAP Request message to the wsa:RelatesTo value.

6.4.2. Example

```
<soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:wsa="http://www.w3.org/2005/08/addressing"       >
      <soapenv:Header>
      <wsa:To>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpointRespon
se</wsa:To>
      <wsa:MessageID >urn:UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</wsa:MessageID>
      <wsa:Action>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpointRe
sponseAction</wsa:Action>
      <RelatesTo>urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</RelatesTo>
      </soapenv:Header>
      <soapenv:Body>
      …
      </soapenv:Body>
</soapenv:Envelope>
```

6.5. WS-Addressing – ReplyTo

The “WS-Addressing – ReplyTo” header block contains the endpoint to which reply messages should be sent. The ReplyTo header is required when the message is the request in an asynchronous request-reply exchange; it is optional in other cases. ReplyTo headers rely on a WS-Addressing-defined structure called an endpoint reference that bundles together the information needed to properly address a SOAP message.

6.5.1. Profile Conformance

6.5.1.1. SOAP Message Creators

SOAP Message Creators **MUST** include ReplyTo SOAP header to the message.
SOAP Message Creators **MUST** map WS-Addressing anonymous URI for all the request where ReplyTo address is not needed.
SOAP Message Creators **MUST** map a valid endpoint reference to the ReplyTo header value if the request is part of an Asynchronous request-reply exchange.

6.5.1.2. Service Provider

Service Provider **SHOULD** reject an Asynchronous SOAP Request if the message does not contain a valid endpoint reference mapped to ReplyTo header value.

6.5.2. Example

```
<soapenv:Envelope
     xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"       >
     <soapenv:Header>
     <wsa:To>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpointTo</ws
a:To>
     <wsa:MessageID >urn:UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</wsa:MessageID>
     <wsa:Action>http://ns.humanservices.gov.au/dhsoperation/bindingwsdl/2008/06/05/endpoint</
wsa:Action>
     <wsa:ReplyTo>
          <wsa:Address>http://reply.endpoint.reference/</wsa:Address>
     </wsa:ReplyTo>
     </soapenv:Header>
     <soapenv:Body> ... </soapenv:Body>
</soapenv:Envelope>
```

7. WEB SERVICES SECURITY

7.1. WS-Security 1.1

WS-Policy defines a framework for allowing Web services to express their constraints and requirements. Such constraints and requirements are expressed as policy assertions. WS-Security is an OASIS standard for specifying security and other security standard policy assertions. WS-Security describes how to attach signatures and encryption headers to SOAP messages. In addition, it describes how to attach security tokens, including binary security tokens such as X.509 certificates and Kerberos tickets, to messages.
WS-Security is OASIS Recommendation, more information can be found at [WSS2006 WSSPL2007].

7.1.1. Profile Conformance

7.1.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the use of WS-Security 1.1 to specify various parts of the SOAP messages to be included and signed.

7.2. TLS Basic Authentication

Transport Layer Security (TLS) is a protocol to provide security over the network. TLS provides the encryption capability for the applications at the transport layer. This profile uses the TLS version 1.0 standard.
TLS is a Standard RFC, more information can be found at [RFC2246].

7.2.1. Profile Conformance

This profile conforms to the WSI Basic Security profile version 1.1 [WSIBSP2010] Section 4 Transport Layer Mechanisms.

7.2.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the use of TLS for transport security.

7.2.1.2. Service Message Creator

Service Message Creator **MUST** support TLS v1.0 [RFC2246] or above.
Service Message Creator **MUST** establish connections using TLS basic authentication.

7.2.1.3. Service Message Consumer

Service Message Consumer **MUST** support TLS v1.0 [RFC2246] or above.

7.3. TLS Mutual Authenticated

TLS Mutual Authentication is an extension to the TLS basic authentication, where in the client is required to have a valid certificate and that certificate to be presented to the service provider for authentication and authorisation purpose.

7.3.1. Profile Conformance

The profile includes those standards from the **7.2 TLS Basic Authentication** and additionally mandates the use of the following:

7.3.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the use of TLS for transport security.

7.3.1.2. Service Message Creator

Service Message Creator **MUST** support TLS v1.0 [RFC2246] or above.
Service Message Creator **MUST** establish connections using TLS mutual authentication.

7.3.1.3. Service Message Consumer

Service Message Consumer **MUST** support TLS v1.0 [RFC2246] or above.
Service Message Consumer **MUST NOT** accept connections which are not mutually authenticated.

7.4. XML Digital Signature

Digital signature provides integrity, message authentication and signer authentication. XML Digital signature specification [XSSP2008] defines signature creation and verification rules and syntax.

7.4.1. Profile Conformance

This profile confirms to the WSI Basic Security Profile 1.1 [WSIBSP2010] section 9.0 XML-Signature except where referenced.

7.4.2. Types of Signature Profile Conformance

7.4.2.1. SOAP Message Creators

SOAP Message Creators **MUST NOT** use Enveloping Signature as defined by the XML Signature specification [XSSP2008].
SOAP Message Creators **MUST NOT** use Enveloped Signature as defined by the XML Signature specification [XSSP2008].
SOAP Message Creators **MUST** use Detached Signature as defined by the XML Signature specification [XSSP2008].

7.4.2.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the Signature type used is not detached.

7.4.3. Signed Element References Profile Conformance

7.4.3.1. SOAP Message Creators

SOAP Message Creators **MUST** create a “wsu:Id” attribute for referencing for every signed element.
SOAP Message Creators **MUST r**eference signed elements with “wsu:Id” attribute only as defined by the XML Signature specification [XSSP2008].
The Signature Reference **MUST** contain a URI attribute referencing the Signed Element in the following format: URI=”#{wsu:Id value}”.

7.4.3.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the Signature reference used is not in the “wsu:Id” format.

7.4.4. Canonicalization Method Algorithm Profile Conformance

7.4.4.1. SOAP Message Creators

SOAP Message Creators **MUST** use “http://www.w3.org/2001/10/xml-exc-c14n#” algorithm for Canonicalization Method during signing process as defined by the XML Signature specification [XSSP2008].

7.4.4.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the algorithm used for Canonicalization Method during signing process is not "http://www.w3.org/2001/10/xml-exc-c14n#"

7.4.5. Signature Method Algorithm Profile Conformance

7.4.5.1. SOAP Message Creators

SOAP Message Creators **MUST** use either "http://www.w3.org/2000/09/xmldsig#rsa-sha1" algorithm for Signature Method during signing process as defined by the XML Signature specification [XSSP2008] or "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" as defined by XML Signature Syntax and Processing Version 2.0.

7.4.5.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the algorithm used for Signature Method during signing process is not "http://www.w3.org/2000/09/xmldsig#rsa-sha1" or "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256".

7.4.6. SignedInfo Reference Method Transforms Profile Conformance

7.4.6.1. SOAP Message Creators

SOAP Message Creators **MUST** have exactly one transform with the algorithm for SignedInfo Reference Method Transforms set to "http://www.w3.org/2001/10/xml-exc-c14n#" as defined by the XML Signature specification [XSSP2008].

