INTERNATIONAL STANDARD

ISO 24097-1

Second edition
2017-07

Intelligent transport systems — Using web services (machine-machine delivery) for ITS service delivery —

Part 1: Realization of interoperable web services

Utilisation des services du Web (livraison de machine à machine) pour la livraison de services ITS —

Partie 1: Réalisation des services du Web interopérables

Reference number
ISO 24097-1:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by ISO/TC 204, *Intelligent transport systems*.

This second edition cancels and replaces the first edition (ISO 24097-1:2009), which has been technically revised.

A list of all the parts in the ISO 24097- series, can be found on the ISO website.

Introduction

ITS services have been evolving from single functional and limited area specific services, to a broad range of services in which many systems co-operate to provide effective and efficient service provision across a wide area. Today, ITS services are required to communicate not just with other parts of the same ITS service, but between different ITS services, and even with non-ITS services or a user's system directly, e.g. traffic management systems, route guidance systems, homeland security systems, environment protection systems, private freight management systems, etc.

These systems (even those limited to ITS services) are usually deployed in a heterogeneous environment that may use different hardware, operating systems (OS), middleware, and/or development languages. This creates a challenge to realize system coordination across the organizations in a way that is flexible and quick, at a reasonable cost. Web services (WS) are a recent methodology that overcomes these difficulties. Using web service technology for ITS services can significantly simplify and reduce the cost of internet based service provision, which may well affect the speed at which ITS services are deployed.

The World Wide Web Consortium (W3C) defines web services as follows:

"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 (Web Services Description Language)). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards."

Web services require significant functionality, and as a result, an architecture is indispensable. Web service standardization organizations construct standards by SOA. SOA is the evolutional form of distributed computing and object orientation.

By applying SOA-based standards to the ITS services, the following benefits are expected.

From a business viewpoint:

— Increased service value;

— Internationalization; and

— Expansion towards business automation.

From a system development viewpoint:

— Easy and quick development of ITS service coordination and service area expansion;

— More efficient service development (web services enable system developers to focus on the "what" rather than the "how." "How" is covered by a set of standard base tools. This enables quick and easy system software development;

— More reusable software because web service standards have a composable structure, and

— Easier connections to legacy systems.

In the ITS sector, a significant number of messages have been or are being developed (and in some cases standardized). Message standardization is intended to improve system coordination, interoperability and re-use, so the conditions for web services are already considered mature. In addition, the use of web services will increase the flexibility of ITS services to interoperate and communicate beyond the ITS sector and in areas where the delineation between ITS services and general commercial services converge.

From the perspective of web services standards evolution, 2007 was an epoch-making year. WSDL 2.0 became the W3C recommendation. Correspondingly, relevant web service specifications were standardized by open standards bodies (W3C and OASIS). These standards cover all functional layers. By using these standards, the ITS sector has a rigid base for interoperable web services.

ITS service collaboration with other sectors is expected to increase mutual effectiveness. Economic globalization also requires communication across the country, often across the world. All of these collaborations rely on interoperability of services. Interoperability is only achieved based on open international standards.

Web services were developed to use distributed network resources in an interoperable way. However, to realize interoperable web services various capabilities are required.

Using web services (machine-machine delivery) for ITS service delivery has been developed considering these circumstances. ISO 24097 consists of two parts: ISO 24097-1 and ISO/TR 24097-2.

This document focuses on a way to realize interoperable ITS web services. ISO/TR 24097-2 will be an example-based document which will show how to realize interoperable web services.

Intelligent transport systems — Using web services (machine-machine delivery) for ITS service delivery —

Part 1:
Realization of interoperable web services

1 Scope

This document establishes a Service Oriented Architecture (SOA) for the realization of interoperable web services for Intelligent Transport Systems (ITS).

Web service behaviour is described at the metadata level, i.e. a higher level of abstraction, to enable auto-generation of both a 'service requester' program as well as a 'service provider' program. Figure 1 presents the principal entities involved in a web service scenario. They are service provider, service requester, and 'registry'. The registry includes business information and technical information such as interface and policy. Figure 1 also depicts the actions of the service provider and the service requester. A service provider interacts with the registry to enable it to "publish" the provided service. The service is characterized in the form of a web service interface describer in the form of a standardized web service description language (WSDL) and policy (WS-Policy). A service requester interacts with the registry to "discover" a provider for the service he is seeking. That interaction takes place through "Universal Description Discovery, and Integration" (UDDI) dialogue and endpoint reference (EPR). Once the service requester identifies a service provider, he "binds" to the service provider via an SOA protocol.

This document is applicable to inter-ITS sector web services as well as ITS web services for non-ITS users.

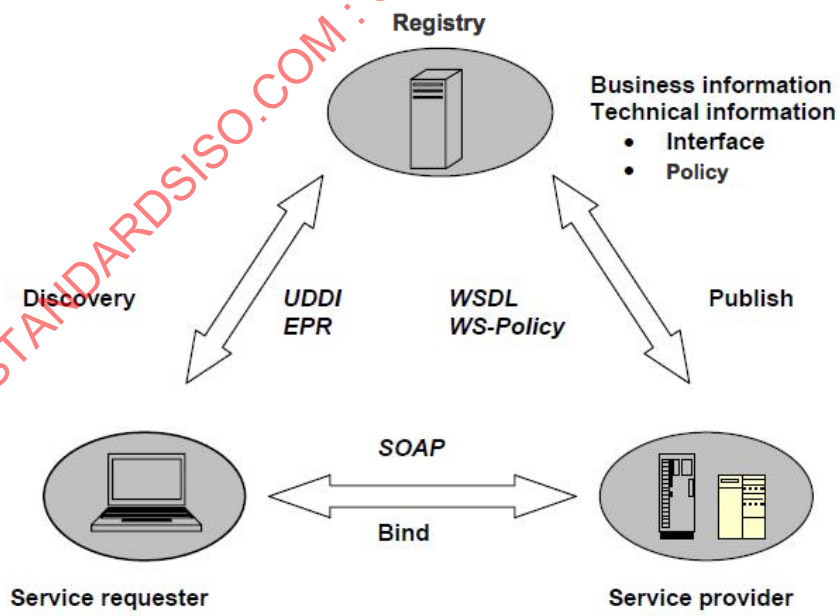

Figure 1 — Web service entities and their relationships

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 14817 (all parts), *Intelligent transport systems — ITS central data dictionaries*

NOTE See Bibliography for general W3C references.

3 Terms, definitions and abbreviated terms

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

General terms of W3C web service definitions can be obtained from the website www.w3.org/tr/ws-arch/ and terms defined in a specific web service standard are also referable.

3.1 Terms and definitions

3.1.1
composability
facility enabling web services to add new features incrementally

3.1.2
domain
functional area in a policy assertion

EXAMPLE Security, reliability, transaction, messaging optimization.

3.1.3
ITS WS
web service that is designed specifically to support ITS services via the internet

3.1.4
international standard web service
web service conformant to this document

3.1.5
platform
hardware, operating system, middleware, and application development language which provide a system environment

3.1.6
policy assertion
element of service metadata which identifies a domain (such as messaging, security, reliability and transaction) specific behaviour

3.1.7
skeleton
elements of service-side code used for receiving remote method calls, invoking them and returning the result to the sender

3.1.8
stub
client code required to talk to a remote service

3.1.9
WS metadata
service metadata
metadata
high-level service description of a web service that controls provision of that service

3.2 Abbreviated terms

BNF	Backus-Naur Form
BP	basic profile (of Web Services Interoperability Organization)
BPEL	business process execution language
DD	data dictionary
DR	data registry
EPR	endpoint reference
HTTP	hypertext transfer protocol
HTTPS	hypertext transfer protocol security
IRI	internationalized resource identifier
MIME	multipurpose internet mail extension
MOF	meta object facility
MTOM	(SOAP) Message Transmission Optimization Mechanism
OID	Object Identifier
OMG	object management group
OSI	open system interconnection
QoS	quality of service
REC	recommendation
RM	reliable messaging
RMI/IIOP	remote method invocation/internet inter-ORB protocol
RPC	remote procedure call
SMTP	simple mail transfer protocol
SOA	service-oriented architecture
TCP/IP	transmission control protocol/internet protocol
tModel	technical model
UDDI	universal description, discovery, and integration
URI	uniform resource identifier
UTF-8(/16)	8(/16)-bit universal character set transformation format
W3C	world wide web consortium
WS	web service
WS-I	web services interoperability (organization)
WSDL	web services description language
XML	eXtensible markup language
XSD	XML schema definition

4 Conformance

There are no explicit conformance tests within this document. Conformance is achieved by conforming to the provisions of the requirements clauses of ISO 24097-1. Specific conformance tests may be added at a subsequent date as an additional part to the ISO 24097- series.

5 Notation

5.1 Prefixes and namespace URI used in core specification

This document uses predefined namespace prefixes throughout; as provided in Table 1. Other prefixes and namespaces (e.g.' Web Services Policy' and 'Web Services Addressing' etc.) are shown in specific clauses of this document.

NOTE 1 The choice of any namespace prefix is arbitrary and not semantically significant (see [Namespaces in XML]). However, the prefix must be unique in any single document.

NOTE 2 For reasons of brevity, not all examples are shown as full schemas. In this case, it is assumed that the namespace has been declared in a parent element.

Table 1 — Namespace prefix and namespace URI

Category	Prefix	Namespace URI
WS-I namespace	`wsi`	http://ws-i.org/profiles/basic/1.1
WSDL 2.0 namespace for WSDL framework.	`wsdl`	http://schemas.xmlsoap.org/wsdl/
WSDL 1.1 namespace	`wsdl11`	http://schemas.xmlsoap.org/wsdl
WSDL namespace for WSDL SOAP binding.	`soapbind`	http://schemas.xmlsoap.org/wsdl/soap/
WSDL namespace for WSDL HTTP GET & POST binding.	`http`	http://schemas.xmlsoap.org/wsdl/http/
Encoding namespace as defined by SOAP 1.1	`soapenc`	http://schemas.xmlsoap.org/soap/encoding/
Envelope namespace as defined by SOAP 1.1	`soapenv`	http://schemas.xmlsoap.org/soap/envelope/
Instance namespace as defined by XSD	`xsi`	http://www.w3.org/2000/10/XMLSchema-instance/
Schema namespace as defined by XSD	`xsd`	http://www.w3.org/2000/10/XMLSchema/
The "this namespace" (tns) prefix as a convention to refer to the current document.	`tns`	(`various`)
All other namespace prefixes are samples only. In particular, IRIs starting with "http://example.com" represents application-dependent or context-dependent IRI.	(`other`)	(`various`)

5.2 Web service syntax notation: BNF pseudo-schemas

BNF pseudo-schemas are used to represent web service syntax.

— The syntax appears as an XML instance, but values in italics indicate data types instead of literal values.

