Tasmanian Government Web Services Technical Standard

Version 1.1 – 7 January 2010

Tasmania
Explore the possibilities

Table of Contents

1 Introduction

The Tasmanian Government Web Services Technical Standard has been produced as a part of the Interoperability Program of the Office of eGovernment within the Department of Premier and Cabinet.

The aim of the Interoperability Program is to improve whole-of-government efficiency, effectiveness, and agility. The use of common standards and guidelines is one way to build the capacity for interoperability between Agencies. Interoperability standards and guidelines are developed in consultation with stakeholders, generally from Tasmanian Government Agencies.

1.1 Purpose of the Standard

There is an increasing need in government for the exchange of information between systems. This provides a challenge because many new and legacy systems have proprietary interfaces that provide limited scope for interoperability. Web Services provide one solution to this problem by providing open standards for information exchange that is supported by a number of application vendors. However, the use of Web Services alone does not guarantee interoperability. There are a number of design and technology choices that need to be made to ensure that systems are interoperable. This document is intended to provide standards information to enable and enhance interoperability when developing Web Services in the Tasmanian Government.

1.2 What are Web Services?

Web Services can be viewed from both a business and technical perspective and two definitions have been provided below to explain web services in each context for the different audiences.

1.2.1 Example definition from a business perspective

Web services are automated information services that are conducted over the Internet, using standardised technologies and formats/protocols that simplify the exchange and integration of large amounts of data over the Internet. They make it easier to conduct work across organisations regardless of the types of operating systems, hardware/software, programming languages, and databases that are being used[1] e.g. they facilitate automated transactions between incompatible systems.

[1] US Environmental Protection Agency, Environmental Information Exchange Network and Grant Program Glossary, http://www.epa.gov/exchangenetwork/glossary.html#W, view date Aug 2008.

1.2.2 Example definition from a technical perspective

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards.[2]

1.3 Development of Standard

This technical standard was developed using a consultative process by forming a cross agency Web Services Working Group. The contribution from the members of the Web Services Working Group or through other approaches is gratefully acknowledged, which was represented by the following agencies:

- Department of Education (DoE)
- Department of Health and Human Services (DHHS)
- Department of Infrastructure, Energy and Resources (DIER)
- Department of Justice (DoJ)
- Department of Police and Emergency Management (DPEM)
- Department of Premier and Cabinet (DPAC)
- Department of Primary Industry and Water (DPIW)
- Department of Treasury and Finance (DTF)

Dr David Legge (DIER) is acknowledged as the principle author of the standard. Recognition is also given to Michael Ross of *XVT Solutions* for drafting the original document that this standard is based upon.

Maintenance and review of this document is co-ordinated by Office of eGovernment, Department of Premier and Cabinet.

1.4 Authority and History

The Tasmanian Government Web Services Technical Standard has been approved for use in Tasmanian Government Agencies by the Tasmanian Government's Inter Agency Steering Committee (IASC) on 8 September 2008.

Version	Date approved	Reason	Amended Sections
V1.0	8 September 2008	Initial Release	
V1.1	7 January 2010	Change of unit name from Inter Agency Policy and Projects Unit (IAPPU) to Office of eGovernment	

[2] W3C, Web Services Architecture W3C Working Group Note, February 2004, http://www.w3.org/TR/ws-arch/, view date August 2008.

1.5 Intended Audience

1.5.1 Technical focus for Web Service Standard

Collaboration is understood from the perspective of collaboration within agencies and collaboration between agencies by ensuring the approach to Interoperability takes place within four specified 'architectural domains':

- Business Domain
- Information Domain
- Applications domain
- Technology domain

The Tasmanian Government Web Services Technical Standard focuses on the Technology domain. There may be other guidelines developed in the future on web services, which complement this technical standard, that focus on the business, information or application domains.

1.5.2 Structure of the Standard

The document is structured such that Australian, Tasmanian and Agency standards issues are presented first, and these would be of relevance to Managers and Application Business Owners.

Subsequent chapters detail technical Web Services standards in-depth, and these would be of relevance to Developers and System Administrators.

1.5.3 Points of Interest for Managers

Most areas of this document are relevant to developers. The following chapters or sections will be most relevant:

- Web Services Best Practice
- Choosing Web Services
    - Choosing Web Services or Other Technologies
    - Choosing Web Services Standards
    - Choosing Versions of Web Services Standards
    - Problem Domains Supported by Web Services

1.5.4 Points of Interest for Developers

Most areas of this document are relevant to developers. The following chapters or sections will be most relevant:

- WSDL Design Recommendations
- Choosing Web Services
    - Choosing Web Services or Other Technologies

- Choosing Web Services Standards
- Choosing Versions of Web Services Standards
- Problem Domains Supported by Web Services

- Web Services Technical Standards Information

1.5.5 Points of Interest for Administrators

Some chapters or sections of this document will be more applicable to application administrators.

- Coordination and Discovery
- Service Broker Model
- Enterprise Service Bus

1.6 Feedback and assistance

The Office of eGovernment welcomes feedback on this standard. Please direct your feedback and enquiries to:

Office of eGovernment
Department of Premier and Cabinet
Ph: (03) 6232 7722 or email: egovernment@dpac.tas.gov.au

This document can be found at www.egovernment.tas.gov.au.

2 Web Services Best Practice

There is an increasing need in government for the exchange of information between systems. There are a number of design and technology choices that can be standardised to ensure systems are interoperable.

To assist with maintaining compatibility, when developing Web Services across government we should apply a set of Web Services good practices, which support:

- **Interoperability** – the use common standards and guidelines to build the capacity for interoperability between agencies.
- **Security** – to enable the exchange of information to be adequately protected.
- **Flexibility** –the use of the Tasmanian Government Web Services Technical Standard can increase the flexibility to share information more easily across government.