7.4.6.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if there is none or more than one SignedInfo Reference Method Transforms algorithm set to "http://www.w3.org/2001/10/xml-exc-c14n#".

7.4.7. SignedInfo Reference DigestMethod Profile Conformance

7.4.7.1. SOAP Message Creators

SOAP Message Creators **MUST** have one or more SignedInfo/Reference/DigestMethod value set to "http://www.w3.org/2000/09/xmldsig#sha1" as defined by the XML Signature specification [XSSP2008] or "http://www.w3.org/2001/04/xmlenc#sha256" as defined by XML Signature Syntax and Processing Version 2.0.

7.4.7.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if SignedInfo/Reference/DigestMethod is not "http://www.w3.org/2000/09/xmldsig#sha1" or "http://www.w3.org/2001/04/xmlenc#sha256".

7.4.8. KeyInfo Profile Conformance

7.4.8.1. SOAP Message Creators

SOAP Message Creators **MUST** have only one SecurityTokenReference element as defined by the XML Signature specification [XSSP2008].

SOAP Message Creators **MUST** use the BinarySecurityToken SecurityTokenReference mechanism described below:

1. BinarySecurityToken
   a) Reference URI must refer to the BinarySecurityToken wsu:Id attribute
   b) Reference ValueType must be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"

c) The BinarySecurityToken EncodingType must be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"

d) The BinarySecurityToken ValueType must be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"

e) Example

```
<wsse:Security
     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <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="X509-86CFEEBEBBC882547713439705419372 5">...</wsse:BinarySecurityToken>
   <ds:Signature
     wsu:Id="SIG-172"
     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:SignedInfo>...</ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo wsu:Id="KI-86CFEEBEBBC88254771343970541937 26">
         <wsse:SecurityTokenReference
          wsu:Id="STR-86CFEEBEBBC882547713439705419372 7">
            <wsse:Reference
          URI="#X509-86CFEEBEBBC882547713439705419372 5"
          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>
```

7.4.8.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if there are zero or more than one SecurityTokenReference elements. Service Provider **MUST** respond with a SOAP-Fault if the SecurityTokenReference used is not one of the two styles listed above.

7.5. Signed Timestamp

WS-Security defines Timestamp as optional. However, this profile defines the WS-Security timestamp as a mandatory element.

7.5.1. Profile Conformance

This profile conforms to the WSI Basic Security profile version 1.1 [WSIBSP2010] Section 7 Timestamps. Please refer to [WSIBSP2010] for more information. The following additional conformance points should be noted.

7.5.1.1. SOAP Message Creators

SOAP Message Creators **MUST** sign the timestamp header.

SOAP Message Creators **MUST** ensure that that the created- (and if applicable, the expires-) timestamp is within the maximum clock skew allowance of +/-5 minutes.

SOAP Message Creators **MUST** use the XML Digital signature process specified in this document for signing the timestamp header.

7.5.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if timestamp element is not present.

Service Provider **MUST** account for the maximum allowable clock skew of +/-5 for the created (and if applicable, expires element) timestamp when processing.

Service Provider **MUST** respond with a SOAP-Fault if timestamp element is not signed using the XML Digital Signature processes from this document.

Service Provider **MUST** respond with a SOAP-Fault if timestamp element has incorrect values as per WSI Basic Security Profile 1.1 [WSIBSP2010].

7.6. Signed Payload

The payload in this document is referring to the SOAP:Body element. All external services with this policy are required to sign the SOAP:Body element.

7.6.1. Profile Conformance

7.6.1.1. SOAP Message Creators

SOAP Message Creators **MUST** sign the SOAP:Body element.

SOAP Message Creators **MUST** use the XML Digital signature process specified in this document for signing the SOAP:Body element.

7.6.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the SOAP:Body element is not signed using the XML Digital Signature defined this document.

7.7. WS-Username Token

The WS-Security [WSS2006 WSSPL2007] defines wsse:UsernameToken as an optional element to be used where username based authentication, authorisation is required. This profile specifies the use of wsse:UsernameToken as a mechanism to increase the authentication strength rating. It could be used for those services that require a higher security authentication rating then what is provided by organisation signing and a simple user header.

7.7.1. Profile Conformance

This profile conforms to the WSI Basic Security profile version 1.1 [WSIBSP2010] Section 12 Username Token. Please refer to [WSIBSP2010] for more information. The following additional conformance points should be noted.

7.7.1.1. SOAP Message Creators

SOAP Message Creators **MUST** include the UsernameToken header as part of the WS-Security header

SOAP Message Creators **MUST** use Password digest as the only method for including the password.

SOAP Message Creators **MUST** include Nonce.

SOAP Message Creators **MUST** include Created.

7.7.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the UsernameToken element is not present.
Service Provider **MUST** respond with a SOAP-Fault if the UsernameToken element has incorrect values as per WSI Basic Security Profile 1.1 [WSIBSP2010].
Service Provider **MUST** respond with a SOAP-Fault if the UsernameToken password methods is not Password Digest.
Service Provider **MUST** respond with a SOAP-Fault if the UsernameToken username or password are incorrect.
Service Provider **MUST** respond with a SOAP-Fault if the UsernameToken Nonce or Created are not supplied.
Service Provider **MUST** remember recently used Nonces and respond with a SOAP-Fault for UsernameTokens with a used Nonce.

7.8. Non-repudiation Logging

Non-repudiation is a way to guarantee that the sender of the message cannot later deny having sent the message or the content of the message.
Non-repudiation is obtained by the use of “Signed Payload” profile.

7.8.1. Profile Conformance

7.8.1.1. SOAP Message Creators

SOAP Message Creators **MUST** use the digitally sign the SOAP:Body element as explained in the section “XML Digital Signature”

7.8.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the SOAP:Body element is not signed.
Service Provider **MUST** respond with a SOAP-Fault if the SOAP:Body digital signature is incorrect.
Service Provider **MUST** log the complete SOAP:Envelope without making any modifications to it.

7.9. Signed WSA Headers

WSA Headers provide transport – neutral mechanisms to address web service resources and to secure end-to-end endpoint identification in messages. Signing these headers will improve the security and increase the message reliability. All external parties implementing the signature profile are required to confirm to these policy.

7.9.1. Profile Conformance

7.9.1.1. Service Interface Specification

Service Interface Specification **MUST** specify the WSA headers to be signed as part of the specification.

7.9.1.2. SOAP Message Creators

SOAP Message Creators **MUST** use the XML Digital signature process specified in this document for signing the WSA elements.
SOAP Message Creators **MUST** sign the message parts in accordance to the WSDL. Signing of the wsa:MessageID element is mandatory and other wsa addressing elements is strongly recommended.

7.9.1.3. Service Provider

Service Providers **MUST** respond with a SOAP-Fault if elements are not signed in accordance with policies expressed in the WSDL.
Service Provider **MUST** respond with a SOAP-Fault if the wsa:MessageID element is not signed.