— Characters are appended to elements and attributes to indicate cardinality:

  o "?" (0 or 1)

  o "*" (0 or more)

  o "+" (1 or more)

— The character "|" is used to indicate a choice between alternatives.

— The characters "(" and ")" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

— The characters "[" and "]" are used to call out references and property names.

— Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or

owner, respectively. By default, if a receiver does not recognize an extension, the receiver SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated below.

5.3 XPath 1.0 notation

XPath 1.0 notation is used to specify an XML element and/or attribute.

5.4 Notation of service provider, service consumer combination

There are four combinations of service provider and service consumer. In this document the combination is represented using the “service provider”, “service consumer” notation.

EXAMPLE Traffic service provider, freight industry.

5.5 SOA stack name notation

SOA stack name is represented by bold italics.

EXAMPLE ***messaging***

5.6 Set notation

Set is represented by being embraced in curly brackets (“{ “and”}”).

EXAMPLE Integer set of 1 to 9

{1, 2, 3, 4, 5, 6, 7, 8, 9}

5.7 Tentative IRI expression

Some constructs cannot determine their value when creating standards. In this case, a tentative value is expressed by ***/tentative*** in bold italics. The final value will be given using real IRI.

EXAMPLE WSDL `soapbind:address` (real web service address)

```
<definitions name=…
   xmlns="http://schemas.xmlsoap.org/wsdl"… >
   …
   <service name=…>
      <port name=…>
         <soapbind:addrss location="http://www.example.com/tentative/">
         </port>
      </service>
```

In this case, location is real service IRI and cannot determine the standardization point but it is necessary to be expressed to provide a valid WSDL document.

5.8 Rnnnn (nnnn: digits integer)

Rnnnn is used to display the WS-I Basic Profile requirement identifier number. The expression is shown as “[Rnnnn]”.

6 Requirements

6.1 Basic concept of web services standardization

6.1.1 Web services architecture

Given that web services require significant functionality, an architectural context is essential. Web service standardization organizations construct standards within the framework of an SOA.[4],[5] An SOA is an evolutionary form of distributed computing and object-orientation.

The fundamental SOA philosophy (architecture) is:

— Systems shall be coupled loosely with messages;

— Systems shall be linked dynamically; and

— Systems shall be composable by functional stacks.

In a web service SOA, functional stacks are as follows:

— ***Service composition*** stack: the stack that describes the coordination of business processes. This stack is used to automate real business;

— ***Service description*** stack: the stack that describes the service interface and its related service policy. This stack is used for metadata description;

— ***Quality of service (QoS)*** stack: the stack that ensures message quality, security, and transaction quality;

— ***Messaging*** stack: the stack that describes message behaviour;

— ***Transport*** stack: the stack that transports the message; and

— ***Service publication and discovery*** stack: the stack that publishes a web service and supports its discovery.

Web services are constructed upon an SOA-based open body of standards (see Figure 2 below). Each standard is constructed in a platform independent manner. As a result, a web service (service and client) can communicate with each other independent of their platform. In this case, interoperability is realized when both sides conform to the same standard.

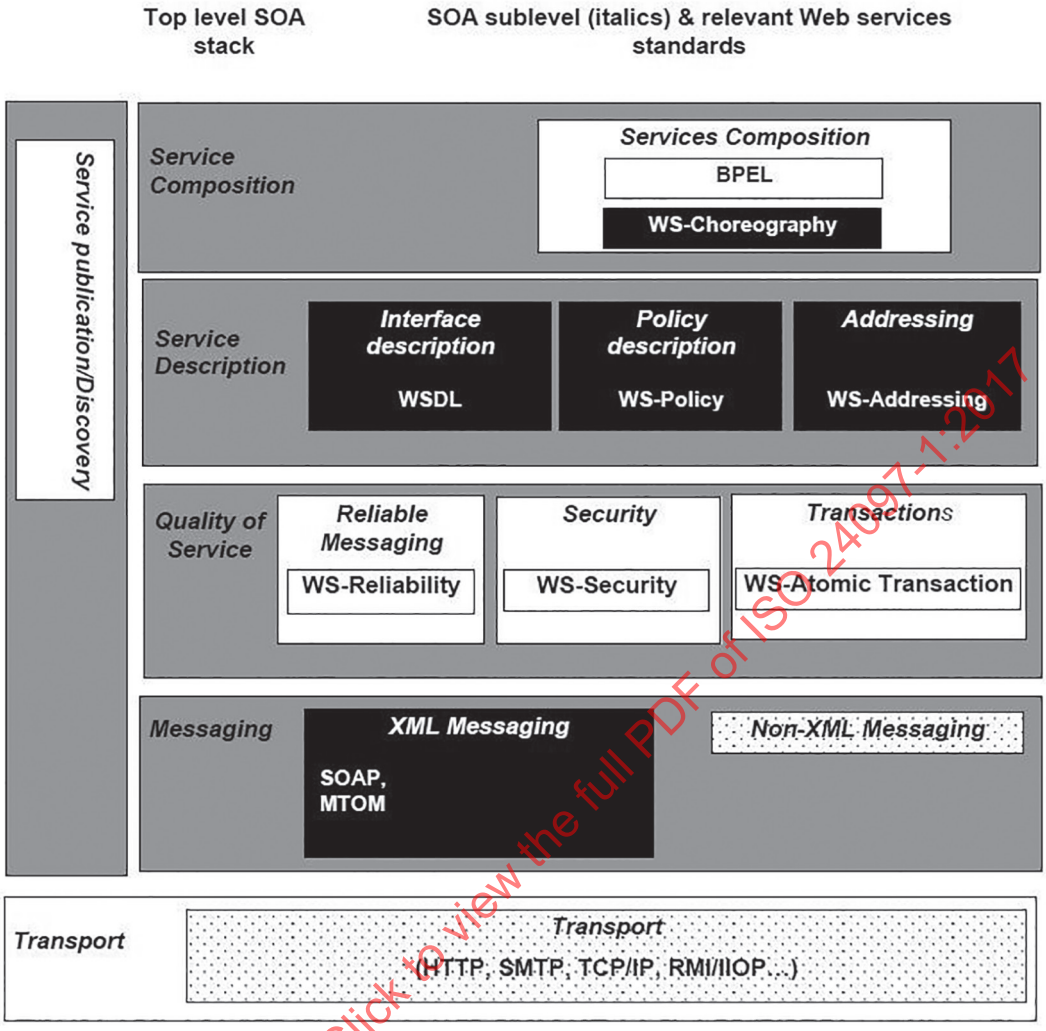

Figure 2 — SOA and its construct standards

NOTE 1 Currently, many software vendors provide a variety of development tools from integrated web service development tools to component level tools. Using these tools enables the developer to make rapid and comparatively easy enhancements.

NOTE 2 Some architects depict ***QoS layer*** as being higher than the upper layer of ***Service description layer***. Other architects depict the reverse. This document describes the ***Service description layer*** as the higher of the two layers. The reason for this is that ***Service description layer*** uses the ***QoS layer*** and it controls the behaviour of the ***QoS*** layer.

6.1.2 International standard web service standardization

Figure 3 depicts an MOF-like view of web services. The dashed arrow shows reference relationships.

M3 Layer (XML + XML Schema and Namespace) provides the syntax of the web service standards. ISO 24531 is the standard schema used within the ITS sector.

M2 Layer (Web service standards, WS-I BP, and this document) provide the rules and guidance for web service development.

M1 Layer (ITS Web service standards) provides rules and guidance for web service development that is particular to ITS. If M1 Layer instances of specific web service (ITS web service) follow this document, basic interoperability should be achieved.

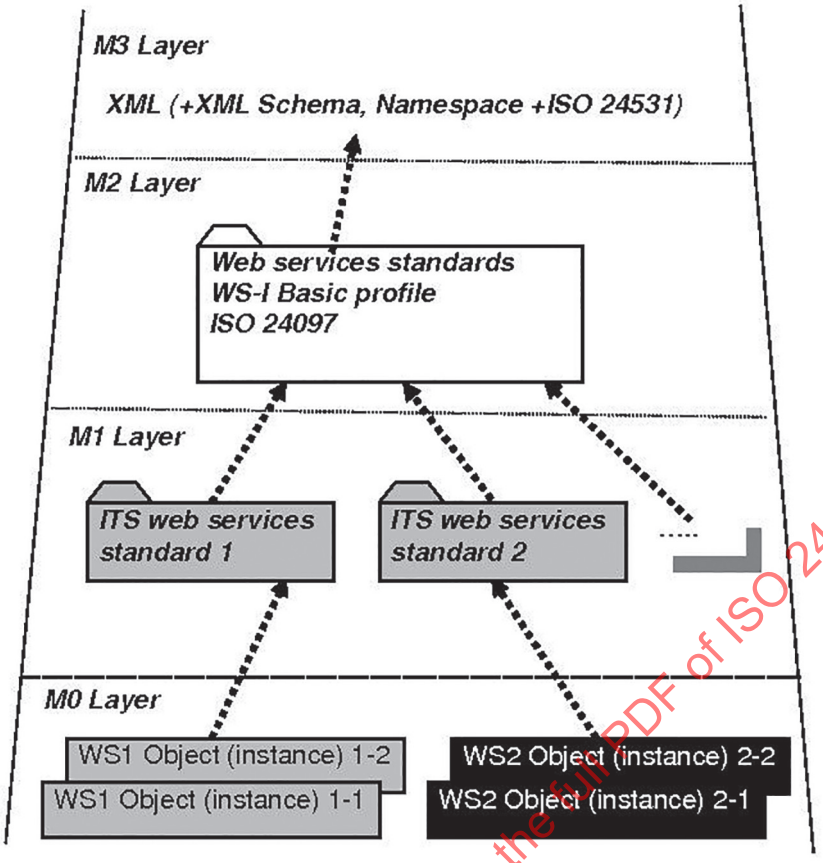

Figure 3 — ITS web service standard structure

6.2 Web service metadata

Web service standards are based on an SOA. This means that web services are constructed using a collection of layered functions. The fundamental layers are depicted in Figure 2. The topmost layer is the "***Service composition***" layer. This layer covers, as the name indicates, the composition of multiple services. As this document covers only a single web service applications, this layer description is not included in this document.

The second highest layer is the "***Service description***" layer. This layer is a metadata layer in general terms. The "***Service description***" layer consists of three sub-components, namely interface description, policy description and addressing. These sub-components are standardized as WSDL, WS-Policy, and WS-Addressing, respectively.

The WSDL standard provides interface information description. The WS-Policy standard covers web services capabilities and constraints. The WS-Addressing standard presents the basis of metadata end point reference (EPR) and some message addressing functionality. Using these standards, ITS web services can respond to real world requirements. WS-Policy and WS-Addressing can be used independently of the particular WSDL version being used.

