INTERNATIONAL STANDARD

ISO 24097-1

First edition
2009-09-15

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 24097-1:2009(E)

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 24097-1 was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

ISO 24097 consists of the following parts, under the general title *Intelligent transport systems — Using web services (machine-machine delivery) for ITS service delivery*:

— *Part 1: Realization of interoperable web services*

An example-based document on the elaboration of interoperable ITS web services will form the subject of part 2.

Introduction

Intelligent Transport Systems (ITS) services have been evolving from single functional and limited area specific services, to a broad range of services in which many systems cooperate to provide effective and efficient service provision across a wide area. In today’s world, ITS services are required to communicate not just with other parts of the same ITS service provision, but between different ITS services and even with non-ITS services or a user’s system directly. Some examples of these systems are communications between traffic management, route guidance systems, homeland security systems, environment protection systems and private freight management systems.

These systems (even those limited to ITS services) are usually deployed in a heterogeneous circumstance, use different hardware, different operating system (OS), middleware, or development languages. This therefore creates a challenge in order to realize system coordination across the organizations in a way that is flexible, quick and at 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 can affect the level and speed of take up of use of ITS services.

The World Wide Web Consortium (W3C) defines WS as follows: “A web service is a software system designed to support interoperable machine-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.”

WS require quite a lot of functionalities and as a result, an architecture is indispensable. Web service standardization organizations construct standards by Service-Oriented Architecture (SOA). SOA is an evolutional form of distributed computing and object orientation.

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

From a business viewpoint:

— increased service value;

— internationalization;

— expansion towards business automation.

From a system development perspective:

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

— WS enable system developers to focus on the “what” not the “how.” “HOW” is covered by standard base tools. This enables quick and easy system software development;

— composable structure of web service standards promote reusability of software;

— easy connection to a legacy system.

In the ITS sector, message standardization of many applications has already been completed, are well advanced, or are determined regionally. Message standardization is intended to improve system coordination, interoperability and re-use and so the conditions for WS are considered already mature. In addition, the use of WS 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 the evolution of standards in WS, 2007 was an epoch-making year. WSDL 2.0 became the W3C recommendation. Correspondingly, relevant web service specifications were standardized by open standardization bodies (W3C and OASIS). These standards cover all functional layers. In using these standards, the ITS sector has a rigid base for interoperable WS.

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.

WS were developed to use distributed network resources in an interoperable way. However, to realize interoperable WS, various functionalities are required. ISO 24097 (all parts) has been developed based on these circumstances.

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

Part 1: Realization of interoperable web services

1 Scope

This part of ISO 24097 establishes a Service-Oriented Architecture (SOA) for the realization of interoperable Intelligent Transport Systems (ITS) web services (WS). Web service behaviour is described at the metadata level (i.e. a higher level of abstraction) to enable auto-generation of both a “Service requestor” program, as well as a “Service provider” program.

The principal entities involved in a web service scenario are “Service provider“, “Service requestor” and “Registry” (see Figure 1). The registry includes business information and technical information such as interface and policy. A service provider interacts with the registry to enable it to “publish” the service he/she is able to provide. The service is characterized by a web service interface describer in the form of a standardized web service description language (WSDL) and policy (WS-Policy). A service requestor interacts with the registry to “discover” a provider for the service he/she is seeking. That interaction takes place through Universal Description Discovery and Integration (UDDI) dialogue and endpoint reference (EPR). Once the service requestor identifies a service provider, he “binds” to the service provider via an SOA protocol.

NOTE Figure 1 depicts the actions of the service provider and the service requestor.

This part of ISO 24097 is applicable to inter-ITS sector WS, as well as ITS WS for non-ITS users.