8. GENERIC WEB SERVICES STANDARDS

8.1. General Naming Standards

8.1.1. General XML Naming Types

Several XML naming schemes have been used in standardizing XML names, some of which are borrowed from programming languages. The major characteristics of these schemes are the use of upper- and lowercase letters, the delimitation of words or abbreviations, and how abbreviations, if required, are implemented.
There are four main options for how to use casing within your XML specifications:

1. Upper Camel Casing
2. Lower Camel Casing
3. Lower Casing
4. Upper Casing

8.1.2. Upper Camel Casing

Upper Camel Casing capitalizes the first letter of words or word parts thereafter to create the element names. Examples of Upper Camel Casing are:

1. CustomerName
2. LineItem
3. ShippingAddress
4. Age

8.1.3. Lower Camel Casing

Lower Camel Casing capitalizes the first letter of every word except the first word. Java uses lower camel casing. Here are some examples of Lower Camel Casing:

1. firstName
2. socialSecurityNumber
3. birthDate
4. weight

8.1.4. Lower Casing

Lower casing removes any question about whether a specific word or abbreviation should be capitalized by forcing everything to go lower case. Here are some examples of Lower Casing:

1. cancelorder
2. trackingnumber

8.1.5. Upper Casing

Upper casing removes any question about whether a specific word or abbreviation should be lowered by forcing everything to go upper case. Here are some examples of Upper Casing:

1. ORDERNUMBER
2. SHIPMENTID

8.1.6. Abbreviation

Many of the words used in XML entities are quite long, and often, a series of several words makes up a single XML entity. To help reduce the burden of typing long entity names many people elect to use abbreviations, but without a set standard it will quickly become out of control. For example, customer can be abbreviated as cust or cst.
It is recommended not to use any abbreviations.

8.1.7. Prefix & Suffix

It is recommended to use the following suffix / prefix where necessary.

Type	Prefix	Example
Date	xxxxDate	startDate endDate
Name	xxxxName	FirstName CompanyName
Type	xxxxType	NameType OrganisationType
Boolean	xxxInd	statusInd

8.2. Naming Standards

This section defines some elements of an enterprise-wide XML naming scheme to be used in DHS artefacts.
Name Syntax: A **Name** should be made up of one or more words, with no spaces, hyphens or underscores between them.

8.2.1. General Artefacts

Rule	**Example**
Folder and File structure	
Abbreviations in names should be avoided. XSD, WSDL, XSL names must be written in Upper Camel Case XML file names must be written in Upper Camel Case.	PingRequestMessageInternal.xsd EsbPingBinding.wsdl TransformExternalCommonHeader.xsl ReadProviderRequest.xml
Namespaces	
Namespace must be written in lower case Namespaces should be URLs Use of abbreviations must be avoided Date stamp if used should follow ‘yyyy/mm/dd’ format	http://ns.humanservices.gov.au/dhsoperation/schema/2008/07/22/webservices.xsd http://ns.humanservices.gov.au/dhsoperation/wsdlinterface/2008/07/22/interfaces.wsdl http://ns.humanservices.gov.au/dhsoperation/wsdlinterface/2008/07/22/bindings.wsdl http://ns.humanservices.gov.au/policy/2008/07/22/securityalgorithm.xml

8.2.2. XML Schema Data Types

Rule	**Example**
Simple Types	
Simple types must be written in Upper Camel Case	StreetAddress

Rule	Example
	PostCode Country Surname PhoneNumber
Complex Types	
Complex Types must be written in Upper Camel Case	PostalAddress RegistrationDetails
Elements	
Element declaration names must be written in Lower Camel Case.	businessPhone profileDetails request response

8.2.3. WSDL Types

Rule	Example
Policy Name	
Names representing Policy must be written in Upper Camel Case. Name should clearly identify what policy is being enforced.	SecurityAlgorithmsPolicy SecurityCertificateTransmissionPolicy
Policy ID	
Names representing Policy ID must be written in Upper Camel Case.	SecurityAlgorithmsPolicy SecurityCertificateTransmissionPolicy
Message	
Names representing WSDL Message must be written in Lower Camel Case.	customerDetails addressDetails postalAddress
Message Part	
Names representing WSDL Message Part names must be written in Lower Camel Case.	address accountInformation
Port Type	
Names representing WSDL Port Type must be written in Upper Camel Case.	OrganizationalAccountPortType IndividualAccountPortType
Port Type Operation	
Port Type operation names MUST be in written Lower Camel Case. Operation name SHOULD follow CRUD+S rule. CRUD+S Rule: Where possible operations names should start with one of Create, Update, Read, Delete, or Search.	addAccount modifyUser deleteCustomer
Operation Input	
Operation Input names must be written in Lower Camel Case The local input message should match the message operation name.	inMessage createAccountCutomerDetails accountData
Operation Output	
Operation Output names must be written in Lower Camel Case Operation Output names should be the same as input with word Response appended.	serviceReponse createAccountResponse
Operation Fault	

Rule	Example
Operation fault names must be written in Lower Camel Case	standardError
Binding	
Binding names must be written in Upper Camel Case	OrganizationalAccountHTTPBinding IndividualAccountHTTPBinding
Service Name	
Service names must be written in Upper Camel Case.	IndividualAccountPorts OrganisationalAccountPorts
Service Port	
Service Port names must be written in Upper Camel Case.	IndividualAccountPort OrganisationalAccountPort

8.3. Namespace Naming Standards

The XML Namespace standard is described in the Core messaging standards section earlier in the document. This section will describe the DHS XML Namespace naming standards.
The DHS XML Namespace value is divided into 6 parts.
http://[CompanyName]/<<SystemName>>/<<ServiceName>>/[ArtefactType]/[VersionInformation]/[ArtefactName]

e.g. http://ns.humanservices.gov.au/dhs/dhsoperation/schema/2008/07/22/webservices.xsd

Company hostname: [Mandatory part] The artefact providing company hostname. e.g. ns.humanservices.gov.au
System Name: [Optional part] The name of the system under which the artefact is developed
Service Name: [Optional part] Artefacts are developed to support a service specification or implementation. The service name part should specify the service for which this artefact belongs to.
Artefact Type: [Mandatory part] There are various types of artefacts e.g.

1. Schema
2. Policy
3. WSDL Interface
4. WSDL Binding

Version Information: [Mandatory part] Version information specifies a particular version of an artefact. The artefact versioning in this profile should use date stamp {yyyy/mm/dd} pattern. e.g. 2009/02/13.
Artefact name: [Mandatory part] This part of the namespace specifies the name of the artefact e.g. dhs_schema.xsd register_interface.wsdl etc.

8.3.1. General Naming Standards

1. Namespace must be in all lower case.
2. The Namespace should be a URI
3. Use of abbreviations must be avoided.
4. Date stamp if used should follow ‘yyyy/mm/dd’ format

8.3.2. Profile Conformance

8.3.2.1. Service Interface Specification

Service Interface Specification **SHOULD** use the above mentioned DHS Namespace values pattern.