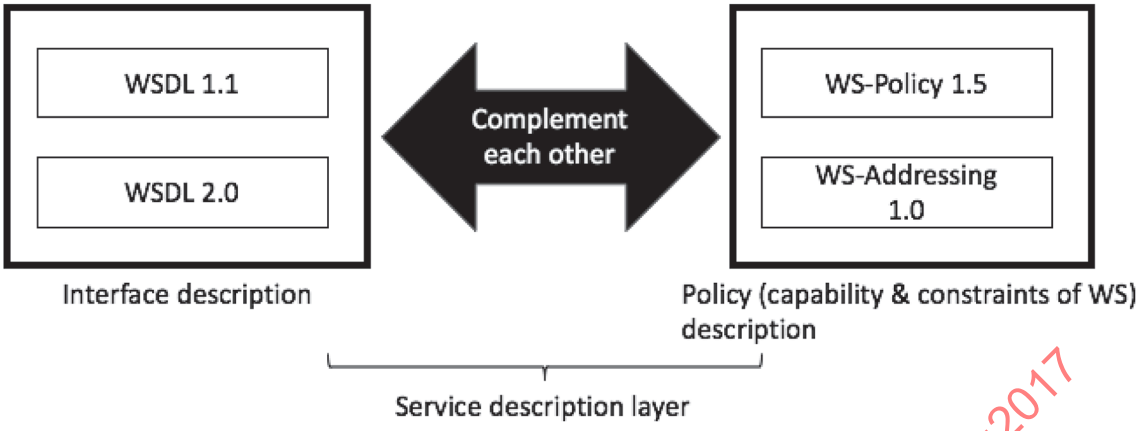

Figure 4 — WSDL, WS-Policy, and WS-Addressing relationship

A single service document is required for an easy understanding of the service by the service consumer and automated tool support. W3C provides two mechanisms: *WSDL extensible points* and *WS-Policy* attachment specification. Extensible points are the logical points where other standard functionality (in this case *WS-Policy*) can be attached to the WSDL description. The *WS-Policy* defines the attaching mechanism to WSDL. Using these mechanisms, web service behaviour is described in a single document. Web services provide enriched functionality in a modular manner.

6.2.1 Common requirements and recommendations for metadata

6.2.1.1 International standard web service metadata description

International standard web service metadata shall be declared in a formalised format.

NOTE A metadata formal expression is a contract between a service provider and a service requester (usually different parties). In addition, formal metadata supports automatic creation of requester and provider programs.

6.2.1.2 Use of utf-8 or utf-16

UTF-8 or UTF-16 shall be used for a metadata declaration.

NOTE Using UTF-8 or UTF-16 is standard for international XML applications.

6.2.1.3 Version upgrade scenario

When metadata changes, support shall be provided for all previous versions to the extent possible.

REASON There are multiple scenarios for upgrading web services (see Figure 5). If the service is for an anonymous user, and/or requires assured response, continued support for the old version should be provided as depicted in the Type 2 scenario (see Figure 5).

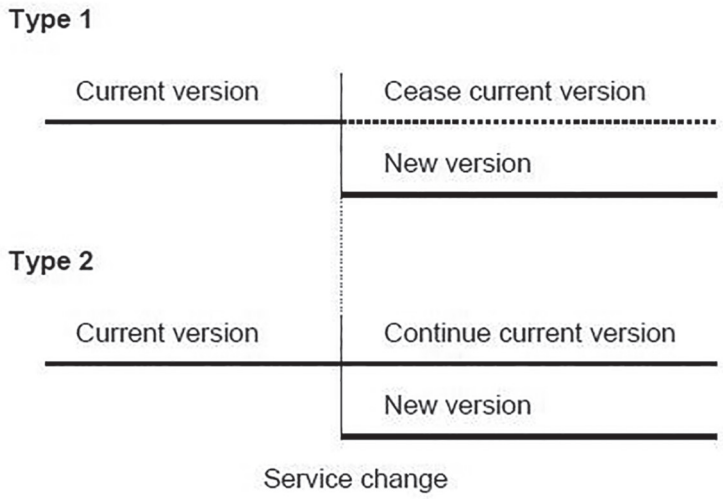

Figure 5 — Version upgrade scenario

6.2.1.4 Interface or policy change

When changing the interface or policy definitions, the analyst shall analyse the impact that the change has on the other (policy or interface) definition.

REASON Web services work only with a consistent interface and policy pair. For example, if a new binding is added in the interface description, it could impact *WS-Policy*. The reverse is also true. The cross check ensures that the interface is consistent with the policy. Any change should be made within the constraints defined in 6.2.1.3. If the change cannot meet the requirements of 6.2.1.3 it becomes a new service, and the old service is retained.

NOTE There are many ways to manage physical files so that they continue to correspond with a change to the WSDL or *WS-Policy*. One method is to change the same file (Type 1 of Figure 6); an alternative is to create new physical files (Type 2). Type 1 in Figure 6 minimises change but the method selected is a service provider's choice.

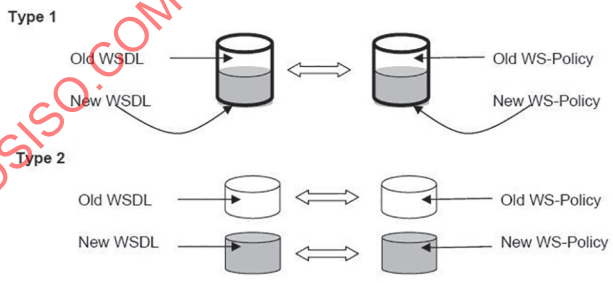

Figure 6 — WSDL WS-Policy change

6.2.1.5 Metadata versioning

Metadata versioning is not required but is highly recommended. Where used, the version number shall be attached to WSDL and *WS-Policy* independently.

REASON Version numbering facilitates detecting changes in the metadata. Independent version numbering for both WSDL and *WS-Policy* is necessary because each can change independently.

NOTE For WSDL versioning rule see 7.2.4. For *WS-Policy* versioning rule see 7.3.2.5.

7 Service description layer

7.1 Service description layer structure

Since web service standards are based on an SOA, a set of standards shall be identified to address all required functions. The ***Service description layer*** shall consist of the following sub-layers: Interface description, Policy description, and Addressing. The layer shall explicitly identify the standards for: WSDL, *WS-Policy*, and *WS-Addressing*. The following subclauses describe how to realize interoperable ITS web services using these standards.

7.2 Service description layer: Requirement and recommendation for interface description sublayer

7.2.1 Role of WSDL

'Web Service Description Language' is the formal service description language of a web service interface. It becomes a contract condition between a service provider and service consumers.

"Contract" also implies that the service provider and a service consumer can create a program from WSDL without ambiguity.

The following describes the role of WSDL in ITS:

— Description of technical conditions agreed between service provider and service consumer, and

— Can provide automatic generation of stub and skeleton using web service development tools. This reduces the software development load for both the provider and consumer.

7.2.2 Multiple WSDL specifications

The current version of WSDL is WSDL 2.0 and this version of this document is focused towards WSDL 2.0. However, this is a recent evolution. Figure 7 shows the evolution of WSDL and SOAP. The current versions (at the time of developing this document) of WSDL and SOAP are WSDL 2.0 and SOAP 1.2, respectively.

Adoption of WSDL 2.0 has been slow; therefore, support for WSDL 1.1 is expected to be employed for some time and has to be accommodated where encountered.

This is important because WSDL 2.0 and SOAP 1.2 are not backwards-compatible. Therefore only (WSDL 1.1, SOAP 1.1), (WSDL 2.0, SOAP 1.2) and (WSDL 2.0, SOAP 1.1) are acceptable combinations. As a result, the user must select the version of the specification.

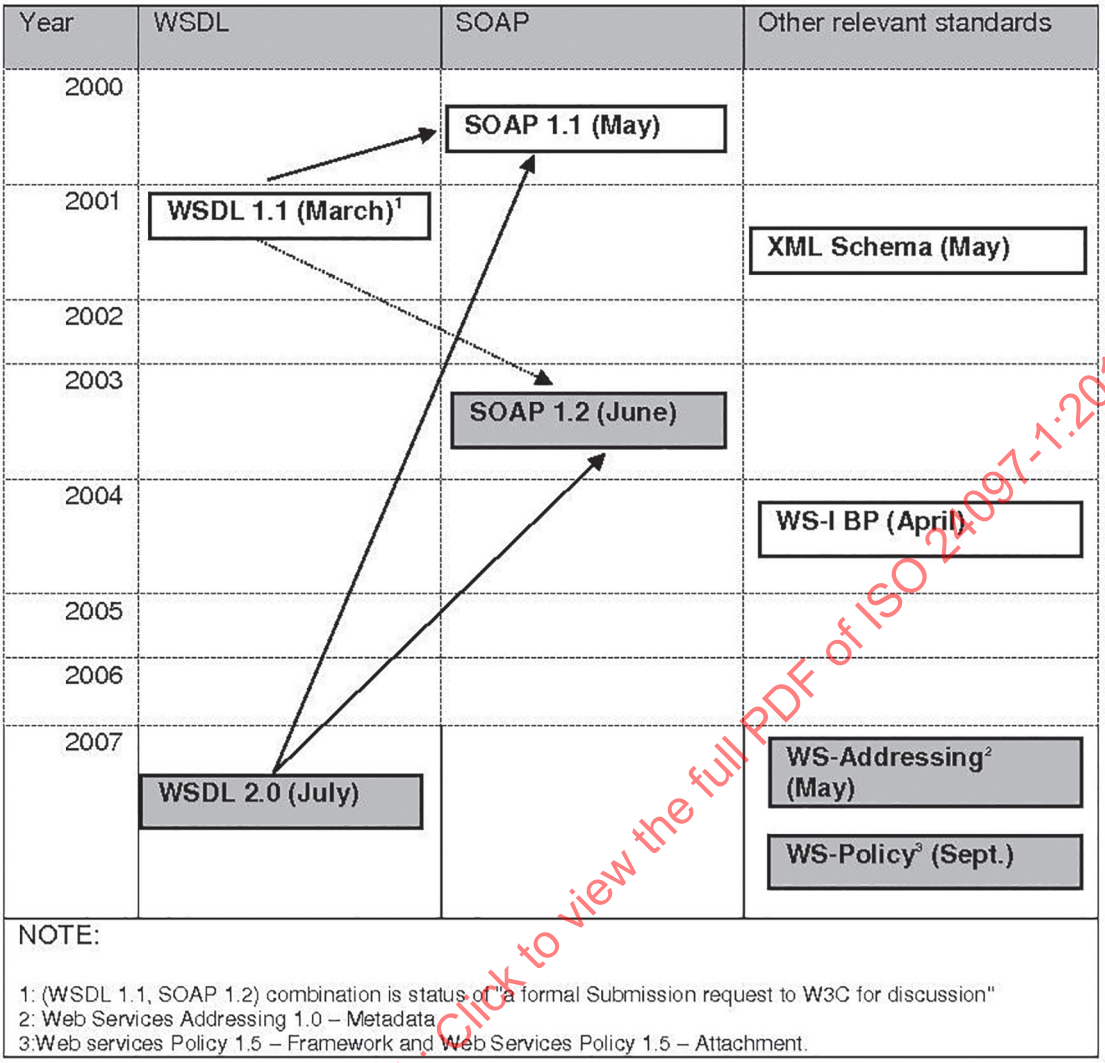

Figure 7 — WSDL and SOAP version correct combination

WSDL 2.0 rules are defined in the body of this document. The corresponding WSDL 1.1 rules are presented in Annex A (normative). Detailed WSDL syntax for both version 1.1 and 2.0 is shown in Annex B (informative).