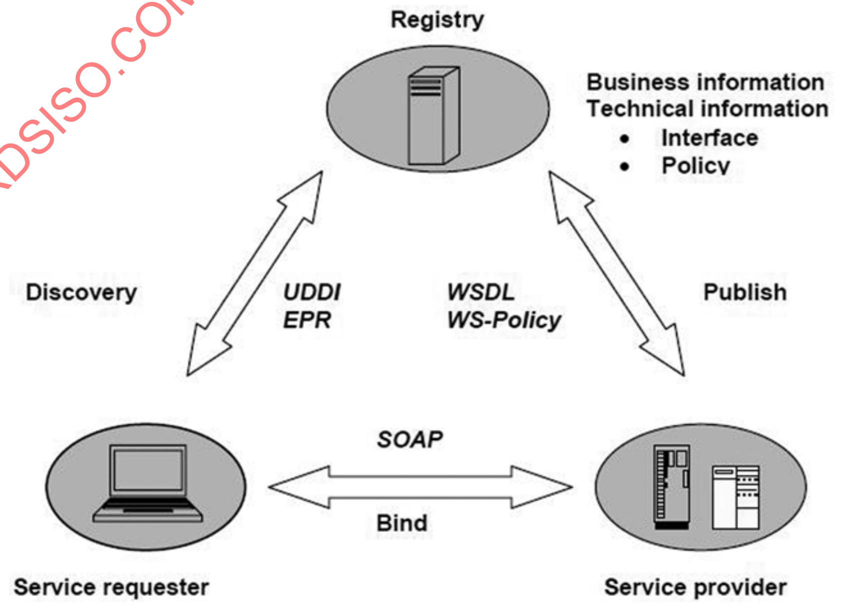

Figure 1 — Web service entities and their relationships

2 Conformance

There are no explicit conformance tests in this part of ISO 24097. Conformance is achieved by conforming to the requirements of ISO 24097-1. Specific conformance tests can be specified in another part of ISO 24097.

3 Normative references

The following referenced documents are indispensable for the application 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, *Transport information and control systems — Requirements for an ITS/TICS central Data Registry and ITS/TICS Data Dictionarie*s

4 Terms, definitions and abbreviated terms

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

NOTE 1 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.

NOTE 2 For general W3C references, see the Bibliography.

4.1 Terms and definitions

4.1.1
composability
facility enabling web services to add new features incrementally

4.1.2
domain
functional area in a policy assertion (e.g. security, reliability, transaction and messaging optimization)

4.1.3
ITS WS
web service that is designed specifically to support ITS services via the Internet

4.1.4
International Standard web service
web service conformant to this part of ISO 24097

4.1.5
platform
hardware, operating system, middleware and application development language, which provide a system environment

4.1.6
policy assertion
element of service metadata which identifies a domain (such as messaging, security, reliability and transaction) specific behaviour

4.1.7
skeleton
elements of service side code used for receiving remote method calls, invoking them and returning the result to the sender

4.1.8
stub
client code required to talk to a remote service

4.1.9
WS metadata
service metadata
metadata
high-level service description of a web service that controls provision of that service

4.2 Abbreviated terms

4.2.1
BNF
Backus Naur Form

4.2.2
BP
basic profile (of web services interoperability organization)

4.2.3
BPEL
business process execution language

4.2.4
DD
data dictionary

4.2.5
DR
data registry

4.2.6
EPR
endpoint reference

4.2.7
HTTP
hypertext transfer protocol

4.2.8
HTTPS
hypertext transfer protocol security

4.2.9
IRI
internationalized resource identifier

4.2.10
MIME
multipurpose Internet mail extension

4.2.11
MOF
meta object facility

4.2.12
MTOM
⟨SOAP⟩ message transmission optimization mechanism

4.2.13
OID
object identifier

4.2.14
OMG
object management group

4.2.15
OSI
open system interconnection

4.2.16
QoS
quality of service

4.2.17
REC
recommendation

4.2.18
RM
reliable messaging

4.2.19
RMI/IIOP
remote method invocation/Internet inter-ORB protocol

4.2.20
RPC
remote procedure call

4.2.21
SMTP
simple mail transfer protocol

4.2.22
SOA
service-oriented architecture

4.2.23
TCP/IP
transmission control protocol/internet protocol

4.2.24
tModel
technical model

4.2.25
UDDI
universal description, discovery and integration

4.2.26
URI
uniform resource identifier

4.2.27
UTF-8(/16)
8(/16)-bit universal character set transformation format

4.2.28
W3C
World Wide Web Consortium

4.2.29
WS
web service

4.2.30
WS-I
web services interoperability (organization)

4.2.31
WSDL
web services description language

4.2.32
XML
eXtensible markup language

4.2.33
XSD
XML schema definition

5 Notation

5.1 Prefixes and namespace URI used in core specification

This part of ISO 24097 uses predefined namespace prefixes throughout as given in Table 1. Other prefixes and namespaces (e.g. “Web Services Policy” and “Web Services Addressing”) are given in this part of ISO 24097.