To achieve robust Web Services good practices, we should:

1. Use recognised standards to develop Web Services as specified in this document.

2. Consider Tasmanian Government agency policies and guidelines that relate to Web Services including:

    - Information Security Plan
    - Information Communications and Technology (ICT) standards and policies
    - Business requirements

Appendix A shows an example of specific standards that are related to Web Services in the Department of Infrastructure, Energy and Resources (DIER).

3. Comply with relevant Tasmanian Government whole of government policies and guidelines including:

    - *Tasmanian Government Information Security Framework*, which can be obtained from www.egovernment.tas.gov.au
    - *Tasmanian Government Interoperability Standards and Guidelines*, which can be obtained from www.egovernment.tas.gov.au.

2.1 Web Services Interoperability Organisation – Basic Profile

2.1.1 Summary

The Web Services Interoperability Organisation (WS-I) is an online organisation that is focused on delivering standardised best practice methodologies for Web Service development. The Basic Profile Working Group has release version 1.1 of the *Basic Profile* document.

The *WS-I Basic Profile 1.1* can be found at: http://www.ws-i.org/Profiles/BasicProfile-1.1.html

The *WS-I Basic Profile* recommends the following standards be adhered to:

- WSDL 1.1
- SOAP 1.1
- UDDI 2.0
- XML 1.0
- XML Schema 1.0

While this document recommends adherence to the *WS-I Basic Profile 1.1* recommendations, most Web Services do not require service discovery. Therefore UDDI 2.0 is not a mandatory Web Services protocol, and in the future the competing standard ebXML Registry may be chosen if it better fits future business and technical requirements. See chapter Web Services Technical Standards Information for more discussion of this.

An additional consideration is that more advanced versions of WSDL, SOAP and UDDI have been ratified by standards bodies. While the *WS-I Basic Profile 1.1* advocates the use of the earlier versions of these Web Services specifications, it is possible to choose to use the latter versions. See the section Choosing Web Services on more information on this.

2.1.2 Validation

One of the deliverables of the Basic Profile Working Group is a functional analysis tool that can be used to verify the standards-compliance of Web Services.

It is recommended that all WSDL files pass the validation process.

Both Java and .NET versions of the tool are available:

.NET :http://www.ws-i.org/Testing/Tools/2005/06/WSI_Test_CS_Final_1.1.zip

Java :http://www.ws-i.org/Testing/Tools/2005/06/WSI_Test_Java_Final_1.1.zip

2.2 WSDL Design Recommendations

2.2.1 XSD Import

Importing references to XSD documents rather than embedding the definition in the WSDL is recommended for a couple of reasons. By referencing XSD documents, we can firstly simplify our WSDL, and hence improve the readability by humans. Secondly, it helps improve reuse and maintainability, as the data schemas are stored in individual files that can be either modified as required, or referenced independently by other WSDL documents.

WS-I has some rules that should be adhered to in using the import functionality for XSD files.

R2002 To import XML Schema Definitions, a DESCRIPTION MUST use the XML Schema "import" statement.

R2003 A DESCRIPTION MUST use the XML Schema "import" statement only within the xsd:schema element of the types section.

R2004 In a DESCRIPTION the schemaLocation attribute of an xsd:import element MUST NOT resolve to any document whose root element is not "schema" from the namespace "http://www.w3.org/2001/XMLSchema".

WS-I

A corollary of the first rule (R2002) is that the schema “includes” statement should not be used.

The import statement can also be used to import elements.

The last rule (R2004) is difficult to understand, so here is a translated version:

In a DESCRIPTION the schemaLocation attributes of an xsd:import element MUST resolve to a document whose root element is “schema” from the namespace “http://www.w3.org/2001/XMLSchema”.

```
<?xml version="1.0"?>
<definitions name="StockQuote"
        targetNamespace="http://example.com/stockquote/definitions"
    xmlns:tns="http://example.com/stockquote/definitions"
    xmlns:xsd1="http://example.com/stockquote/schemas"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/schemas"
           location="http://example.com/stockquote/stockquote.xsd"/>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
</definitions>
```

2.2.2 Documentation Elements

WSDL definitions can be easily read by computers, but ultimately humans are still required to instruct the application in how to use the service for the tasks that is required. Understanding WSDL from a developer’s point of view is not always straightforward, and when a service is consumed remotely, and no documentation is available, then understanding how to use the service is not easy.

Document tags are a way to embed comments and descriptions in the middle of the WSDL. You can choose to store extra metadata in these elements, or just choose to store some simple text.

R2030 In a DESCRIPTION the wsdl:documentation element MAY be present as the first child element of wsdl:import, wsdl:part and wsdl:definitions in addition to the elements cited in the WSDL1.1 specification.

WS-I Basic Profile

An example of using the documentation tags.

```
<?xml version="1.0"?>
<definitions name="StockQuote"
        targetNamespace="http://example.com/stockquote/service"
        xmlns:tns="http://example.com/stockquote/service"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:defs="http://example.com/stockquote/definitions"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
        <import namespace="http://example.com/stockquote/definitions"
            location="http://example.com/stockquote/stockquote.wsdl"/>
        <binding name="StockQuoteSoapBinding"
                type="defs:StockQuotePortType">
                . . .
        </binding>
        <service name="StockQuoteService">
                <documentation>My first service</documentation>
                <port name="StockQuotePort"
                        binding="tns:StockQuoteBinding">
                        <soap:address
                                location="http://example.com/stockquote"/>
                </port>
        </service>
</definitions>
```

2.2.3 Schema Versioning

Even the most well thought out schema design may need to change as business and technical constraints.

This introduces the problem of how to control versioning of the schema.