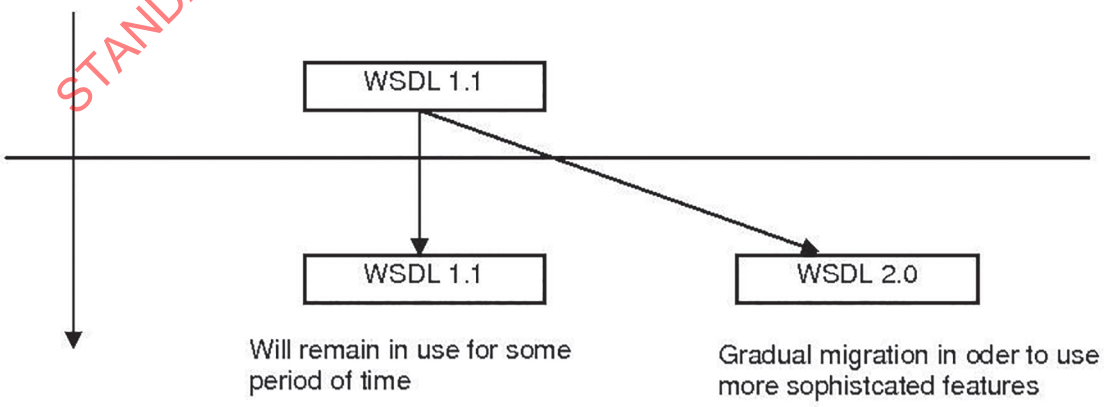

Figure 8 — WSDL migration

7.2.3 WSDL and SOAP relationship

SOAP is a messaging protocol. Both WSDL 1.1 and WSDL 2.0 can select the SOAP protocol (as well as other protocols like direct HTTP or MIME).

WSDL 2.0 can designate SOAP 1.1 and /or SOAP 1.2 (default). SOAP is considered as the communication infrastructure. Therefore, WSDL 2.0 supports the use of SOAP 1.1 (backward compatibility).

In the case of WSDL 1.1, selecting SOAP binding requires the use of SOAP 1.1.

7.2.4 ITS web service *interface* metadata (WSDL) versioning rule

a) An ITS web service *interface* metadata version is represented by the following form:

m.n.a

— m: major version number (`xsd:positiveInteger`),

— n: minor version number (`xsd:nonNegativeInteger`), and

— a: draft version letter (`xsd:NCName`)

b) Version change rule:

— m: the major version number shall be changed to m+1 when the change from the previous version of the service WSDL will cause existing documents to fail to validate;

— n: the minor version number shall be changed to n+1 when the change to the service WSDL will result in all existing documents continuing to validate. However, some new documents which validate against the new version will fail against the old version; and

— a: the draft version letter shall be changed every time when a new draft is issued. If “a” is null, it means the version is official (not draft).

NOTE 1 The XML version is explicitly declared in the XML declaration. Others are specified by namespace attribute.

NOTE 2 The version number can explicitly be declared in WSDL 1.1; wsdl11/definition/@name

7.2.5 Requirement and recommendation for applying WSDL 2.0

This clause determines the requirement for applying WSDL 2.0.

7.2.5.1 WSDL 2.0

WSDL 2.0 offers the following benefits over WSDL 1.1:

a) It provides a more modular structure for easy WSDL development and reuse,

b) It supports an object-oriented style (e.g. inheritance), and

c) It provides a clear definition of an extension point (e.g. ‘Web Service Policy’ is described using this extension point). This enables enriching functionality of web services.

NOTE At the time of this documentation, tool support for WSDL 2.0 is still poor. By comparison, the older WSDL 1.1 has rich tool support. Considering this fact, it is important to select the most appropriate WSDL version in any deployment effort.

7.2.5.2 WSDL 2.0 conformance test

In order to promote interoperability, ITS WS shall use validation via W3C’s WSDL 2.0 web site.

REASON WSDL 2.0 validation to W3C's WSDL website is a minimum condition for interoperable web services.

NOTE This website supports online checking of files.

7.2.5.3 Import message and fault

It is strongly recommended to construct a message as an independent XML document to WSDL and import them to WSDL (but not required).

REASON Separating ***service description layer*** from ***messaging layer*** enables a modular style of development which simplifies maintenance or evolution of systems. Most fault messages are common to various applications and therefore reusable. This helps rapid development of ITS web services. Many countries have already completed message standardization which also supports interoperability and rapid development of web services.

7.2.5.4 WSDL 2.0 name (wsdl:description/@name)

Wherever possible, ISO 14813-1 based service classification should be used in WSDL 2.0 name attribute (strongly recommended but not required).

NOTE This will promote more consistent names and the meaning will be more understandable.

7.3 Service description layer: Requirement and recommendation for policy description sublayer

WSDL allows the formal description of an interface. As previously described, *WS-Policy* complements other facets of web services facilities and requirements (see Figure 4). Describing security, reliable messaging, and/or transaction in a formal manner is indispensable for real world web services. A formal description also supports automation. In this clause, the basics of *WS-Policy* and requirements and recommendations for ITS web services is described.

Table 2 depicts the *WS-Policy* related prefixes and XML Namespace.

Table 2 — Prefixes and XML Namespaces

Prefix	XML Namespace	Specifications
mtom	http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization	[WS-MTOMPolicy]
soap	http://www.w3.org/2003/05/soap-envelope	[SOAP 1.2 Messaging Framework (Second Edition)]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	[WS-SecurityPolicy]
wsa	http://www.w3.org/2005/08/addressing	[*WS-Addressing* Core]
wsam	http://www.w3.org/2007/05/addressing/metadata	[*WS-Addressing* Metadata]
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL 1.1]
wsp	http://www.w3.org/ns/ws-policy	[Web Services Policy Framework, Web Services Policy Attachment]
wss	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[WS-Security 2004]
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512	[WS-Trust]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[WS-Security 2004]

7.3.1 WS-Policy role and syntax

This sub-clause describes *WS-Policy* basics (role and syntax) and ITS web services requirements and recommendations.

7.3.1.1 WS-Policy — Framework

As previously described, web service requirements and constraints relate to topics like security, reliability, transaction, and messaging. It is desirable to pull together all these topics into one document because it facilitates understanding and simplifies software development. These topics (security, reliability, etc.) are organized into a single policy document using the W3C standard, "Web Services Policy 1.5 - Framework". It provides a mechanism to define policy topics in a consistent fashion (see Figure 9).

NOTE Before *Web Services-Policy* 1.5, *WS-Policy* had been developed by vendor groups. The final specification version was version 1.4. It moved to W3C and its version is now 1.5.

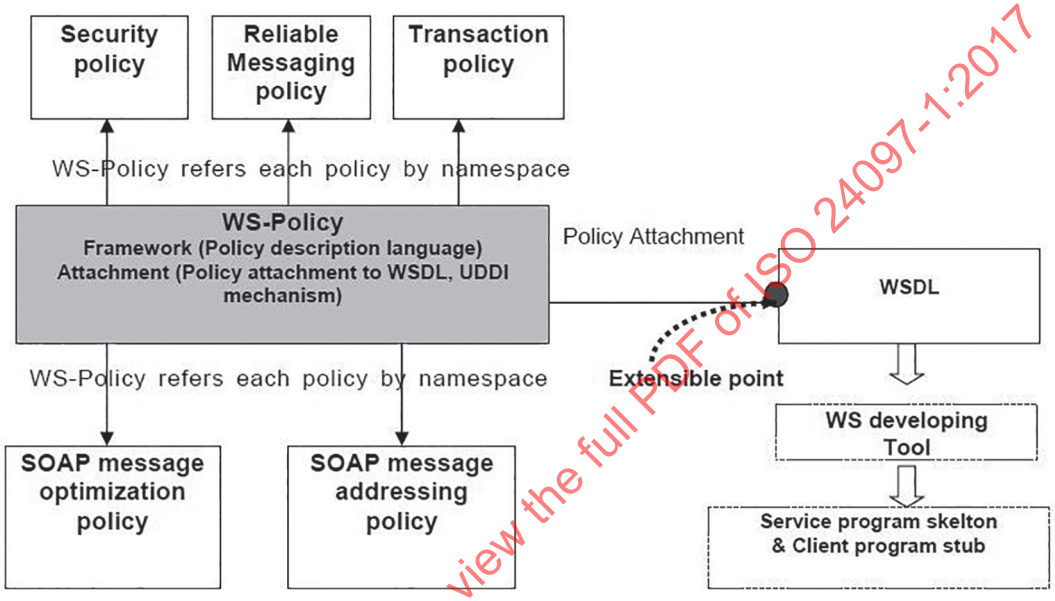

Figure 9 — WS-Policy structure

This language consists of four elements: `wsp:Policy`, `wsp:All`, `wsp:ExactlyOne`, `wsp:PolicyReference` elements, and two attributes: `wsp:Optional` and `wsp:Ignorable`. The namespace prefix "wsp" distinguishes *WS-Policy* vocabularies in XML documents. In this case `wsp:Policy`, `wsp:All`, `wsp:ExactlyOne`, and `wsp:PolicyReference` are used as *WS-Policy* vocabulary. These four elements act as operators. As a computer language operator does something using operand(s), and construct expressions, *WS-Policy* operand(s) do the same thing, namely do something to domain specific topics like security, reliability, transaction, and messaging. Attributes give some property to a policy.

The following listing shows a *WS-Policy* assertion example with namespace reference.

```
<wsp:All>
  <wsam:Addressing>…</wsam:Addressing>
  <mtom:OptimizedMimeSerialization wsp:Optional="true"/>
  <wsp:ExactlyOne>
    <sp:TransportBinding>…</sp:TransportBinding>
    <sp:AsymmetricBinding>…</sp:AsymmetricBinding>
  </wsp:ExactlyOne>
</wsp:All>
```

In this example, the prefix "wsp" means it belongs to the *WS-Policy* vocabulary (`wsp:All`, `wsp:ExactlyOne`) and the attribute(`wsp:Optional`) and their functionality is defined in *WS-Policy*. Other prefixed elements and attributes mean these constructs belong to other policy related vocabularies. `mtom:OptimizedMimeSerialization` element is a direction to apply SOAP Message Optimization. Each prefixed construct functionality is defined in their policy related standards.