8.3.2.2. SOAP Message Creators

SOAP message creators **SHOULD** use the namespace standards as specified above.

8.3.2.3. SOAP Message Consumers

SOAP message consumers **SHOULD** reject the message if the namespaces does not comply with the above standard.

8.3.3. Example

```
http://ns.humanservices.gov.au/dhsoperation/schema/2008/07/22/webservices.xsd
http://ns.humanservices.gov.au/dhsoperation/wsdlinterface/2008/07/22/interfaces.wsdl
http://ns.humanservices.gov.au/dhsoperation/wsdlinterface/2008/07/22/bindings.wsdl
http://ns.humanservices.gov.au/policy/2008/07/22/securityalgorithm.xml
```

8.4. Service Versioning

Software component versioning is a common problem in the design of any distributed system and Web services are unfortunately no exception. This section will outline the scope of the versioning difficulties facing Web services and provide some best practices for addressing the problem.
There are two types of Web Service changes that can happen,

1. Backwards-compatible changes
2. Non-backwards-compatible changes

8.4.1. Backward Compatible Changes

There are 2 types of changes which can be considered as backward compatible change.
New operation in the WSDL: If a new operation is added to an existing WSDL and if existing requestors are unaware of a new operation, then they will be unaffected by its introduction.
New XML Schema Types in the WSDL: If a new XML schema element / complex / simple type is added to the existing WSDL and it is not used by any existing messages or bindings then this change can be considered as a backward compatible change and it won't affect any existing users.
Even though a backwards compatible change to a WSDL does not affect any existing users it is recommended that the following best practices be followed:

1. Every new edition of a WSDL document is stored using the version-control process with a different namespace.
2. XML comments are used to indicate unique version information or a version history.

8.4.2. Non - Backward Compatible Changes

Non – Backward compatible change can be described as any change impacting existing users.

1. Removing an operation
2. Renaming an operation
3. Changing the parameters (in data type or order) of an operation
4. Changing the structure of a complex data type

Most of these problems can be solved by making use of XML namespaces to clearly delineate the versions of a document that are compatible. Sending along specific namespace value with every SOAP message and result will allow Web Service implementation to correctly determine what version of the message type has been sent by the Web Service Consumer (SOAP Requester).
If non-backwards-compatible changes need to be made to a WSDL document, then the first step is to ensure that the namespace for the XML elements resulting from that document is unique. To ensure that the various editions of a WSDL

document are unique, the recommendation is to use the naming scheme described in the section "8.2 Naming Standards". This follows the general guidelines given by the W3C for XML namespace definition and an XML type definition might look something like the following:

8.4.3. Example

```
<xsd:schema
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:tns="http://ns.humanservices.gov.au/dhsoperation/schema/2008/07/22/webservices.xsd"
     targetNamespace="http://ns.humanservices.gov.au/dhsoperation/schema/2008/07/22/webservice
s.xsd"
     elementFormDefault="unqualified"
     attributeFormDefault="unqualified">
          <xsd:element name="pingRequest" type="tns:pingRequest"/>
          <xsd:element name="pingResponse" type="tns:pingResponse"/>
          <xsd:complexType name="PingRequest">
               <xsd:sequence>
                    <xsd:element
                         name="inputParam"
                         nillable="true" type="xsd:string"/>
               </xsd:sequence>
          </xsd:complexType>
          <xsd:complexType name="PingResponse">
               <xsd:sequence>
                    <xsd:element
                         name="outputParam"
                         nillable="true" type="xsd:string"/>
               </xsd:sequence>
          </xsd:complexType>
</xsd:schema>
```

8.5. Payload Structure

Setting up a payload structure standard early in the project and conforming to that standard will help in creating and maintaining interfaces and specifications in the later stages of the project. There are 2 topics of interest when discussing payload structure.

1. **Request Payload**: The request payload should follow the document / literal and wrapped convention with no other special structural requirements.
2. **Response Payload**: A response message can be a
    a. **Valid Scenario**: The service has performed the action it is intended for. The response should be a schema valid response as defined by the WSDL.
    b. **Fault Scenario**: The service has failed to perform the action. The response should be a valid SOAP Fault message. Service Message Pattern should be used to represent the Fault details.

8.5.1. Service Message Pattern

Service Message Pattern adds reliability to understanding the Web Service fault scenarios by adding a generic service message node to fault details node. This structure is described below.
Service Message Fault Details Node: The service message fault detail node can have 1 or more service messages per message. The HighestServerity element refers to the highest severity service message code.

```
<xsd:complexType name="ServiceMessage">
     <xsd:sequence>
          <xsd:element name="code" type="xsd:string" nillable="false"/>
          <xsd:element name="severity" type="xsd:string" nillable="false"/>
          <xsd:element name="reason" type="xsd:string" nillable="false"/>
          <xsd:element name="details" nillable="true"/>
     </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="ServiceMessages">
     <xsd:sequence>
          <xsd:element name="highestSeverity" type="xsd:string"/>
          <xsd:element name="serviceMessage" type="tns:ServiceMessage"
               maxOccurs="unbounded"/>
     </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="BusinessResponse">
     <xsd:sequence>
          <xsd:element name="resultSize" type="xsd:string" nillable="true"/>
          <xsd:element name="results" type="tns:Record" nillable="true"
               minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
</xsd:complexType>
```

8.5.2. Profile Conformance

8.5.2.1. Service Interface Specification

Service Interface Specification **SHOULD** use the service message pattern while describing the service faults in the interfaces.

8.5.2.2. SOAP Message Creators

SOAP message creators **SHOULD** understand and process Service Messages in the SOAP-Faults.

8.5.2.3. Service Providers

Service Provider **SHOULD** capture all error scenarios and map them to Service Messages element and add that node to the SOAP Fault detail node.

9. DHS WEB SERVICES STANDARDS

9.1. DHS PKI

Public Key Infrastructure, commonly referred to as PKI is an Information Technology (IT) infrastructure that enables the secure exchange of data. PKI is a set of software tools, hardware, network services and management techniques (policy and procedures) that work together to provide a web of 'trust'.
PKI uses asymmetric cryptographic algorithms, which use a set of keys “key pair” mathematically related to each other.
DHS PKI is based on the Australian Gatekeeper framework and conformed to the International Organisation for Standardisation (ISO): Health Informatics - Public Key Infrastructure technical specification (ISO/TS 17090).
PKI has been adopted by the Australian Government to provide a robust system of security for online health transactions.
For DHS programs, the administration process of DHS PKI is the responsibility of Medicare Australia program.
PKI helps implementing the following for Web services:

1) **Authentication**: Who sent the message
2) **Integrity**: The message content has not been altered in any way between sender and receiver.
3) **Non-Repudiation**: The sender cannot at some later stage dispute the sending the message and its contents.
4) **Confidentiality**: That only the person the message is directed to can open it.