Best practices for XML schema versioning and modification[3]:

- Capture the schema version using the version attribute of the schema tag. If an instance document is a compound document - that is, an assembly of XML fragments - then place a version attribute on the root of each fragment.
- Identify in the instance document, what version/versions of the schema with which the instance is compatible.
- Make previous versions of an XML schema available.
- When an XML schema is only extended, (e.g., new elements, attributes, extensions to an enumerated list, etc.) one should strive to not invalidate existing instance documents.
- To prevent breaking old instance documents give the new Schema version a different filename or a different URL location or both.
- To facilitate an application in recognising that the elements content has changed, do not use anonymous types. Instead, use named types that contain version information.
- New Elements or Attributes can be added, but compulsory Elements or Attributes should not be removed.
- Optional Elements and Attributes can be removed or renamed, but existing clients that utilise those Element or Attributes will no longer be able to forward that information.

[3] http://www.xfront.com/Versioning.pdf and http://www.xfront.com/SchemaVersioning.html

2.2.4 Schema Release Management

When new versions of applications are developed that involve schema changes, the changes, their rationale and the new version number of the scheme should be documented in the release notes. The release notes should include instructions for deploying the new schema to the application.

A copy of the new schema should be placed in a repository along with older versions of the schema for reference purposes. Ideally this schema and associated documentation should be available at the namespace URL.

2.2.5 Modification of WSDL Definitions

Rules for modifying WSDL definitions are as follows:

- Add new methods to the interface.
- Modify the schema of the objects passed in and returned from the method. The changes to the objects will need to pass the rules for changing the XSD Schema Definitions.
- Do not add or remove parameters from any methods.
- Do not change the name of any methods.

3 Coordination and Discovery

3.1 Service Broker Model

3.1.1 Introduction

The Service Broker Model refers to the publishing of services in a central directory, and the ability for service consumers to request the location of the service so they can consume it. The two main industry-wide standards for this are Universal Description Discovery and Integrations (UDDI) and Electronic Business and XML Registry (ebXML Registry). Both are ratified by OASIS, but although ebXML Registry provides more functionality than UDDI, UDDI has more support with vendors. Should the need arise for a registry, the Agency will decide which of UDDI or ebXML Registry will be used.

Configuration for UDDI/ebXML Registry is an infrastructure management task. All Web Services should be capable of being registered in the UDDI, and all client applications should be capable of requesting the address of a service from the UDDI.

3.1.2 References

OASIS UDDI Technical Committee: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec

OASIS ebXML Registry Technical Committee: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=regrep

3.1.3 Environment Notes

UDDI is an add-on component that can be installed with Oracle Application Server.

UDDI is an add-on component for Windows 2003 Server.

ebXML Registry support is available is an add-on component to Oracle Application Server called Oracle Application Server 10g Integration B2B: ebXML Adapter.

3.2 Enterprise Service Bus

3.2.1 Introduction

An Enterprise Service Bus (ESB) is not a mandatory component for building interoperable services. An ESB may not be necessary unless there is strong business or technical requirements to do so. It is mentioned in this document to raise awareness of its existence and what functionality such a middleware solution can provide. Furthermore, ESBs are used around the Tasmanian Government (e.g. DPIW) and so its role in Web Services should at least be outlined.

The goal of an ESB is to provide a middleware integration layer through which all core business communications are processed. Processing information through a central system offers the following services:

- Event Triggers
- Routing
- XML Translation

In a simple Web Services implementation, when one application needs to talk to another, it would either know the address of the service, or it would request the address from the UDDI directory, then it would execute its request.

With an ESB in place, when an application needs to talk to another, it would talk to the ESB, and the ESB would follow its configured rules to forward the message in the appropriate format and to the appropriate destination while following the security policies of the organisation.

Processes that need to be followed as business requirements change do not need to be complex and expensive procedures.

Example

When a person attends a Service Tasmania shop to change their address details, a message is generated containing the details identifying the person, and new address information to be updated in the appropriate systems. This message is forwarded to the ESB.

The ESB looks at its rules, and it notices that the message is a "Change of Address" message, so it forwards this message to a remote Change of Address Web Service in another Agency.

Suppose in the future, the Tasmanian Government Client Update Service (CUS) comes online. The CUS is a proposal for the notification of change of contact information to Agencies via a brokered service.

Given that an ESB is in place, to meet the new requirements, a new set of rules is added to the ESB. The rules might do the following:

- Notify the Change of Address Web Service of the change, if this fails, respond to client with failure message, and do not do any more processing.
- If the update succeeds:
- Transform the message to the CUS format using an XSLT Transform.
- Apply the appropriate security functions to the message.
- Forward the message to the CUS service. If this fails, the ESB will automatically notify the appropriate parties that this inter-agency update failed.

The requirements for the "Change of Address" message changed, but no changes were required to the Client or the Service application. The only changes were to configured rules inside the ESB.

4 Web Services Technical Standards Information

4.1 Introduction

This chapter outlines mandatory and optional Web Service technical standards. As the terms suggest, mandatory standards are those that should be used to develop Web Services for the Agency.

Optional standards are also given to provide greater flexibility, but their use should be discussed with appropriate personnel in the Agency as early in the development process as possible. The reason for this is that the optional Web Service standards are not necessarily yet supported on Agency hosting infrastructure, but can be if there is sufficient notice given of their intended use. Furthermore, the partner Agency should be comfortable with the Web Service standards chosen.

If there is a conflict between an advocated Web Service standard and either the mandatory or optional standards in this document, their use should be discussed with appropriate personnel in the Agency as early in the development process as possible. Since Web Services specifications are rapidly evolving, it is recognised that the standards recommendations in this document will also evolve. This means that there may be room to accommodate Web Services that fall outside the recommendations in this document. However, valid business or technical reasons for requiring standards other than those advocated here needs to be provided, and agreement from the Agency needs to be reached to use it.

4.2 Service Orientated Architecture Reference Model

It is recommended that the concepts and terminology employed in discussing Web Services and SOA be compatible with the *OASIS Reference Model for Service Orientated Architecture*. The document can be obtained from the OASIS SOA Reference Model Technical Committee at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

4.3 Choosing Web Services

There is a large choice of Web Service standards available, but not all are suitable to use for a variety of reasons. While this document provides a set of preferred Web Services standards used within the Tasmanian Government, it is advisable to ensure that any choice of standard meets business needs and is supportable within your application environment. The following sub-sections outline some of factors that can influence the decision to use Web Services, the choice of Web Service standards, and also the particular version of a standard that might be employed.