NOTE 1 The choice of any namespace prefix is arbitrary and not semantically significant (see [Namespaces in XML]). However, the prefix is 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 and 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” represent 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:

“?” (0 or 1);

“*” (0 or more);

“+” (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 SHALL 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 part of ISO 24097, the combination is represented by a (service provider and 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

Braces enclose a set: "{" "}".

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, however it shall be expressed in order 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 "[Rnnnn]".

6 Requirements

6.1 Basic concept of web services standardization

6.1.1 Web services architecture

Given that WS require a number of functionalities, an architectural context is therefore essential. Web service standardization organizations construct standards within the framework of an SOA. An SOA is an evolutionary form of distributed computing and object-orientation).

The fundamental SOA philosophy (architecture) is:

— systems shall be coupled loosely by message;

— systems shall be linked dynamically;

— systems shall be composable by functional stacks.

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

a) Service composition stack: the stack that describes coordination of business processes. This stack is used to automate real business.

b) Service description stack: the stack that describes service interface and related service policy. This stack is used for metadata description.

c) Quality of service (QoS) stack: the stack that ensures message quality, security and transaction quality.

d) Messaging stack: the stack that describes message behaviour.

e) Transport stack: the stack that transports message.

f) Service publication and discovery stack: the stack that publishes a web service and its discovery.

WS are constructed on SOA-based open body standards (see Figure 2). 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.

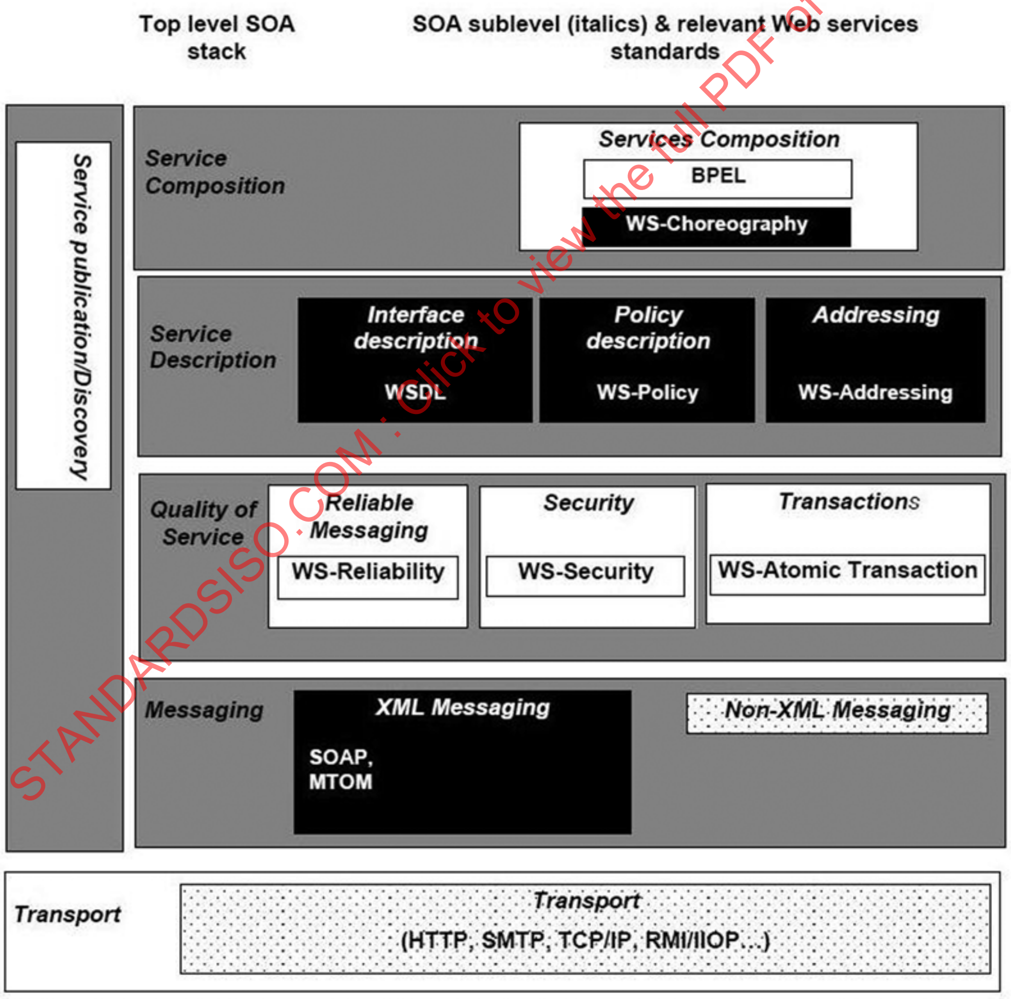

Figure 2 — SOA and its construct standards

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

NOTE 2 Some architects depict QoS layer as an upper layer of Service description layer. Other architects depict the reverse. This part of ISO 24097 describes the Service description layer as upper layer of the QoS layer. The reason for this is that the Service description layer uses QoS layer and it controls behaviour of QoS.