9.1.1. Profile Conformance

9.1.1.1. SOAP Message Creators

SOAP message creators **MUST** use DHS issued current valid X509 v3 Certificate to sign the message.
SOAP message creators **MUST** use certificate with “Key Usage” set to “Digital Signature (80)” for signing the message.

9.2. DHS Product Header

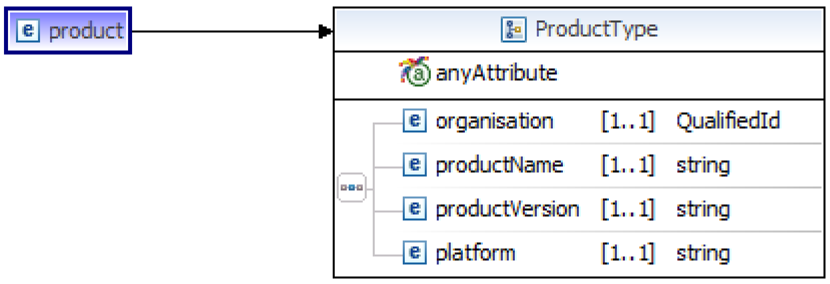
The Product Header is a standard that identifies the software component that is initiating a service request. The product header supports the following operational processes:

- SOA Governance
- Software certification
- Service level management.

All software products that will be invoking DHS services will be accredited to ensure it complies with the Web Service standards for the services that it is being used to access. A certified product will be registered by DHS and the accreditation details to be included in the product header will be issued by DHS.

9.2.1. Complex Type ProductType

The **ProductType** complex type defines the following elements:


product
ProductType
anyAttribute
organisation [1..1] QualifiedId
productName [1..1] string
productVersion [1..1] string
platform [1..1] string

Element Name	Type	Required	Desorption
organisation	QualifierId	Mandatory	A name of complex type QuailifiedId. This identifies the organisation that provides the software product.
productName	String	Mandatory	The name of the software product
productVersion	String	Mandatory	The version of the software product
Platform	String	Mandatory	The software platform used by the software product

The following schema fragment defines the **ProductType** complex type:

```
<xsd:complexType name="ProductType">
     <xsd:sequence>
          <xsd:element name="organisation" type="mgqi:QualifiedId" minOccurs="1"
maxOccurs="1" />
          <xsd:element name="productName" type="xsd:string" minOccurs="1" maxOccurs="1" />
          <xsd:element name="productVersion" type="xsd:string" minOccurs="1" maxOccurs="1"
/>
          <xsd:element name="platform" type="xsd:string" minOccurs="1" maxOccurs="1" />
     </xsd:sequence>
     <xsd:anyAttribute/>
</xsd:complexType>
```

9.2.1.1. Example

```
<ce:product wsu:Id="id-15" xmlns:ce="
http://ns.humanservices.gov.au/common/schema/2014/03/15/elements" xmlns:cqi="
http://ns.humanservices.gov.au/common/qualifiedidentifier/schema/2014/03/15">
    <ce:organisation>
       <cqi:qualifier>
           http://ns.humanservices.gov.au/qualifier/organisation
       </cqi:qualifier>
       <cqi:id>DHSClientOrg</cqi:id>
    </ce:organisation>
    <ce:productName>DHSClient</ce:productName>
    <ce:productVersion>v1.0</ce:productVersion>
    <ce:platform>windows soapui</ce:platform>
</ce:product>
```

9.2.2. Profile Conformance

9.2.2.1. Service Interface Specification

Service Interface Specification **MUST** specify the product header as a required element for the SOAP messages of all operations.

9.2.2.2. Message Creator

Message Creator **MUST** include the product header for every SOAP request message.
Message Creator **SHOULD** choose one of the ‘organisation.qualifier’ from the list provided in the appendix section “12 - Appendix – C – Qualifier Identifiers types”..
Message Creator **SHOULD** set the product header values as provided by DHS.

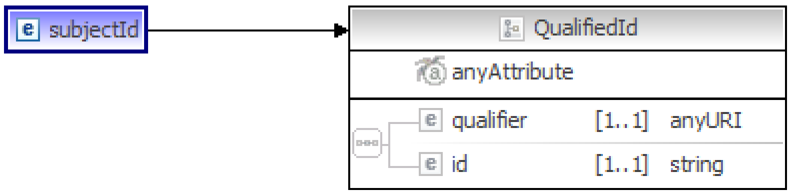
9.2.2.3. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the SOAP request message does not contain a product header.
Service Provider **MUST** validate the product header and respond with a SOAP-Fault if the message contains an unregistered product details.
Service Provider **MUST** validate the product header and respond with a SOAP-Fault if the product is not certified to use the Web service they are calling.

9.3. DHS SubjectId Header

The subjectId Token is included in the web services requests to identify the user entity (DHS user) that the web services request relates to. This header element is optional and will be only used if there is a need for passing the user entity information.

9.3.1. Element subjectId



The **subjectId** element is of complex type **QualifiedId.**
The following schema fragment defines the **subjectId** element:

```
<xsd:element name="subjectId " type="cqi:QualifiedId" />
```

9.3.1.1. Example

```
<ce:subjectId wsu:id="id-serviceProvider-01" xmlns:ce="
http://ns.humanservices.gov.au/common/schema/2014/03/15/elements" xmlns:cqi="
http://ns.humanservices.gov.au/common/qualifiedidentifier/schema/2014/03/15">
<cqi:qualifier>http://ns.humanservices.gov.au/qualifier/organisation/mca</cqi:qualifier>

    <cqi:id>c7906754-010c-e4a8-3d5d-bab57d9a16c3</cqi:id>

</ce:subjectId >
```

9.3.2. Profile Conformance

9.3.2.1. Service Interface Specification

Service Interface Specification **MUST** specify the subjectId header as a required element if communicating with external providers and Service provider based routing needs to be done for the SOAP messages.

9.3.2.2. Service Invoker

Service Invoker **MUST** include the subjectId header for SOAP request message if this header is defined in the service interface specification.
Service Invoker **SHOULD** choose one of the ‘subjectId.qualifier’ from the list provided in the appendix section “12 - Appendix – C – Qualifier Identifiers types”.

9.3.2.3. Service Provider

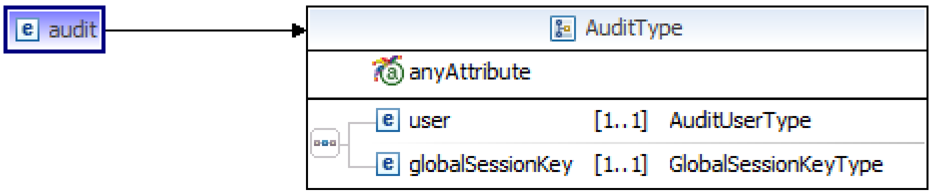
Service Provider **MUST** respond with a SOAP-Fault if the SOAP request message does not contain a subjectId header.

9.4. DHS Audit Header

The Audit header is included in the web services requests to include the user and logging attributes that the web services request relates to.