WS-Policy prepares the selection operator. In the above example `wsp:ExactlyOne` means that the service consumer can select <`sp:TransportBinding`> or <`sp:AsymmetricBinding`>. These are policy alternatives. The service consumer can select one desirable policy alternative, but the service provider must support both alternatives.

Service policy is one condition for selecting a service provider (others might be service price, quality of contents, etc.). If a service consumer selects the service, she/he must follow the policy. As a result two parties (consumer and provider) can maintain interoperability (see Figure 10).

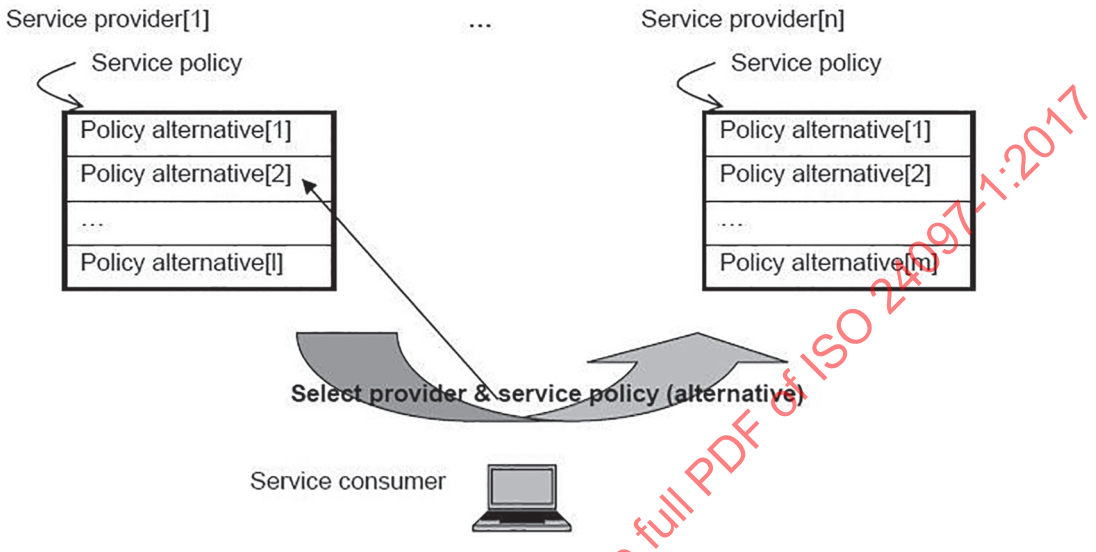

Figure 10 — Service provider and policy alternative selection

7.3.1.2 WS-Policy — Attachment

In addition to a consistent policy description facility ("Framework"), a policy shall be linked to a WSDL as a single description of a web service. "WS Policy 1.5 – Attachment" responds to this requirement. Both WSDL 1.1 and WSDL 2.0 define the extensible point. Using extensible point, a policy may be attached to a WSDL document and UDDI by the service provider. The "Attachment" defines the semantics as well as the attachment syntax. Figure 11 depicts a policy attachment mechanism.

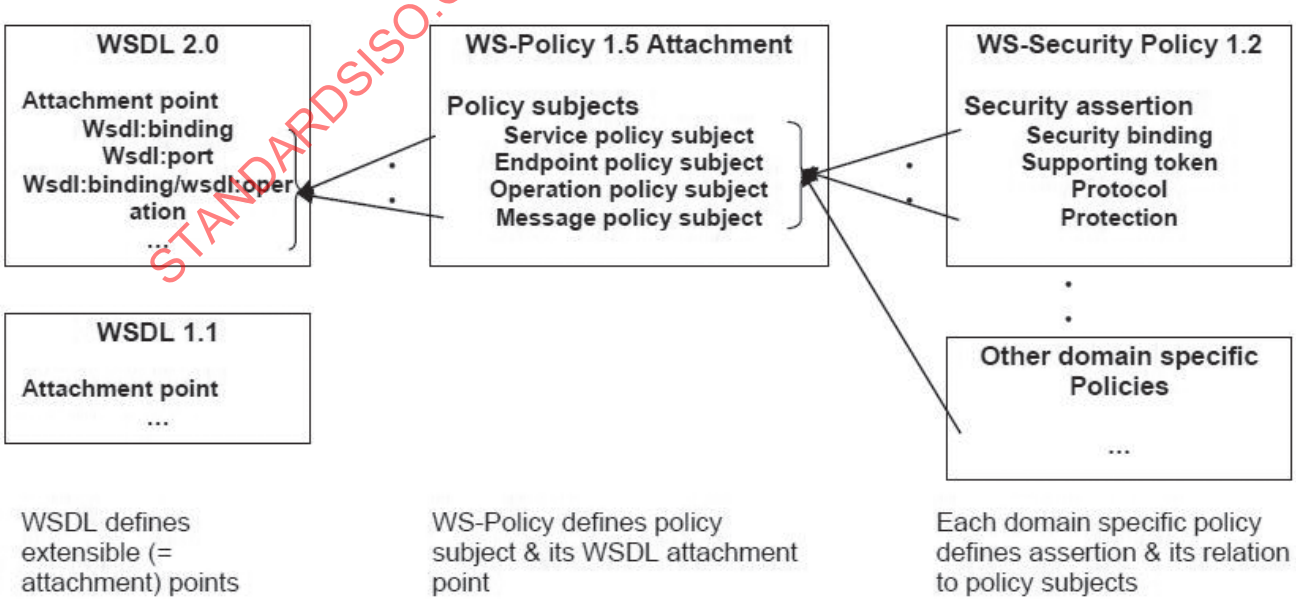

Figure 11 — WSDL, WS-Policy, and domain specific policy

7.3.2 Requirement and recommendation for policy description

This clause describes *WS-Policy* requirements and recommendations for ITS web services.

7.3.2.1 Explicit policy declaration of all applied domain specific policy

In a web service description all relevant domain specific policies shall be declared explicitly and comprehensively.

REASON The *WS-Policy* has a rule that if a web service does not declare a policy explicitly, the service consumer should not suppose anything about an undeclared domain. If a service program uses an undeclared domain, it might cause incompatibility. Therefore, an explicit declaration of the policy is necessary.

NOTE Usually the service provider is supposed to develop a policy such as the following: 1) analyse and listing of the necessary capabilities and restrictions, 2) decide each policy item as mandatory, optional, or ignorable, 3) describe policy based on *WS-Policy*, 4) attach the policy to the WSDL, 5) development. In steps 1) to 4) a comprehensive policy domain is covered and explicitly declared.

7.3.2.2 WSDL and policy description separation

WSDL and policy description separation is strongly recommended (but not required).

REASON Separation increases readability. If the WSDL and *WS-Policy* descriptions are not separated, they will become complex and difficult to understand. Separation increases readability. This also increases maintainability.

7.3.2.3 Policy change

If changing an existing web services policy, the following rules shall apply.

7.3.2.3.1 Adding new assertion

If adding a new assertion has no relation to the current assertion, the new assertion shall be added with `wsp:Optional`="true". Otherwise, if adding the new assertion has an exclusive relationship to the existing policy, 1) the new and current assertions shall be enclosed with `wsp:ExactlyOne`, and 2) the new assertion shall be added as a child element of `wsp:ExactlyOne` tag.

7.3.2.3.2 Changing current assertion to new one

A revision shall not prohibit an existing assertion. This will ensure continued support for existing service consumers. However, if a minor correction or improvement is required, adding a new assertion rule shall be allowed.

REASON The rule is necessary to ensure successive service support.

NOTE If the policy change is achieved with the same physical file (see Figure 4 type 1) a related WSDL change is not required.

7.3.2.4 Non-standardized policy declaration

It is strongly recommended (but not required), even where policy declaration is non standardized, that the policy declaration is explicit.

REASON This will assist in avoiding conflicts with users. For example, logging might endanger privacy. In this case the service provider can declare that it logs information so that users are aware of this potential privacy issue.

NOTE Logging is not standardized as a policy. `wsp:Ignorable` is acceptable predefined vocabulary of `wsp:Policy` In such a case, using a non-standard namespace (policy, security, reliability, etc) the policy processor ignores the expression. So the policy is only for the human reader. In addition using `wsp:Ignorable` attribute the policy has no direct impact on the message sent on the wire, and does not affect interoperability. See how in the following example: "http://www.example.com/tentative" could include a policy description in natural language for the reader.

```
<wsp:Policy>
  <orgX:loggingxmlnX orgX="http://www.example.com/tentative"
  wsp:Ignorable="true"/>
</wsp:Policy>
```

7.3.2.5 *Policy* (QoS) metadata versioning rule

Policy metadata applies the same versioning rule as the ***interface*** metadata does.

NOTE 1 In this sub-clause please read "*interface* metadata (WSDL)" in 7.2.4 as "policy (QoS)" metadata.

Reason:

— the interface metadata versioning rule is considered proper, so is the ***policy*** metadata,

— a service consumer can select the best version of the service easier when there is a common rule, and

— a service provider can manage the service evolution systematically by the common rule.

NOTE 2 *WS-Policy* semantics does not have an explicit versioning attribute. The syntax of `wsp:Policy` element is <`wsp:Policy` ...>. The character "..." means a policy extension point. Therefore a version can be added as per the following example:

```
<wsp:Policy>
  <orgX:version
    xmlns:orgX="http://www.example.com/tentative"
    wsp:Ignorable="true">"1.0.0"
  </orgX:version>
</wsp:Policy>
```

7.4 Service description layer: Requirement and recommendation for addressing sublayer

WW-Addressing for ITS applications shall provide transport-neutral mechanisms to address ITS web service messages. Examples will be provided in Part 2.

8 Quality of service layer

8.1 Quality of service layer: Requirement and recommendation for reliable messaging sublayer

8.1.1 Requirement and recommendation for reliable messaging policy description

A facility to control message reliability shall be identified. Reliable messaging (RM) policy is standardized by OASIS as "WS Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.1". WS-RM Policy grammar is standardized in a consistent manner that the topmost *WS-Policy* claims.

Generally, web service requirements and constraints (i.e. policies) shall be described with various domain specific policies (like reliability, security, transaction, etc). WS-RM constructs a web service policy with a coordination of various domain specific policies, for example, security with RM.

Table 3 depicts reliable messaging relevant namespaces.