6.1.2 International standard web service standardization

Figure 3 depicts a MOF-like view of WS. 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 schema usage standard for the ITS sector.

M2 Layer (Web service standards, WS-I BP and this part of ISO 24097) provide rules and guidance for web service development.

M1 Layer (ITS Web service standards) provided rules and guidance for web service development particular to ITS. As long as M1 Layer instances of specific web service (ITS web service) follow this part of ISO 24097, basic interoperability is achieved.

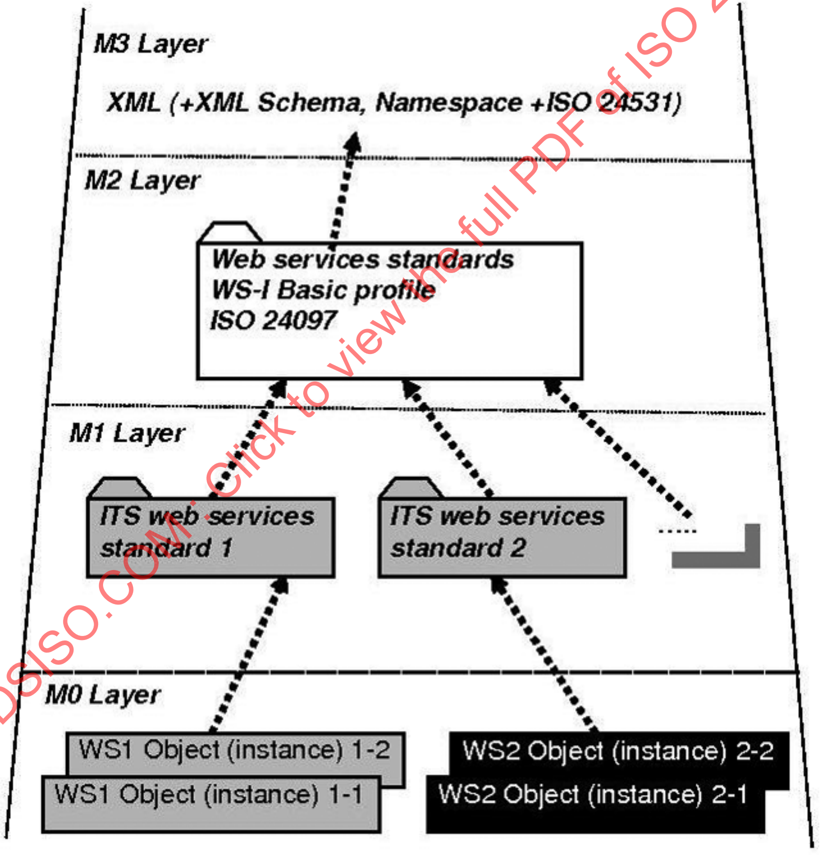
M3 Layer
XML (+XML Schema, Namespace +ISO 24531)
M2 Layer
Web services standards
WS-I Basic profile
ISO 24097
M1 Layer
ITS web services standard 1
ITS web services standard 2
M0 Layer
WS1 Object (instance) 1-2
WS1 Object (instance) 1-1
WS2 Object (instance) 2-2
WS2 Object (instance) 2-1

Figure 3 — ITS web service standard structure

6.2 Web service metadata

Web service standards are based on an SOA. This means that WS are constructed by the 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 part of ISO 24097 covers only single web service application, this layer description is not included in this part of ISO 24097.

The second upper-most 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 WS capabilities and constraints. The WS-Addressing standard presents the basis of metadata EPR and some message addressing functionality. Using these standards, ITS WS can respond to real world requirements. WS-Policy and WS-Addressing can be used independent of the particular WSDL version being used.