4.3.1 Choosing Web Services or Other Technologies

A primary decision is to whether to use Web Services or some other interoperability technology. There are a number of factors that influence this decision:

- Most fundamentally, chosen interoperability technologies should support the underlying business requirements. The business requirements that should drive the selection of technologies, and not the other way around.

- Web Services are platform and vendor neutral, and so are advantageous for interoperability in a multi-vendor or multi-platform environment. These advantages are diminished when there is less diversity in application vendors or platforms. Since there is a variety of platforms supported and vendors products utilised across the Tasmanian Government, the neutrality of Web Services in these areas are a major driver for their adoption.
- The partners in the data exchange must be able to support the chosen technologies. Support includes the design, implementation and maintenance of the technology supporting the standard, and any additional limitations imposed by the choice of technology.
- Web Services are an open, non-proprietary interoperability mechanism, which are supported by a wide range of application vendors. Other interface types can sometimes rely on closed proprietary technologies that are tied to a particular vendor or platform.
- Many Web Service specifications are ratified standards, but there can be vendor-specific implementation details that can complicate the adoption of Web Services. This is somewhat mitigated by reference documents such as the *WS-I Basic Profile*.
- Web Services sometimes are not as highly performant as some other data exchange technologies due to the overheads of processing XML. When transferring large volumes of data, particularly with high bandwidth/low latency requirements, the adoption of Web Services should be reviewed.
- Web Services can sometimes be complex, and sometimes other interface technologies can be simpler for specific purposes.
- Sometimes a vendor will have a preferred interface technology that is different to Web Services. Utilising such a interface technology over Web Service becomes a tradeoff between the vendor's requirements over other issues discussed in this sub-section.
- Web Services are often simpler to route through firewalls and reverse-proxy than other interface technologies due to the primary transport being HTTP.
- Web Services and the tools that support them are quite new and rapidly changing, and some aspects of these do not have the maturity of other interface technologies. Generally, it is advisable to adopt Web Services that satisfy requirements outlined later in section 4.3.4 . For more exotic requirements, Web Services should be used with care.
- Web Services can be designed to be loosely-coupled, which allows relative independence between systems. Some other interface technologies can impose restrictive dependencies between systems that make them fragile.

4.3.2 Choosing Web Services Standards

Once the decision to adopt Web Services has been made, the second choice is what particular Web Service standards to use. This is dependent on a number of factors:

- As discussed above, the business requirements influence the choice of technology, but also the particular choice of Web Service standard. While some technologies merely provide low-level functionality (e.g. data encoding), others provide higher-level functionality that can enable certain business requirements (e.g. data transformation, encryption, access control, non-repudiation or guaranteed delivery).

- Whether or not the standard has been ratified by a recognised standards body. A ratified standard may have greater longevity and vendor support than a published specification without ratification. The most recognised standards bodies for Web Services are OASIS and W3C. This being said, some of the most commonly used specifications have never been ratified (e.g. SOAP 1.1), although published specifications in a stable form still exists.
- The level of support of the standard by vendors, including in operating systems, application platforms, individual applications, developer toolsets, libraries and documentation. This is generally straightforward to determine for Web Services in use between similar operating systems or application platforms, but can be a challenge where two partners have different operating systems (e.g. Windows vs Unix), or are using different application platforms (e.g. .NET vs Java). The standards advocated in this document are deliberately chosen to generally have cross-platform support, but this issue should be should be reconsidered on a case-by-base during the design phase of a project to implement Web Services.
- The maturity of support of the standard by vendors. Some early implementations of new Web Services standards by vendors can contain bugs or may not be feature complete, and these can degrade interoperability. Therefore try to choose a Web Services standard that has well-regarded and complete implementations with a good track record.
- The dependencies of the standards chosen on other related standards. For example, a number of Web Service standards use WS-Addressing.
- Security is a major issue in the selection of Web Service standards. Security is an end-to-end business problem and should be considered in the entire context of the business problem. Web Services can provide a level of security in the problem domains of access control, encryption, non-repudiation but are limited to areas like system-level, application-level and user-level security. The choice of appropriate security standards should be based on an assessment of the risks.

4.3.3 Choosing Versions of Web Services Standards

Once a Web Service standard is selected, the third issue is choosing the particular version of standard to be utilised. The choice of Web Services standard is dependent on a few major factors:

- Where possible it is generally recommended to go with the latest version of a Web Service standard that has been ratified by a standards committee. This is likely to have a longer lifetime than older versions of a standard.
- Also, it is recommended to go with a version that is supported by the application vendor and is preferably a mature and relatively bug-free implementation.
- It is also necessary to understand the dependencies between Web Service standards. For example, a number of Web Service standards are tied to a particular version of SOAP and WSDL. Care needs to be taken to ensure all the particular versions will interoperate together.

4.3.4 Problem Domains Supported by Web Services

The following sections outline the main Web Service specifications that are likely to be supported within the Tasmanian Government. These cover a variety of problem domains that the Web Service standards attempt to support. To better understand this, the table below summarises what the standards cover and provides a basic indication of when to use a particular Web Service standard.