Table 3 — Reliable messaging namespaces

Prefix	Namespace URI	Specification
wsdl11	http://schemas.xmlsoap.org/wsdl/	[WSDL 1.1]
wsp	http://www.w3.org/ns/ws-policy	WS-Policy *1.5*
wsrmp	http://docs.oasis-open.org/ws-rx/wsrmp/200702	WS-RM Policy
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS-Security-Utility Schema

Reliable messaging policy grammar is as follows (BNF Pseudo-Schemas):

```
<wsrmp:RMAssertion [wsp:Optional="true"]? ... >
  <wsp:Policy>
   [ <wsrmp:SequenceSTR/> |
        <wsrmp:SequenceTransportSecurity/> ] ?
   <wsrmp:DeliveryAssurance>
     <wsp:Policy>
       [ <wsrmp:ExactlyOnce/> |
         <wsrmp:AtLeastOnce/> |
         <wsrmp:AtMostOnce/> ]
         <wsrmp:InOrder/> ?
       </wsp:Policy>
     </wsrmp:DeliveryAssurance> ?
   </wsp:Policy>

</wsrmp:RMAssertion>
```

Figure 12 depicts how `ExactlyOne`, `AtLeastOne`, `AtMostOne`, and `InOrder` affect messaging reliability. Both designations of `ExactlyOne` and `InOrder` keep no loss, no duplication, and order. Determining which assertion is appropriate depends on an application requirement.

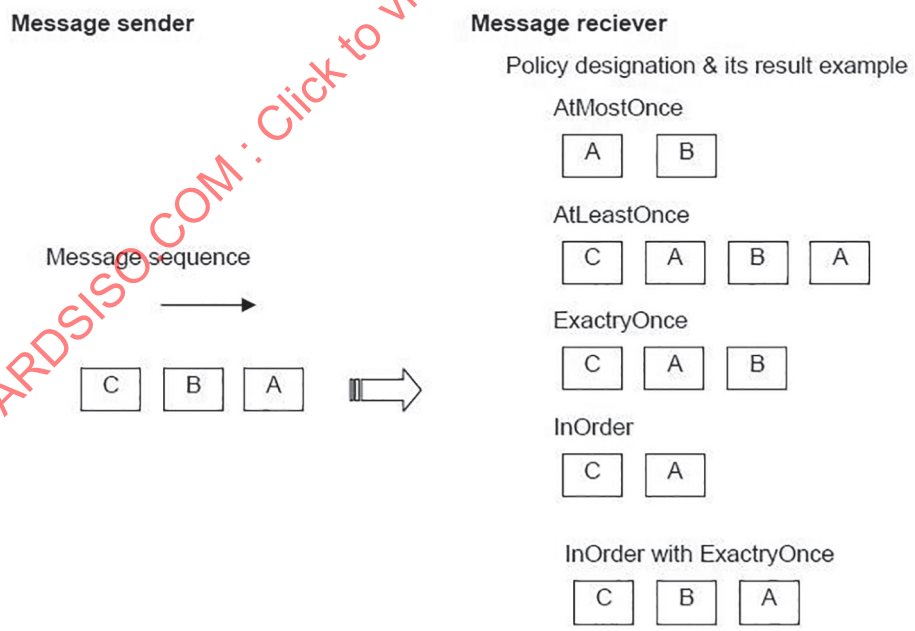

Figure 12 — WS-RM designation and its result example

Considering web services reliability requirements (e.g. whether the web service includes database update or whether the web service is a part of the transaction) and future extension (e.g. this web service will be used as a component of BPEL in future), RM policy shall be specified.

8.2 Quality of service layer: Requirement and recommendation for security sublayer

Web services are a machine-to-machine service delivered over insecure Internet (if security is not applied). Securing web service is realized by applying *WS-SecurityPolicy*. *WS-SecurityPolicy* relates to many standards. Table 4 summarizes relevant standards.

Table 4 — Security related standards

Standard title	Function	Standard body
Web Services Security: SOAP Message Security 1.1 (1 February 2006)	Specification for SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication.	OASIS
WS-SecurityPolicy 1.2 (1 July 2007)	WS's metadata specification that deals with defining "policy assertions".	OASIS
WS-Trust 1.3 (19 March 2007)	Specification that defines how to acquire security tokens.	OASIS
WS-SecureConversation 1.3 (1 March 2007)	This defines extensions that build on WS-Security to provide secure communication.	OASIS
Security Assertion Markup Language (SAML) v2.0 (15 March 2005)	Define an XML framework for exchanging authentication and authorization information.	OASIS
XML-Signature Syntax and Processing (12 February 2002)	Provide integrity, message authentication, and/or signer authentication.	W3C
XML Encryption Syntax and Processing (10 December 2002)	Specifies a process for encrypting data and representing the result in XML.	W3C
Canonical XML Version 1.0 (15 March 2001)	These algorithms guarantee that logically-identical documents produce exactly identical serialized representations.	W3C

OASIS developed a standard named "*WS-Security Policy*". Using this standard, the service providers and service consumers can communicate safely in a compatible manner. As the name of the standard implies, it works under the general control of a *WS-Policy*. As a result the security requirements and constraints can cooperate with other requirements and constraints such as messaging reliability and/or SOAP message optimization (MTOM). MTOM can be used to transmit binary security token effectively.

Security policy assertion is attached to WSDL. Which security policy assertions that can attach to which WSDL attachment points are specified in "*WS-SecurityPolicy*". When attaching a security assertion, the security assertion must be described as a part of a *WS-Policy*. To apply a security policy is not easy. ISO/TR 24097-2 provides an example of how to apply a security policy.

8.3 Quality of service layer: Requirement and recommendation for transaction sublayer

This sublayer relates strongly to the service composition layer. Therefore this subject will be covered in a future document.

9 Messaging layer

9.1 Messaging layer: Requirement and recommendation for XML messaging

9.1.1 Role of SOAP

SOAP defines the protocol between service provider and service consumer programs (including communication to UDDI with web services).

9.1.2 SOAP Structure

SOAP is configured from `soap:envelope`, `soap:header`, and `soap:body`. `soap:envelope` is a container of the SOAP message, namely `soap:header` and `soap:body`. The SOAP header describes the control function involved in communication, and the body for the contents of the service request and service results. If a fault occurs in the process, the fault message is returned as a child element of the `soap:body` element.

A web service is a machine-to-machine communication and is therefore different from web applications. In the case of a web application, if fault occurs the web browser operator is notified and he/she recovers from the fault manually. By comparison, in a web service, the fault is processed by an automated client program. As a result the specification of the kind of fault and its structure is necessary.

9.1.3 SOAP 1.2 relationship to WSDL 1.2

WSDL is a service description stack that enables the description of web services. This implies that WSDL must be able to describe the behaviour of the messaging stack (in this case SOAP control).

Table 5 describes how to specify SOAP behaviour by WSDL.

Table 5 — WSDL 2.0 mapping to SOAP 1.2

WSDL building block	Mapping to the SOAP building block	SOAP building block	SOAP building block functionality
wsdl:types & wsdl: binding	SOAP message header	Header	Control of SOAP message
wsdl:types & wsdl: binding	SOAP message body	Body	Data for ultimate node (service)
wsdl:binding	SOAP	-	Indicate SOAP protocol usage (other protocol selection is possible)

9.1.4 SOAP message transmission optimization (MTOM) policy

SOAP messages often become large. MTOM supports SOAP message transmission optimization (i.e. compressing the serialization). It is said that if the original message size is over 1 KB, this method is effective. "MTOM Serialization Policy Assertion 1.1" works as a domain-specific policy based on *WS-Policy* rules. ITS Web services can apply this feature.

NOTE The namespace URI of the specification is http://www.w3.org/2007/08/soap12-mtom-policy. The conventional prefix and element is `wsoma:MTOM`. Syntax of MTOM is `<wsoma:MTOM wsp:Optional? .../>`

10 Service publication/discovery layer

10.1 Service publication/discovery layer: requirement and recommendation for universal description, discovery, and integration

10.1.1 Role of UDDI

UDDI is used to discover services needed by users. It provides technical conditions for the use of services (see Figure 1). UDDI is a kind of directory service in various usage scenarios. For example, it can be used as a: public worldwide directory service in which there is no limitation of access, (e.g. business to consumer), or in private or semi-private forms (access is limited to inside an enterprise or consortium). In web services, service providers assume that users will develop programs for the use of WSDL. Therefore, data concerning technical information for the use of services (International Resource Identifiers (IRIs) of WSDL with policy and explanatory documents) must be provided. One is UDDI and another is 'End Point Reference (EPR)' of Web services (see Clause 7). EPR includes only metadata.

Business information is excluded. This document is developed for the purpose of client programs to be developed on the fly.

10.1.2 UDDI components

The UDDI comprises:

a) enterprise and organization information (business entity),

b) information (assertion) concerning the relationships between enterprises and organizations,

c) service information (service entity),

d) technical information (binding template) concerning the use of the service, and

e) interface information (technical model (tModel)).

Item information a) is the equivalent of a white-page directory, while items b) and c) categorize businesses according to some context (e.g. equivalent to a yellow-page directory). The information in items d) and e) is technical information for use of the service (e.g. equivalent to a green-page directory).

UDDI is a type of web service, and accessing UDDI API (Application Program Interface) is defined. Some users are likely to have difficulty creating programs for UDDI retrieval, therefore retrieval by the browser is supported as an option.

10.1.3 Public UDDI

A public UDDI is a registry where any subscriber can publish their service and authorized users can access it without a charge. This facility is provided by several enterprises. Public UDDI can be modelled to provide an electronic commerce registry.

Since January 2006 UDDI ceased new service publication. As a result, it has become impossible to use UDDI for this (public) registry. It may be replaced by governmental, standardization organization, or consortium based web pages. For ITS related organizations, both white page and yellow page information are unlikely to be necessary. Web services will most likely be published by ISO/TC 204 working groups, regional, or national ITS relevant organizations (e.g. on web pages).

10.1.3.1 UDDI and ISO 14817 DD/DR

The most important feature of ***Service publication and discovery*** stack is the use of interface information (WSDL) and the reuse of data concepts. An ITS data registry (ISO 14817 based) and UDDI could both be candidates. Table 6 shows a comparison of the two registries.

Table 6 — Comparison of UDDI and ISO 14817

Comparison	Current ISO 14817 DD/DR	UDDI
Main objectives	Get interface information and reuse of ITS data concept	Acquire business service information and get interface information by one registry.
User	ITS stakeholder. If DD/DR opened general consumer	Public UDDI: Enterprise, general consumer Private/semi-private UDDI: limited user, e.g. consortium member or enterprise.
Administration of the registry	ISO TC 204, consortium, or enterprise	Public UDDI: acknowledged organization by UDDI consortium Private/semi-private: Enterprise or consortium: managed by their responsibility
Current deployment status	- Not yet deployed in ISO level - Some county services and some country trials	Deployed

Table 6 *(continued)*

Comparison		Current ISO 14817 DD/DR	UDDI
Fee		Unknown	Public: no fee Private/semi-private: Unknown (software purchase is required)
Registration process		Specified	Specified
Multilanguage support		Possible	Possible
Registration contents		Data concepts and WSDL	Business, service, and technical information
Technical information acquisition		Acquired by various method. Most directly downloadable from IRI meta attribute	Get from IRI of the tModel
Interface description	Message sequence	Possible (by message dialog)	Possible
	Message structure	Possible	Possible
	Fault	Possible (register as a kind of message)	Possible
	Protocol	Directly impossible (but using WSDL IRI possible)	Possible
Explanation of data concept		Explained by description metadata	Possible
Acquire of business information		Not intention of the DR	Possible
Retrieval of service		Not intention of the DR	Possible
Taxonomy use for hit rate increase		Descriptive name context source, architecture reference, architecture name is prepared in metadata	Taxonomy for business category
Policy description		Not specified	Access limitation, security, and change notification can be described by publishing enterprise and/or registry user
Notification of registry contents change		Not specified	User specified auto notification or notification at access
Retrieve		Browser	Program/browser(optional)

At an abstract level the ISO 14817 data registry and UDDI have almost the same functionality. In this document, use of the UDDI is not considered an essential condition. However, it is possible that the UDDI could be used for the Web services directory.