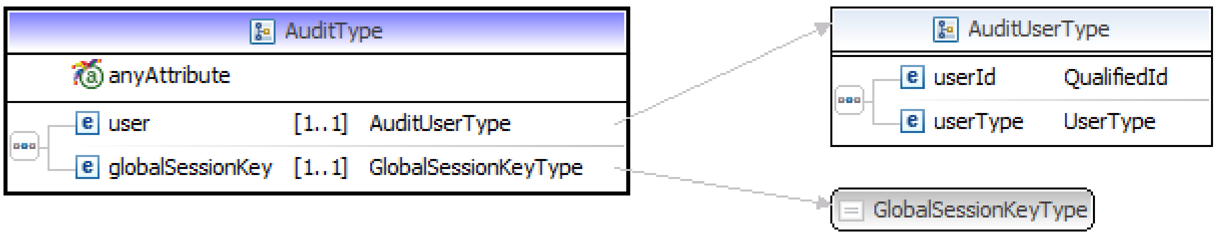
9.4.1. Element <audit>

The <audit> element is the root element for a DHS audit header.


audit
AuditType
anyAttribute
user [1..1] AuditUserType
globalSessionKey [1..1] GlobalSessionKeyType

Element Name	Type	Required	Desorption
Audit	AuditType	Mandatory	Describes the audit context that is associated a service request.

9.4.2. Complex Type <AuditType>



The **AuditType** complex type defines the following elements:

Element Name	Type	Required	Desorption
User	AuditUserType	Mandatory	User describes the service client in the logical web services interaction
globalSessionKey	GlobalSessionKeyType	Mandatory	A unique identifier that can be used to correlate audit logs. This identifier is unique to logical session and spans any physical sessions created within the same logical session. See section **9.4.4 Global Session Key**.

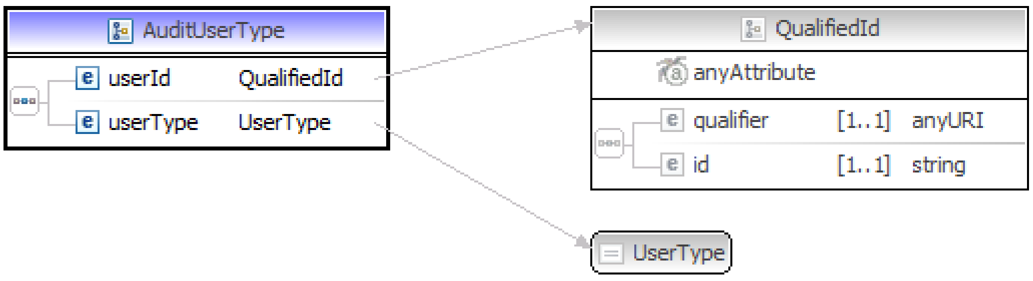
The following schema fragment defines the **AuditType** complex type:

```
<xsd:complexType name="AuditType">
     <xsd:sequence>
          <xsd:element name="user" type="tns:AuditUserType" minOccurs="1" maxOccurs="1" />
          <xsd:element name="globalSessionKey" type="cdt:GlobalSessionKeyType" minOccurs="1"
maxOccurs="1" />
     </xsd:sequence>
     <xsd:anyAttribute />
```

```
</xsd:complexType>
```

9.4.3. Complex Type <AuditUserType>

The **AuditUserType** complex type describes a user and contains the qualified user identifier and type of a user. It defines the following attributes and elements:



Element Name	Type	Required	Desorption
userID	QualifiedId	Mandatory	User describes the service client in the logical web services interaction
userType	UserType	Mandatory	The User Type of the Subject. User types are defined in Section **Error! Reference source not found. Error! Reference source not fo und.**

The following schema fragment defines the **UserSubjectType** complex type.

```
<xsd:complexType name="AuditUserType">
     <xsd:sequence>
          <xsd:element name="userID" type="mgqi:QualifiedId" />
          <xsd:element name="userType" type="cdt:UserType" />
     </xsd:sequence>
</xsd:complexType>
```

9.4.4. Global Session Key

The Global Session Key (GSK) is a globally unique identifier that can be used for the purposes of correlating audit logs. This should be treated as an opaque value by the Receiver and included in all audit events.
The Sender is responsible for ensuring that the GSK is globally unique across all the parties. It is recommended that the GSK be based on a UUID in the following format:

<prefix><uuid>

The prefix is a 4 character code unique to the Sender. The Sender may choose to use an alternative format to a uuid as defined by the RFC 4122.
If the generated uuid can be guaranteed to be globally unique the sender prefix is redundant. In this case the following standard format should be used:

uuid:<uuid>

The receiver should treat the GSK as an opaque value, the above standard formats are documented to ensure the GSK will be unique across senders. In general, a receiver should assume that the GSK that may have a maximum length 45 chars to ensure that existing auditing systems are not impacted.

9.4.5. Sample DHS Audit Header

```
Staff Access:
A DHS Audit Header for a DHS staff member accessing a DHS client application.
<ce:audit
     xmlns:ce="http://ns.humanservices.gov.au/common/schema/2014/03/15/elements"
     xmlns:cqi="http://ns.humanservices.gov.au/common/qualifiedidentifier/schema/2014/03/15">
     <ce:user>
          <ce:userID>
               <cqi:qualifier>http://ns.humanservices.gov.au/loginid/1.0</cqi:qualifier>
               <cqi:id>p12345 </cqi:id>
          <ce:userID>
         <ce:userType>STAFF</ce:userType>
     </ce:user>
     <ce:globalSessionKey>uuid:a40d132b-7179-403c-a9cd-c73146e47ce0</ce:globalSessionKey>
</ce:audit>
```

```
A DHS Audit Header for a DHS user accessing a DHS client application.
<ce:audit
     xmlns:ce="http://ns.humanservices.gov.au/common/schema/2014/03/15/elements"
     xmlns:cqi="http://ns.humanservices.gov.au/common/qualifiedidentifier/schema/2014/03/15">
     <ce:user>
          <ce:userID>
               <cqi:qualifier>http://ns.humanservices.gov.au/loginid/1.0</cqi:qualifier>
               <cqi:id>p12345 </cqi:id>
          <ce:userID>
         <ce:userType>USER</ce:userType>
     </ce:user>
     <ce:globalSessionKey>uuid:a40d132b-7179-403c-a9cd-c73146e47ce0</ce:globalSessionKey>
</ce:audit>
```

9.4.6. Profile Conformance

9.4.6.1. Service Interface Specification

Service Interface Specification **MUST** specify the audit header as a required element for the SOAP messages of all operations.

9.4.6.2. Message Creator

Message Creator **MUST** include the audit header for every SOAP request message.
Message Creator **SHOULD** choose one of the ‘userID.qualifier’ from the list provided in the appendix section “12 - Appendix – C – Qualifier Identifiers types”..
Message Creator **MUST** set the globalSessionKey values in the UUID format as described above.