Domain	Issue	Web Service standard	What the Web Service standard is used for	When to use this standard
Administration	Management of Web Services	WSDM	Provides a framework for managing Web Services.	Optional. Recommended when managing a multiplicity of Web Services.
Data	Representation	XML	Used in all Web Services for representing or encoding data.	Compulsory. Used in most other Web Service standards.
Data	Structure and validation	XML Schema	Used to define the structure of XML data and to provide validation.	Compulsory. Used in most other Web Service standards.
Data	Context	Namespaces in XML	Used to distinguish between different items in XML data.	Compulsory. Used in most other Web Service standards.
Data	Data transformation	XML Transformations	Used to transform XML from one representation to another.	Optional. Required when changing XML data from one format to another.
Data	Discovery	UDDU or ebXML Registry	Used to remotely discover and search for Web Services across the network.	Optional. Recommended only when remote Web Service endpoints are unknown or change regularly. In most scenarios, this is not a problem.
Message Delivery	Interface definition	WSDL	Defines and describes the public interface for a Web Service.	Compulsory. Always used, since it provides the fundamental service contract for message interchange.
Message Delivery	Message encoding	SOAP	Used to define an envelope format for the exchange of Web Service XML messages.	Compulsory. Always used, since it provides the envelope format for data exchange.
Message Delivery	Message routing and addressing	WS-Addressing	Used to define location endpoints for the routing and delivery of Web Service data.	Optional. Generally required only when the use of other Web Services specifications call for it.
Message Delivery	Guaranteed delivery	WS-ReliableMessaging	Used to ensure whether or not a Web Service message has been delivered	Optional. Generally required only when it is critical to business function that a Web Service message has been delivered.
Security	Access control	WS-Security or SSL	Confers the ability to only provide access to a resource when an access token is provided. WS-Security is typically employed for application-level or system-level access control. SSL can be used for these and also user-level access control.	Optional. Use of security mechanisms is based on risk assessment. This level of security is generally required when the Web Service must be protected from unauthorised use. For access control alone an alternative to WS-Security or SSL is using Basic HTTP Authentication.

Domain	Issue	Web Service standard	What the Web Service standard is used for	When to use this standard
Security	Encryption	WS-Security or SSL	Provides the ability to encrypt data either within the SOAP envelope (WS-Security) or in the underlying network transport (SSL). WS-Security is typically employed for application-level or system-level encryption. SSL can be used for these and also user-level encryption.	Optional. Use of security mechanisms is based on risk assessment. This level of security is generally required when the Web Service is passing sensitive data, particularly on an untrusted network or host.
Security	Non-repudiation	WS-Security or SSL	Provides the ability to prove (to some degree) the identity of a partner in a transaction[4]. WS-Security is typically employed for application-level or system-level non-repudiation. SSL can be used for these and also user-level non-repudiation.	Optional. Use of security mechanisms is based on risk assessment. This level of security is generally required when the identity of the sender and recipient must be identified during the transaction.

The table highlights a few issues with the current mix of Web Service standards:

- Some specifications are always used within Web Services (e.g. XML), while others are only used according to other, usually business requirements (e.g. various security specifications)
- There are problem domains that are not covered by the recommended standards in this document that are listed in the table above. Some examples of these problem domains are: Transactioning, stateful services and advanced session management, federated identity, complex or long running business processes (including choreography and orchestration), message metadata and data policy enforcement. The reason for this is that specifications covering these problem domains do not yet have either the standards ratification, maturity and vendor adoption to warrant inclusion. Later revisions of this document may include additional specifications that provide these capabilities.

4.4 Mandatory Web Service Standards

The following Standards are listed as Mandatory as they are the standards that Web Services are built on.

4.4.1 XML

4.4.1.1 Description

The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages, capable of describing many different kinds of data. It is a simplified subset of SGML. Its primary purpose is to facilitate the sharing of data across different systems, particularly systems

[4] There is some debate about the level of non-repudiation provided by SSL, particularly in validating the identity of the hosting organisation of a commercially-provided SSL certificate. There are variances in quality amongst commercial providers about the proof-of-identity procedures.

connected via the Internet. Languages based on XML (for example, RDF, RSS, MathML, XHTML, SVG, and cXML) are defined in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.

Wikipedia.org

4.4.1.2 Version History

XML 1.0 Fourth Edition – W3C Recommendation, 16 August 2006

4.4.1.3 References

http://www.w3.org/TR/xml

http://www.xml.com/axml/testaxml.htm

4.4.2 XML Schema

4.4.2.1 Description

XML Schema is the first of several schema languages to be published as a recommended standard by the W3C. It is a language built on XML to describe the structure and content of XML documents.

4.4.2.2 Version History

XML Schema 1.0 – W3C Recommendation, 28 October 2004

4.4.2.3 Namespaces

Specification	Commonly Used Prefix	Namespace URI
XML Schema	**Xsd**	**http://www.w3.org/2001/XMLSchema**
XML Schema Instance	**Xsi**	**http://www.w3.org/2001/XMLSchema-instance**

4.4.2.4 References

http://www.w3.org/TR/xmlschema-0/

4.4.3 XML Transformations

4.4.3.1 Description

XML Transformations (XSLT) is a language to allow the transformation of XML documents into other XML documents.

4.4.3.2 Version History

XML Transformations 1.0 - W3C Recommendation, 16 November 1999

4.4.3.3 Namespaces

Specification	Commonly Used Prefix	Namespace URI
XML Transformations	Xsl	http://www.w3.org/1999/XSL/Transform

4.4.3.4 References

http://www.w3.org/TR/xslt

4.4.4 Namespaces in XML

4.4.4.1 Description

> 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.
>
> *W3C*

The W3C have produced a set of best practices for the usage of Namespaces in XML. For each of the standards discussed in this chapter, the namespace name (URI) that should be used is given.

The following aspects should be considered when formulating XML document namespace definitions in Agency XML documents:

- The use of *meaningful* namespace names to be easily comprehendible;
- Who might *reuse* the XML document (application, Agency or whole-of-government) and so define namespace names to be re-usable in that context;
- The ability to *publish* an XML schema and other resources at the URI given in the namespace of the XML schema;
- The *discovery* requirements of the XML schema, particularly the ability of XML schemas to be aggregated in shared schema directories;
- The *versioning* requirements of XML schemas - often denoted within in namespace URIs.