Preferences for using an ISO 14817 data registry are:

— Some countries already have ISO 14817 based registries; and

— In general, a data base is only effective if it includes almost all required information (see Figure 13).

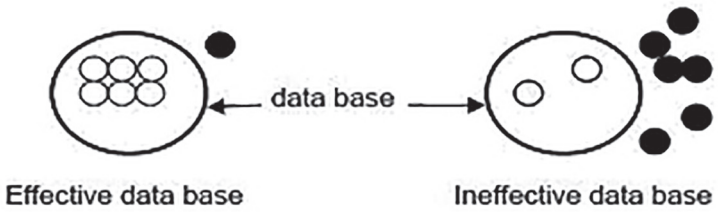

Figure 13 — Effective data base

NOTE Nonetheless this document does not restrict usage of UDDI. It is a matter of choice by ITS stakeholders.

If an ITS stakeholder decides to use an ISO 14817 data registry, the following clause describes conformance to this document.

10.1.4 Requirement and recommendation for *service registration* stack

10.1.4.1 wsdl20:definitions or wsdl:description registration

If a `wsdl20:definitions` or `wsdl:description` element is created, it should be registered with an ISO 14817 complaint data registry, a UDDI or an EPR. If ISO 14817 is applied, it should be registered as an object class. A downloadable URI (IRI) meta attribute is mandatory. As an OID meta attribute `wsdl:definitions` name attribute shall be used.

REASON Use for web service construction.

EXAMPLE See Table 7.

Table 7 — Example of WSDL 2.0 registration

14817 meta-attribute	14817 registration value
Descriptive name	ISO-standard-24531-wsdl-2v-1-0
ASN.1 object identifier	{iso standard 24531 wsdl 2_0 }
Uniform resource locator	http://www.example.com/iso/standard/24531/wsdl/2/v1.0.wsdl
Definition	Service xxxx WSDL 2.0 description
Descriptive name context	ISO-24531
Data concept type	Object class
Standard	ISO 24531
Abstract	False
Referenced data element	ISO-standard-24531-schema-1-v-1-0.ISO ISO-standard-24531-schema-1-v-1-0.ISOStructure ISO-standard-24531-schema-1-v-1-.-0.TC ISO-standard-24531-schema-1-v-1-.-0.TCStructure ISO-standard-24531-schema-1-v-1-.-0.Standard ISO-standard-24531-schema-1-v-1-.-0.StandardStructure

10.1.4.2 Registration of messages

Messages (`wsdl:input`, `wsdl:output`) shall be registered in an ISO 14817 based registry.

REASON This will allow the reuse of the message.

10.1.4.3 Registration of fault

Fault shall be registered in an ISO 14817 based registry.

REASON This will allow the reuse of the message.

10.1.4.4 Object Identifier (OID)

OID of a message and fault shall be an XML OID (delimiter "_").

REASON Message and fault are described by XML Schema. ISO 24531 defines how to give OID for Schema and its constructs.

Annex A
(normative)

Principles and evolution of WSDL from version 1.1 to 2.0

This Annex provides information about the basics of WSDL and its evolution from WSDL 1.1 to WSDL2.0. The provisions in this annex are normative in the case where WSDL 1.1 is used.

A.1 WS-I basic profile

It is to be noted that web services standards (SOAP 1.1, WSDL 1.1, and UDDI v.x) contain some ambiguities and conflicts among them. One reason for this is because WSDL 1.1 and SOAP 1.1 were developed before the XML Schema recommendation. XML Schema is a base component of the web services standard. In order to promote web services interoperability across vendors software, WS-I, vendors and end-user organizations, cooperatively developed the basic profile (BP). The WS-I "Basic Profile" (BP) provides the requirements to which WSDL and SOAP are to comply.

Figure A.1 depicts the relationships among various standard bodies' specifications, including WS-I BP, and this document.

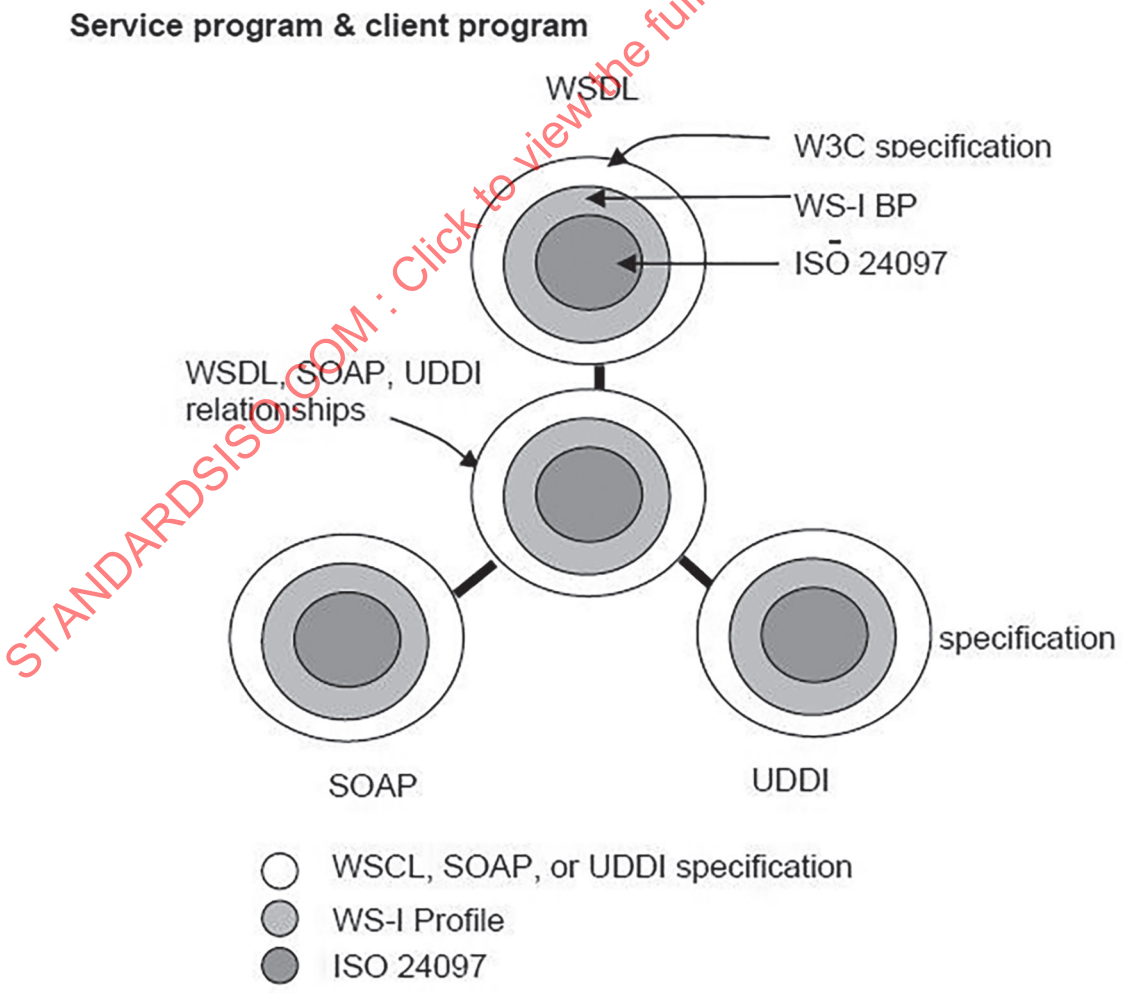

Figure A.1 — W3C recommendations, WS-I BP, and ISO 24097 relationship

The relationship (W3C specification) serves as a base for the WS-I BP specification which serves as a base for this document. This means WS-I BP constrains this document, and the WS-I BP is constrained by the W3C specification.

WS-I is not necessary for the case of WSDL 2.0 and SOAP 1.2, because both are W3C recommendations.

A.2 WSDL 1.1

This section describes WSDL 1.1 basics and requirements for WSDL 1.1 in providing ITS services. The Web Service- Interoperability Basic Profile (WS-I BP) requires:

"**[R0001]** Either an INSTANCE's WSDL 1.1 (or 2.0) description, its UDDI binding template, or both MUST be available to an authorized consumer upon request."

A.2.1 WSDL 1.1 structure

Figure A.2 exhibits WSDL 1.1 structure. As depicted in Figure A.2, WSDL is a collection of building blocks, which are 'types', 'message', 'portType', 'binding', and 'service'.

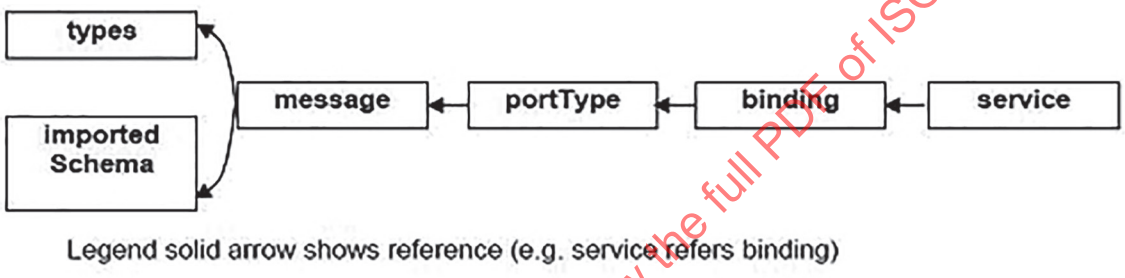

Figure A.2 — WSDL 1.1 structure

All the "Building blocks" have a name attribute which is referred to by another building block (except for imported schema, in this case, the Namespace plays the same role as the name).

"Building blocks" are constructed by abstract parts (types, messages, and portTypes) and concrete parts (bindings and services).

A named "Building block" enables flexibility, reusability and extensibility for web services. It provides:

— Flexibility: one can select the best method from multiple alternatives (e.g. one can select SOAP/HTTP binding or HTTP binding);

— Reusability: one can reuse building blocks; and

— Extensibility: one can add new functional block (s) (e.g. add digital signature for a web service security).

NOTE In WSDL xs:import and wsdl:import are supported for increasing reusability of other XML Schema and WSDL documents, respectively.