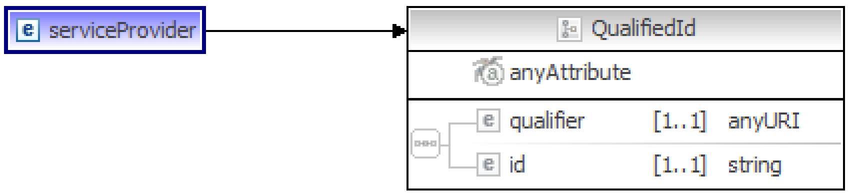
9.4.6.3. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the SOAP request message does not contain a audit header.

9.5. DHS Service Provider Header

The Service Provider header is included in web services requests to identify the DHS Service Provider that the service request relates to. This header element is optional and will be only used for services where service provider based dynamic routing is required.

9.5.1. Element serviceProvider



The **serviceProvider** element is of complex type **QualifiedId.**
The following schema fragment defines the **serviceProvider** element:

```
<xsd:element name="serviceProvider" type="cqi:QualifiedId" />
```

9.5.1.1. Example

```
<ce:serviceProvider wsu:id="id-serviceProvider-01" xmlns:ce="
http://ns.humanservices.gov.au/common/schema/2014/03/15/elements" xmlns:cqi="
http://ns.humanservices.gov.au/common/qualifiedidentifier/schema/2014/03/15">

    <cqi:qualifier>http://ns.humanservices.gov.au/qualifier/agency</cqi:qualifier>

    <cqi:id>CSA</cqi:id>

</ce:serviceProvider>
```

9.5.2. Profile Conformance

9.5.2.1. Service Interface Specification

Service Interface Specification **SHOULD** specify the service provider header as a required element if communicating with external providers and Service provider based routing needs to be done for the SOAP messages.

9.5.2.2. Service Invoker

Service Invoker **MUST** include the service provider header in the SOAP request message.
Service Invoker **SHOULD** choose one of the 'serviceProvider.qualifier' from the list provided in the appendix section "12 - Appendix – C – Qualifier Identifiers types".
Service Invoker **SHOULD** set the service provider header values as provided by RF.

9.5.2.3. Service Provider

Service Provider **SHOULD** respond with a SOAP-Fault if the SOAP request message does not contain a service provider header.

9.6. Signed Product Header

All external services with this policy are required to sign the ce:product element in the header.

9.6.1. Profile Conformance

9.6.1.1. Service Invoker

Service Invoker **MUST** sign the ce:product element.

Service Invoker **MUST** use the **XML Digital Signature** process specified in this document for signing the ce:product element.

9.6.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the ce:product element is not signed using the **XML Digital Signature** process defined in this document.

9.7. Signed SubjectID Header

All external services with this policy are required to sign the ce:subjectId element in the header.

9.7.1. Profile Conformance

9.7.1.1. Service Invoker

Service Invoker **MUST** sign the ce:subjectId element.

Service Invoker **MUST** use the **XML Digital Signature** process specified in this document for signing the ce:subjectId element.

9.7.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the ce:subjectId element is not signed using the **XML Digital Signature** process defined in this document.

9.8. Signed Audit Header

All external services with this policy are required to sign the ce:audit element in the header.

9.8.1. Profile Conformance

9.8.1.1. Service Invoker

Service Invoker **MUST** sign the ce:audit element.

Service Invoker **MUST** use the **XML Digital Signature** process specified in this document for signing the ce:audit element.

9.8.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the ce:audit element is not signed using the **XML Digital Signature** process defined in this document.

9.9. Signed ServiceProvider Header

All external services with this policy are required to sign the ce:serviceProvider element in the header.

9.9.1. Profile Conformance

9.9.1.1. Service Invoker

Service Invoker **MUST** sign the ce:serviceProvider element.

Service Invoker **MUST** use the **XML Digital Signature** process specified in this document for signing the ce:serviceProvider element.

9.9.1.2. Service Provider

Service Provider **MUST** respond with a SOAP-Fault if the ce:serviceProvider element is not signed using the **XML Digital Signature** process defined in this document.

10. APPENDIX – A – XML NAMING CONVENTION SUMMARY

Identifier Type	Rules for Naming	Examples
XML Schema Types		
XSD.SimpleType	Upper Camel Case	DestinationType IssuerType
XSD.ComplexType	Upper Camel Case	PostalAddress RegistrationDetails
XSD.ElementType	Lower Camel Case	readProvider createConsumer
XSD.Attribute	Lower Camel Case	name, id, firstName
WSDL Interface Types		
WSDL.Message	Lower Camel Case	customerDetails addressDetails postalAddress
WSDL.Message.part	Lower Camel Case	address accountInformation
WSDL.PortType	Upper Camel Case	OrganizationalAccountPortType IndividualAccountPortType
WSDL.PortType.Operation	Lower Camel Case	addAccount modifyUser deleteCustomer
WSDL.PortType.Operation.Input	Lower Camel Case	inMessage createAccountCutomerDetails accountData
WSDL.PortType.Operation.output	Lower Camel Case	outMessage serviceReponse createAccountResponse
WSDL.PortType.Operation.Fault	Lower Camel Case	badCutomerData badHeader faultyInformation
WSDL Binding Types		
WSDL.Binding	Upper Camel Case	OrganizationalAccountHTTPBinding IndividualAccountHTTPBinding
WSDL.ServiceName	Upper Camel Case	IndividualAccountPorts OrganizationalAccountPorts
WSDL.Service.Port	Upper Camel Case	IndividualAccountPort OrganizationalAccountPort

11. APPENDIX – B – SOAP FAULT

11.1. SOAP Roles

Role Name	URI
GATEWAY	http://ns.humanservices.gov.au/role/gateway
ESB	http://ns.humanservices.gov.au/role/esb
SERVICE	http://ns.humanservices.gov.au/role/service

11.2. Codes

Code	Type	Short Message	Long Message
ESB Request Validation			
badParam	**ERROR**	INVALID WEB SERVICE REQUEST	The web service request is not valid
badlyFormedMsg	**ERROR**	INVALID WEB SERVICE MESSAGE	The request did not contain the expected message format.
badTimestamp	**ERROR**	INVALID WEB SERVICE MESSAGE	The web services request timestamp in the web service message is not valid.
badWsaAction	**ERROR**	INVALID WEB SERVICE MESSAGE	The web services request WS Addressing Action is not valid.
badWsaMessageId	**ERROR**	INVALID WEB SERVICE MESSAGE	The web services request WS Addressing Message ID is not valid.
badWsaTo	**ERROR**	INVALID WEB SERVICE MESSAGE	The web services request WS Addressing To is not valid.
ESB Security			
notAccredited	**ERROR**	INVALID ACCREDITATION DETAILS	Software not accredited
certificateSkiMissing	**ERROR**	INVALID SECURITY DETAILS	The provided security certificate is missing the SKI
certificateKeyUsage	**ERROR**	INVALID SECURITY DETAILS	The provided certificate usage is invalid
certificateUnidentified	**ERROR**	INVALID SECURITY DETAILS	The provided certificate is unidentified.
invalidCredentials	**ERROR**	INVALID SECURITY DETAILS	The request credentials provided are not valid.
notAuthenticated	**ERROR**	INVALID SECURITY DETAILS	User not authenticated
notAuthorised	**ERROR**	INVALID SECURITY DETAILS	User not authorised.
badSignature	**ERROR**	INVALID SECURITY DETAILS	The signature provided in the request is not valid
badCertificateTransmitted	**ERROR**	INVALID SECURITY DETAILS	The certificate provided in the request is not valid
badAlgorithmDigest	**ERROR**	INVALID SECURITY DETAILS	The security algorithm used for message hash calculation is not valid
badAlgorithmSignature	**ERROR**	INVALID SECURITY DETAILS	The security algorithm used for message signing is not valid
ESB Service Availability			
servicePermanentUnavailable	**ERROR**	INVALID WEB SERVICE REQUEST	Service requested is permanently unavailable.
serviceTemporaryUnavailable	**ERROR**	INVALID WEB SERVICE REQUEST	Service requested is temporarily unavailable.
ESB Unknown Errors			