4.4.4.2 Version History

Namespaces in XML 1.0 Second edition - W3C Recommendation, 16 August 2006

4.4.4.3 References

http://www.w3.org/TR/REC-xml-names/

4.4.5 WSDL

4.4.5.1 Description

> 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, HTTP GET/POST, and MIME.

W3C

4.4.5.2 Version History

Web Services Description Language (WSDL) 1.1 - W3C Note, 15 March 2001

Web Services Description Language (WSDL) 2.0 - W3C Recommendation, 26 June 2007

4.4.5.3 Issues

The *WS-I Basic Profile* sometimes lags the development and ratification of the standards it discusses. This means the profile sometimes recommends versions of standards that have been replaced by newer versions. Care should be taken to choose a version of WSDL that is compatible with partners, tools, applications and other Web Services standards in use.

While transports other than HTTP are potentially available with WSDL, for interoperability always use HTTP or HTTP over SSL/TLS as the underlying transport.

4.4.5.4 Namespaces

Specification	Commonly Used Prefix	Namespace URI
WSDL 1.1	wsdl	http://schemas.xmlsoap.org/wsdl/
WSDL 2.0	wsdl	http://www.w3.org/ns/wsdl

4.4.5.5 References

WSDL 1.1 - http://www.w3.org/TR/wsdl

WSDL 2.0 - http://www.w3.org/TR/wsdl20

4.4.6 SOAP

4.4.6.1 Description

SOAP is a simple protocol designed for interchanging complex structured information in an XML based format that can easily be passed over the internet using existing transport protocols such as HTTP.

4.4.6.2 Version History

SOAP 1.1 - W3C Note, 8 May 2000

SOAP 1.2 - W3C Recommendation, 27 April 2007

4.4.6.3 Issues

The *WS-I Basic Profile* sometimes lags the development and ratification of the standards it discusses. This means the profile sometimes recommends versions of standards that have been replaced by newer versions. Care should be taken to choose a version of SOAP that is compatible with partners, tools, applications and other Web Services standards in use.

4.4.6.4 Namespaces

Specification	Commonly Used Prefix	Namespace URI
SOAP 1.1 envelope	SOAP-ENV	http://schemas.xmlsoap.org/soap/envelope/
SOAP 1.1 encoding	SOAP-ENC	http://schemas.xmlsoap.org/soap/encoding/
SOAP 1.2 envelope	env	http://www.w3.org/2003/05/soap-envelope
SOAP 1.2 encoding	enc	http://www.w3.org/2003/05/soap-encoding

4.4.6.5 References

SOAP 1.1 - http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

SOAP 1.2 - http://www.w3.org/TR/soap12-part0/

4.5 Optional WS Standards

This section of the document describes some of the more common and important standards that are available for Web Service development, that may be optionally used in Agencies. If their use is required, then this should be discussed with appropriate personnel in the Agency, and with any external partners in Web Service interfaces.

4.5.1 UDDI

4.5.1.1 Description

> UDDI is an acronym for Universal Description, Discovery, and Integration – A platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative (sponsored by OASIS) enabling businesses to discover each other and define how they interact over the Internet.
>
> *Wikipedia.org*

4.5.1.2 Version History

UDDI 2.0 - OASIS, published 19 July 2001

UDDI 3.0 - OASIS, ratified February 2005

4.5.1.3 Issues

The *WS-I Basic Profile* sometimes lags the development and ratification of the standards it discusses. This means the profile sometimes recommends versions of standards that have been replaced by newer versions. Care should be taken to choose a version of UDDI that is compatible with partners, tools, applications and other Web Services standards in use.

UDDI and ebXML Registry are two competing standards for discovery of Web Services. If the need for service discovery should arise then one or both of UDDI and ebXML Registry will be chosen by the Agency to provide this.

4.5.1.4 Namespaces

Specification	Commonly Used Prefix	Namespace URI
UDDI 2.0	uddi	urn:uddi-org:api_v2
UDDI 3.0	uddi	urn:uddi-org:api_v3

4.5.1.5 References

http://www.oasis-open.org/specs/index.php

http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

4.5.2 ebXML Registry

4.5.2.1 Description

ebXML Registry is an acronym for Electronic Business using eXtensible Markup Language Registry – A platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. ebXML Registry is part of a technical framework that will enable XML to be utilised in a consistent manner for the exchange of all electronic business data.

xml.coverpages.org

4.5.2.2 Version History

ebXML Registry 3.0 - OASIS, ratified May 2005

4.5.2.3 Namespaces

Specification	Commonly Used Prefix	Namespace URI
ebXML Registry 3.0 Content Management	cms	urn:oasis:names:tc:ebxml-regrep:xsd:cms:3.0
ebXML Registry 3.0 Lifecycle Management	lcm	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
ebXML Registry 3.0 Query protocol	query	urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0
ebXML Registry 3.0 Registry Information Model	rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0
ebXML Registry 3.0 Registry Service Requests	rs	urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0

4.5.2.4 Issues

UDDI and ebXML Registry are two competing standards for discovery of Web Services. If the need for service discovery should arise then one or both of UDDI and ebXML Registry will be chosen by the Agency to provide this.

4.5.2.5 References

http://www.oasis-open.org/specs/index.php

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=regrep

4.5.3 WS-Security

4.5.3.1 Description

WS-Security is a standard devised for securing Web Service calls. Web Service calls are stateless, so the identity of the person or the system that is accessing the Web Service needs to be attached to the Web Service message so that the entire operation is complete in one call/response.

There are three main aims of the WS-Security standard.

1. Protect the **Integrity** of the message. This means that the Web Service can check to see if the message has been tampered with reroute, and can disregard its contents.

2. Protect the **Confidentiality** of the message by encrypting the parts of the message that should not be readable by anyone other than the destination Web Service.

3. Forwarding the **Identity** of the user or the system that is using the Web Service. This is critical for the Web Service to be able to filter returned content based on the access privileges of a system or Web Service.