A.3 Fault

Fault has the following structure (general form):

```
<?xml version='1.0' encoding='UTF-8' ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://example.com"
   targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   elementFormDefault="qualified">
```

```
   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
         schemaLocation="http://www.w3.org/2001/xml.xsd" />

   <xs:annotation>
     <xs:documentation>
         Envelope elemnt and Header element are omitted
     </xs:documentation>
   </xs:annotation>

  <xs:element name="Body">
      <xs:complexType>
         <xs:choice>
            <xs:any>
               <xs:annotation>
                 <xs:documentation>
                    This is a response from service
                    whithout error
                 </xs:documentation>
               </xs:annotation>
            </xs:any>
            <xs:element name="fault"
              type="soapenv:FaultStrucutre">
            </xs:element>
         </xs:choice>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="FaultStrucutre">
      <xs:sequence>
         <xs:element name="faultcode" type="xs:QName" />
         <xs:element name="faultsrting"
           type="soapenv:Faultsrting" />
         <xs:element name="faultactor" type="xs:anyURI" />
         <xs:element name="detail"
            type="soapenv:detail" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="Faultsrting">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute ref="xml:lang" use="optional" />
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="detail">
      <xs:sequence >
         <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax" />
      </xs:sequence>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>
</xs:schema>
```

The meaning of `soapenv:fault` children elements are summarized in Table A.1.

Table A.1 — `soapenv:fault` children and role

`soapenv:fault` children elements	Roll	Mandatory/optional if fault happens
faultcode	The fault identification	mandatory
faultstring	Human readable explanation	mandatory
faultactor	Fault originator IRI	If header fault happens, mandatory
detail	Application specific (client) error information related `soapenv:Body` element. Service provider (application standard developer) must decide soap `envelope:detail` structure and its contents.	If body element fault happens, mandatory

The meaning of pre-defined `soapenv:fault` elements are summarized in Table A.2.

Table A.2 — Pre-defined `soapenv:faultcode`

soapenv:faultcode	Reason	Error correction
VersionMismatch	Service side found an invalid namespace for the SOAP Envelope element	Mainly client side
MustUnderstand	SOAP Header element either not understand or not obeyed	Mainly client side
Client	The message (body) was incorrectly or did not contain the appropriate information in order to succeed.	Client side
Server	The message could not be processed for reasons not directly attributable to the contents of the message itself but rather to the server side	Service side

A.3.1 Fault code

As a `soapenv:faultcode` contents limitation of only "`soapenv:VersionMismatch`", "`soapenv:MustUnderstand`", "`soapenv:Client`", and "`soapenv:Server`" is strongly recommended (but not required).

REASON "Fault" is used to communicate machine-to-machine in the event of a fault occurrence. The W3C intention for `soapenv:faultcode` is to indicate the classification of fault. This `soapenv:faultcode` is processed by a client program, therefore the limitation of value makes the client program simple. In addition using with `soapenv:faultstring` and `soapenv:detail` seems sufficient to tell client how to deal with the fault.

NOTE 1 SOAP 1.1 note does not define the SOAP schema explicitly. WS-I defined the schema to clarify this structure. The schema location is http://schemas.xmlsoap.org/soap/envelope/ . In this document we refer to this schema.

NOTE 2 The following is an example of Fault.

```
<?xml version='1.0' encoding='UTF-8' ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://example.com"
   targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   elementFormDefault="qualified">

   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
          schemaLocation="http://www.w3.org/2001/xml.xsd" />

   <xs:annotation>
      <xs:documentation>
         Enbelope elemnt and Header element are ommited
      </xs:documentation>
   </xs:annotation>

   <xs:element name="Body">
```

```
      <xs:complexType>
         <xs:choice>
            <xs:any>
               <xs:annotation>
                  <xs:documentation>
                    This is a response from service
                    whithout error
                  </xs:documentation>
               </xs:annotation>
            </xs:any>
           <xs:element name="fault"
              type="soapenv:FaultStrucutre">
           </xs:element>
         </xs:choice>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="FaultStrucutre">
      <xs:sequence>
         <xs:element name="faultcode" type="soapenv:faultcodeValue" />
         <xs:element name="faultsrting"
            type="soapenv:Faultsrting" />
         <xs:element name="faultactor" type="xs:anyURI" />
         <xs:element name="detail"
            type="soapenv:detail" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="Faultsrting">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute ref="xml:lang" use="optional" />
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:simpleType name="faultcodeValue">
      <xs:restriction base="xs:string">
         <xs:enumeration value="soapenv:VersionMismatch" />
         <xs:enumeration value="soapenv:MustUnderstand" />
         <xs:enumeration value="soapenv:Client" />
         <xs:enumeration value="soapenv:Server" />
      </xs:restriction>
   </xs:simpleType>

   <xs:complexType name="detail">
      <xs:sequence>
         <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax" />
      </xs:sequence>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>

</xs:schema>
```

A.3.1.1 Simplified ITS fault

The following fault messages are recommended (but not required).

Table A.3 — Recommended fault message

soapenv:faultcode	soapenv: faultsrting	soapenv:fault actor	soapenv:detail	Note
Soapenv: VersionMismatch	Your input SOAP version mismatch to the service side SOAP version	null	null	Message fixed
Soapenv: MustUnderstand	Your input SOAP header soapenvelope: MustUnderstand Could not understand	Fault node IRI	null	Message fixed
Soapenv:Server	Service side server error. Please try again later	null	null	Message fixed
Soapenv:Client	Your input data incorrect	Null	Each message is decided and standardized by TC 204 working group or developer	

If a service provider detects a fault, the service will be stopped immediately and the service consumer shall be informed. File processes (e.g. write, update etc.) will not be processed. The important thing before retrying is to give sufficient information about who (service requester or service provider) caused the fault, and which part caused the fault. The first three fault messages in Table A.3 seem sufficient. Only application related messages shall be defined. Standard fault messages are required for machine-to-machine communication.

EXAMPLE For the upper three rows of Table A.3 (soapenv:VersionMismuch, soapenv: MustUnderstand, and soapenv:Server) the schema looks like this.

```
<?xml version='1.0' encoding='UTF-8' ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:cf="http://example.com/commonFault"
   targetNamespace="http://example.com/commonFault"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <xs:import
      namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="http://www.w3.org/2001/xml.xsd" />

   <xs:complexType name="VersionMismatch">
      <xs:sequence>
         <xs:element name="faultcode" type="xs:QName"
            fixed="soapenv:VersionMismatch" />
         <xs:element name="faultsrting"
            type="cf:Faultsrting"
            fixed="Your input SOAP version mismatch to the
            service SOAP version" />
         <xs:element name="faultactor" type="xs:anyURI"
            minOccurs="0" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="MustUnderstand">
      <xs:sequence>
         <xs:element name="faultcode" type="xs:QName"
            fixed="soapenv:MustUnderstand" />
         <xs:element name="faultsrting"
            type="cf:Faultsrting"
            fixed="Your input SOAP header can not unserstand" />
         <xs:element name="faultactor" type="xs:anyURI"
            minOccurs="0" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="Server">
```

```
      <xs:sequence>
         <xs:element name="faultcode" type="xs:QName"
            fixed="soapenv:Server" />
         <xs:element name="faultsrting"
            type="cf:Faultsrting"
            fixed="Service side server error. Please try again later" />
         <xs:element name="faultactor" type="xs:anyURI"
            minOccurs="0" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="Faultsrting">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute ref="xml:lang" use="optional" />
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

</xs:schema>
```

EXAMPLE Client fault schema is as follows

```
<?xml version='1.0' encoding='UTF-8' ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:af="http://example.com/clientFault"
   targetNamespace="http://example.com/clientFault"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
          schemaLocation="http://www.w3.org/2001/xml.xsd" />

   <xs:complexType name="ApFaultStrucutre">
      <xs:sequence>
         <xs:group ref="af:ClientFault" />
         <xs:element name="detail"
            type="af:detailStructure"
            minOccurs="1" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>

   <xs:group name="ClientFault">
      <xs:sequence>
         <xs:element name="faultcode" type="xs:QName"
             fixed="soapenv:Client”/>
         <xs:element name="Faultsrting"
            type="af:Faultsrting" fixed="InputDataError"/>
      </xs:sequence>
   </xs:group>

   <xs:complexType name="Faultsrting">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute ref="xml:lang" use="optional" />
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="detail">
      <xs:sequence >
         <xs:any namespace="##any" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
      </xs:sequence>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>

</xs:schema>
```

EXAMPLE The upper schema is used in wsdl11:definitions as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl11:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
 xmlns:tns="http://www.example.com/fualtWSDL11File/"
 xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 name="faultWSDL11"
 targetNamespace="http://www.example.com/fualtWSDL11:File/"
 xmlns:cf="http://example.com/commonFault"
 xmlns:af="http://example.com/clientFault">

<wsdl11:types>
<xs:schema targetNamespace="http://www.example.com/fualtWSDL11:File/">

<xs:import
 namespace="http://www.w3.org/XML/1998/namespace"
 schemaLocation="http://www.w3.org/2001/xml.xsd" />

<xs:import
 schemaLocation="http://example.com/commonFaultXMLSchema"
 namespace="http://example.com/commonFault" />

<xs:import
 schemaLocation="http://example.com/clientFault"
 namespace="http://example.com/clientFault" />

<xs:annotation>Common fault</xs:annotation>
<xs:element name="VersionMismatch" type="cf:VersionMismatch"/>
<xs:element name="MustUnderstand" type="cf:MustUnderstand"/>
<xs:element name="Server" type="cf:Server”/>

<xs:annotation>Client fault</xs:annotation>
<xs:element name="Client" type="af:ApFaultStrucutre"/>

</xs:schema>

</wsdl11:types>
</wsdl11:definitions>
```

A.4 Requirement and recommendation in the case of use of WSDL 1.1 for its application

This sub-clause describes WSDL 1.1 requirements and recommendations.

A.4.1 Requirement and recommendation for applying WSDL

In WSDL, including SOAP binding is mandatory.

REASON If WSDL designates SOAP binding, the web services processor uses SOAP. SOAP can include policy information like security and/or reliability. The web services policy is constructed using SOAP. The SOAP message including policy is automatically processed by the receiver. WSDL can include many bindings, so one binding should include a SOAP binding.

A.4.2 Conformance to the WS-I BP

A public ITS web service shall provide at least one service conformant to the WS-I BP.

REASON As described in the previous clause, ITS web services includes public services. In this case, the service requester may demand to access an ITS web service using the most common specification. Therefore at least one basic profile based service is mandatory.

NOTE This is also commonly recommended for ITS web services. Because of increasing data exchanges and increasing coordination with other sector services, WS-I BP compliance is the easiest way to achieve interoperability (e.g. coordination with homeland security).

A.4.3 WS-I BP takes over WSDL 1.1 and SOAP 1.1

If a conflict exists among the WS-I BP and web services standards, WS-I BP shall take over WSDL 1.1 and SOAP 1.1.