Code	Type	Short Message	Long Message
internalError	**ERROR**	INTERNAL ERROR	An internal error occurred in the ESB

12. APPENDIX – C – QUALIFIER IDENTIFIERS TYPES

12.1. Generic Qualifiers

This set of qualifiers should be used where the party needs to be identified by a qualifier type.

Type	Description	URI
Distinguished Name	Certificate Distinguished Name	http://ns.humanservices.gov.au/distinguishedname/1.0
DHS Login Identifier	DHS User Login Identifier	http://ns.humanservices.gov.au/loginid/1.0
Vendor Identifier	Vendor identifier provided by DHS	http://ns.humanservices.gov.au/vendorid/1.0
DHS Card Number	DHS Card Number	http://ns.humanservices.gov.au/dhscardnumber/1.0
External Identifier	Identifier issued by external parties.	http://ns.humanservices.gov.au/externalid/1.0
DHS Organisation	An organisation recognised by DHS	http://ns.humanservices.gov.au/qualifier/organisation
Service Provider	The target Service provider	http://ns.humanservices.gov.au/qualifier/memberservice/1.0
CLK	Centrelink	http://ns.humanservices.gov.au/qualifier/memberservice/clk
CLK CRN	Centrelink CRN	http://ns.humanservices.gov.au/qualifier/memberservice/clk/crn
CLK CAN	Centrelink CAN	http://ns.humanservices.gov.au/qualifier/memberservice/clk/can
MCA	Medicare Consumer ID	http://ns.humanservices.gov.au/qualifier/memberservice/mca
MCA Card Number	Medicare Card number	http://ns.humanservices.gov.au/qualifier/memberservice/mca/card
CSA	Child Support Agency	http://ns.humanservices.gov.au/qualifier/memberservice/csa
CSA Consumer ID	Child Support Agency	http://ns.humanservices.gov.au/qualifier/memberservice/csa/csrn
ATO	Australian Taxation Office	http://ns.humanservices.gov.au/qualifier/memberservice/ato
EHR	National eHealth Record System	http://ns.humanservices.gov.au/qualifier/memberservice/ehr
DVA	Department of Veterans' Affairs	http://ns.humanservices.gov.au/qualifier/memberservice/dva
AusGOV	AusGOV	http://ns.humanservices.gov.au/qualifier/memberservice/ausgov
myGOV	myGov	http://ns.humanservices.gov.au/qualifier/memberservice/mygov
APDM	Australia Post Digital Mailbox	http://ns.humanservices.gov.au/qualifier/memberservice/apdm
DCA	Disability Care Australia	http://ns.humanservices.gov.au/qualifier/memberservice/dca
VIC_DHHS	HousingVic Online Services	http://ns.humanservices.gov.au/qualifier/memberservice/vic_dhhs
VICSRO	State Revenue Office Victoria	http://ns.humanservices.gov.au/qualifier/memberservice/vicsro

13. APPENDIX – D – REFERENCES

Abbreviated terms in square brackets used in this document refer to the corresponding document set out below.

Reference	Description
[RFC2616]	IETF, RFC 2616: HyperText Transfer Protocol - HTTP/1.1, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999, http://ietf.org/rfc/rfc2616.txt
[RFC2246]	The TLS Protocol Version 1.0, T. Dierks, C. Allen, January 1999, http://www.ietf.org/rfc/rfc2246.txt
[SOAP2003a]	W3C, SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) W3C Recommendation, 27 April 2007, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/
[SOAP2003b]	W3C, SOAP Version 1.2 Part 2: Adjuncts (Second Edition), W3C Recommendation, 27 April 2007, http://www.w3.org/TR/2007/REC-soap12-part2-20070427/
[WSA2006]	W3C, Web Services Addressing 1.0 - Core, W3C Recommendation, 9 May 2006, http://www.w3.org/TR/2006/REC-ws-addr-core-20060509.
[WSAM2007]	W3C, Web Services Addressing 1.0 - Metadata, W3C Recommendation, 4 September 2007, http://www.w3.org/TR/2007/REC-ws-addr-metadata- 20070904
[WSDL2001]	W3C, Web Services Description Language (WSDL) 1.1, W3C Note, 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315
[WSPA2007]	W3C, Web Services Policy 1.5 - Framework, W3C Recommendation, 4 September 2007, http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904
[WSPL2007]	W3C, Web Services Policy 1.5 - Framework, W3C Recommendation, 4 September 2007, http://www.w3.org/TR/2007/REC-ws-policy-20070904.
[WSS2006]	OASIS, Web Services Security: SOAP Message Security 1.1, OASIS Standard, 1 February 2006, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
[WSSPL2007]	OASIS, WS-SecurityPolicy 1.2, OASIS Standard, 1 July 2007, http://docs.oasis-open.org/ws-sx/wssecuritypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf
[XCTP2006]	OASIS, Web Services Security X.509 Certificate Token Profile 1.1, OASIS Standard Specification, 1 February 2006, http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf
[XML2006]	W3C, Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-20060816
[XNS2006]	W3C, Namespaces in XML 1.0 (Second Edition), W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816
[XSD2004a]	W3C, XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-1/
[XSD2004b]	W3C, XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/
[XSSP2008]	XML Signature Syntax and Processing (Second Edition), W3C Recommendation 10 June 2008, http://www.w3.org/TR/xmldsig-core/
[ATS5820]	ATS 5820-2010 E-Health Web Services Profile, 5 March 2010, Committee IT-014, Health Informatics, http://ihe-australia.wikispaces.com/file/view/ATS+5820-2010+E-health+web+services+profiles.pdf
[ATS5821]	ATS 5821-2010 E-Health XML secured payload profiles, Committee IT-014, Health Informatics, http://ihe-australia.wikispaces.com/file/view/ATS+5821-2010+E-health+XML+secured+payload+profiles.pdf
[WSIBSP2010]	Basic Security Profile Version 1.1, WSI Organisation, 24 October 2010,