4.5.3.2 Version History

WS-Security 1.0 - OASIS, ratified April 2004

WS-Security 1.1 - OASIS, ratified February 2006

4.5.3.3 Why would you use this?

This standard is highly applicable to any Web Service that will be publishing any information that is of a sensitive nature. This standard is also highly applicable to any Web Service that accepts information from other sources.

- Sensitive information can be encrypted so that prying eyes cannot read it.
- Sensitive information can be given only to parties that are trusted.
- Only verified information will be accepted.

This standard has been widely accepted in both the Java and the .NET worlds, with product support available to use it.

4.5.3.4 Issues

WS-Security can be complex and may not be fully supported on all application platforms. For simple point-to-point Web Services, using SSL/TLS over HTTP may be a simpler choice.

WS-Security can use public key infrastructure (PKI) for the encryption of data within the SOAP messages, so similar issues of can arise to what is discussed in the following section on SSL/TLS.

WS-Security may be suitable for system-level and application-level security, but is generally not employed for user-level security.

4.5.3.5 Namespaces

Specification	Commonly Used Prefix	Namespace URI
XML Digital Signature	ds	http://www.w3.org/2000/09/xmldsig#
Web Services Security 1.0 Security Extensions	wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
Web Services Security 1.1 Security Extensions	wsse11	http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd
Web Services Security 1.0 Utility	wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
XML Encryption	xenc	http://www.w3.org/2001/04/xmlenc#

4.5.3.6 References

http://www.oasis-open.org/specs/index.php

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

4.5.4 SSL/TLS

4.5.4.1 Description

Secure Sockets Layer (SSL), and its successor, Transport Layer Security (TLS) use public key infrastructure (PKI) to provide encryption to a variety of protocols, including HTTP. Since most Web Services SOAP messages are transported via HTTP, SSL provides a mechanism to encrypt Web Services communications. A further benefit of using PKI, is that access control and non-repudiation can also be provided when using SSL or TLS.

SSL was originally developed by Netscape in the 1990s, with SSL 3.0 being released in 1996. Later the Internet Engineering Task Force (IETF) adopted (with minor modifications) the SSL 3.0 specification, naming it TLS version 1.0. Contemporary web browsers and HTTP servers can support most SSL and TLS specifications, but it is generally recommended to only use SSL 3.0 or TLS 1.0 due to security flaws in earlier versions of SSL.

4.5.4.2 Version History

SSL 1.0 – Netscape, never publicly released

SSL 2.0 – Netscape, published 1994

SSL 3.0 – Netscape, published 1996

TLS 1.0 - ITEF, Request for Comment (RFC) published January 1999

4.5.4.3 Issues

While not as complex as WS-Security, the use of SSL/TLS can prove to be challenging because of the issues surrounding PKI certificate generation and management.

The certificates used by SSL/TLS can be generated by the developer or hosting provider, but also can be purchased for public use from a variety of commercial Certificate Authorities (CA). Getting a commercially signed SSL/TLS certificate has an upfront and ongoing financial cost. It also requires proving your identity to the commercial CA and this can take effort and time to accomplish. SSL/TLS Certificates have a finite life and need to be re-issued and re-installed when they expire. While they provide security benefits, they can make diagnosis of Web Services issues harder because of having to deal with encrypted data.

Additionally, there is concern about the validation process that commercial CAs use to prove the identity of the person or organisation that is requesting the certificate. Some feel that some CAs do not provide enough rigor around the validation process and this weakens the non-repudiation value of the certificate. It is best to establish the identity of a particular SSL certificate by discussing the particular attributes of the certificate with partner organisations, rather than merely trusting a certificate at face value. This applies to both commercially and self-signed SSL certificates.

Thus, the entire process of certificate generation, procurement, installation and maintenance reduces the manageability and scalability of this technology. This makes SSL/TLS more suitable for a limited number of point-to-point Web Services operating between trusted partners.

4.5.4.4 References

http://wp.netscape.com/eng/ssl3

http://www.ietf.org/html.charters/tls-charter.html

4.5.5 WS-ReliableMessaging

4.5.5.1 Description

This standard was conceived as a mechanism for message exchange where the following attributes are required:

- Guaranteed delivery.
- No Duplicates.
- Guaranteed ordering.

Web Services calls should be stateless, so this standard works by extending the header of the message to contain the required information to determine its order, its source, and whether this message has been received previously.

4.5.5.2 Why would you use this?

When a Web Service accepts a message, and processes its contents, it is normally unaware of what message will follow the current one. If one message cause some stored data to be altered, and then the next message causes the same data to be changed again, then one would assume the last state of that information is correct. What happens when the first message gets processes after the second? This would leave the data store in an incorrect state!

If your system has a high throughput, and it is likely that similar parts of the data will be modified by many messages, then this standard could be a good safeguard to help maintain your data integrity.

It is highly recommended that this standard or a similar standard be used when you service interacts with a database.

4.5.5.3 Version History of WS-ReliableMessaging standard

WS-ReliableMessaging 1.1 - OASIS, ratified June 2007

4.5.5.4 Issues

While WS-ReliableMessaging is a ratified standard, there is an older rival standard called WS-Reliability that was also ratified by OASIS in November 2004. The vendor involvement in the newer WS-ReliableMessaging standard is essentially a superset of the vendors who developed WS-Reliability. Therefore it appears that WS-ReliableMessaging will ultimately replace WS-Reliability in the marketplace.

4.5.5.5 Namespaces

Specification	Commonly Used Prefix	Namespace URI
Web Services Reliability 1.1 Features, Property, and Compositor Constructs	fnp	http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd
Web Services Reliability 1.1 Service Reference Type	ref	http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd

Web Services Reliability 1.1 Protocol Headers	wsrm	http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd
Web Services Reliability 1.1 Features and Properties	wsrmfp	http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd

4.5.5.6 References

http://www.oasis-open.org/specs/index.php

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-rx

4.5.6 WS-Addressing

4.5.6.1 Description

WS-Addressing supersedes the WS-Routing standard. WS-Addressing was developed to assist clients with message routing while not needing to know the exact destination of their message. In a large organisation, certain message types may be routed to different locations pending on the current configuration of the network, and every client of a Web Service application need not be aware of all these changes.

The client will attach to a message the URI of a specific policy that should be used to determine to destination of the message. The Web Service that receives the message will then use the policy specified to determine the destination location of the message.

4.5.6.2 Why would you use this?

This standard would be applicable in an environment where messages are not directly sent to their destination service. When a middleware system is in place (for example: ESB), this standard can be used to make sure the message is sent to the appropriate service for processing.

4.5.6.3 Version History

WS-Addressing - W3C Recommendation 9 May 2006

4.5.6.4 Issues

Although WS-Addressing is a ratified W3C standard, other Web Service specifications that use it are based on the older W3C draft submission. It is understood that during its development at W3C, the WS-Addressing specification was relatively stable and the ratified version differs only in minor details and errata from draft versions.

4.5.6.5 Namespaces

Specification	Commonly Used Prefix	Namespace URI
Web Services Addressing	wsa	http://www.w3.org/2005/08/addressing

4.5.6.6 References

http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/

4.5.7 WSDM-MUWS and WSDM-MOWS

4.5.7.1 Description

MUWS and MOWS are two separate but related standards.

MUWS - (Management Using Web Services)

MOWS - (Management of Web Services)

OASIS provides a standard called Web Service Distributed Management (WSDM) that dictates a standards schema and methodology for building a management interface into a Web Service. The standard is split into two domains, one part is named Management Using Web Services (WSDM-MUWS) and the other is called Management of Web Services (WSDM-MOWS).

4.5.7.2 Where would you use this?

These standards (or other similar standards) can have benefits when used in an environment where more than one Web Service exists. By building all Web Services this way, you can have a single tool/system that can monitor and manage all your Web Services. Future Web Services that are added to the network that meet the standard will also "plug-in" with the existing management tool/system. This will save money by only requiring one management tool/system, and will increase the efficiency of you system administration team, as only one interface will need to be used to analyse the Web Service.

4.5.7.3 Version History

WSDM - OASIS, ratified March 2005

4.5.7.4 Namespaces

Specification	Commonly Used Prefix	Namespace URI
Web Services Distributed Management: Management Using Web Services Part 1 Schema	muws-p1-xs	http://docs.oasis-open.org/wsdm/2004/12/muws/wsdm-muws-part1.xsd
Web Services Distributed Management: Management Using Web Services Part 2 Schema	muws-p2-xs	http://docs.oasis-open.org/wsdm/2004/12/muws/wsdm-muws-part2.xsd
Web Services Distributed Management: Management Using Web Services	muws-p2-wsdl	http://docs.oasis-open.org/wsdm/2004/12/muws/wsdm-muws-part2.wsdl

Web Services Distributed Management: Management Using Web Services WSDL	muws-events	http://docs.oasis-open.org/wsdm/2004/12/muws/wsdm-muws-part2-events.xml
Web Services Distributed Management: Properties Boolean Match Schema	pbm	http://docs.oasis-open.org/wsdm/2004/12/muws/wsdm-pbm.xsd
Web Services Distributed Management: Management of Web Services	mows-xs	http://docs.oasis-open.org/wsdm/2004/12/mows/wsdm-mows.xsd
Web Services Distributed Management: Management of Web Services	mows-wsdl	http://docs.oasis-open.org/wsdm/2004/12/mows/wsdm-mows.wsdl

4.5.7.5 References

MUWS - http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf

MUWS - http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part2-1.0.pdf

MOWS - http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf

WSLC - http://www.w3.org/TR/wslc/

Appendix A: DIER Standards and Procedures

This section gives an example of Agency-specific policies that relate to the use of Web Services.

The Department of Infrastructure, Energy and Resources (DIER) have Agency-specific standards and procedures that are compatible with those of the Tasmanian Government. The ones directly relevant to Web Service interoperability are the DIER *ICT Strategy Framework* and the DIER *Information Security Plan*.

DIER ICT Strategy Framework

The core of the Information and Communications Technology (ICT) strategy framework consists of a number of objectives that lead naturally to targeted outcomes and principles that apply to the Agency and their relationships with stakeholders. To achieve these outcomes, a number of programs and initiatives within DIER have been begun or identified, including the *DIER Information Security Plan* mentioned below. DIER's ICT Strategy Framework has been carefully crafted to be compatible with the *Tasmanian Government Architectural Principles* and directions.

As with the other standards documents mentioned above the objectives and principles of the *DIER ICT Strategy Framework* should be considered when designing and implementing Web Services.

Many documents associated with the *DIER ICT Strategy Framework* can be obtained from the Information Management Branch by asking for the following information from the DIER Intranet site http://dierlink/corporate_services/information_management/ict/index.html

DIER Information Security Plan

In accordance with the objective of the *DIER ICT Strategy Framework* to “take advantage of the strategic value of information whilst securing information and guaranteeing privacy”, a *DIER Information Security Plan* has been developed. The *DIER Information Security Plan* is designed to improve the protection of DIER’s information assets, minimize potential interruptions to DIER’s business operations and help raise staff awareness of roles and responsibilities.

It is required that any Web Service is compliant with the *DIER Information Security Plan.*

The *DIER Information Security Plan* can be obtained from the Information Management Branch by asking for the following information from the DIER Intranet site http://dierlink/corporate_services/information_management/infostrat/information_security/files/docs/Information%20Security%20Plan%20version%201.0.pdf

This page is intentionally blank

Office of eGovernment
Government Information Services
Department of Premier and Cabinet

Post: GPO Box 123
Hobart TAS 7001
Ph: (03) 6232 7722
Email: egovernment@dpac.tas.gov.au
Visit: www.egovernment.tas.gov.au

Published: 8 September 2008
ISBN: 978 0 7246 5633 2