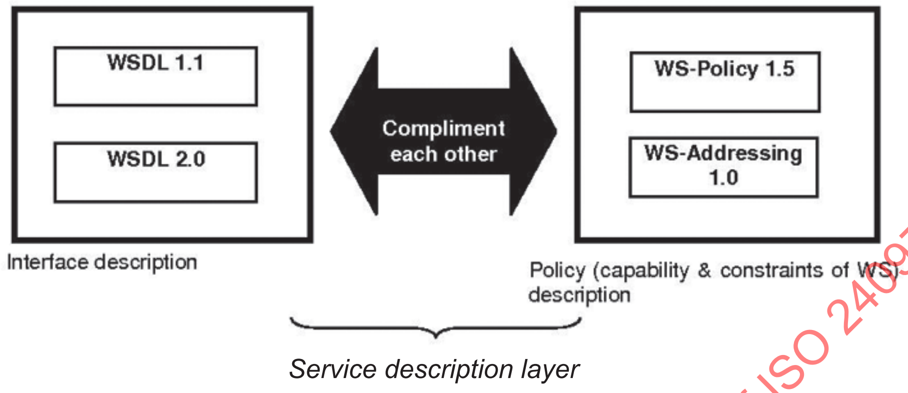

Figure 4 — WSDL, WS-Policy and WS-Addressing relationship

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

6.2.1 Common requirement and recommendation for metadata

6.2.1.1 International Standard web service metadata description

International Standard web service metadata shall be declared in formalized format.

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

6.2.1.2 Use of utf-8 or utf-16

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

NOTE Using UTF-8 or UTF-16 is a standard for internationalization for the XML application.

6.2.1.3 Version up scenario

If changing metadata, support shall be provided for all versions.

REASON There can be multiple scenarios when the web service version is upgraded (see Figure 5). If service is for an anonymous user and/or requires a short recovery time from any trouble with the new version of the software, support from the old version is required. A Type 2 scenario can respond to this requirement (see Figure 5).

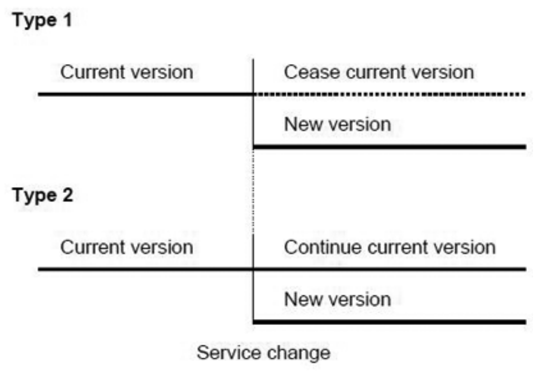

Figure 5 — Version up scenario

6.2.1.4 Interface or policy change

When changing interface or policy, cross impact shall be analysed and proper cross change shall be carried out.

REASON WS work only between a consistent interface and a policy pair. For example if a new binding is added in interface description, it can impact on WS-Policy. The reverse is also true. Therefore, a cross check shall be made. 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 shall become a new service and the old service shall remain.

NOTE There are many ways to manage physical files to correspond to a change of WSDL or WS-Policy. One method is changing in the same file (Type 1 of Figure 6); alternatively creating a new physical file (Type 2). Type 1 minimizes change, but the method selected is left to the discretion of the service provider.

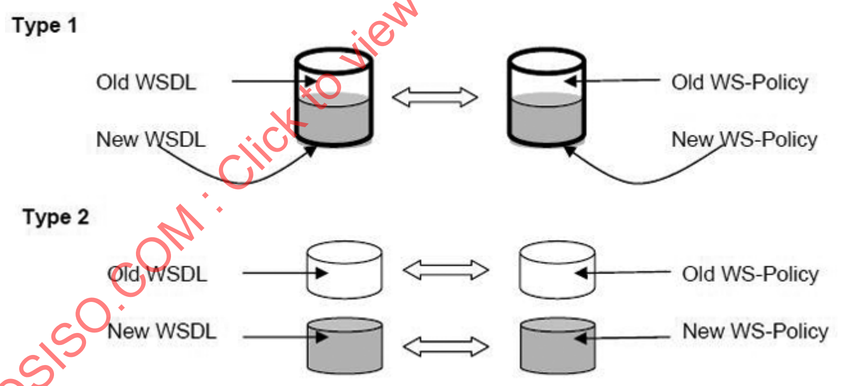

Figure 6 — WSDL WS-Policy change

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 helps to easily distinguish a change of metadata. Independent version numbering for both WSDL and WS-Policy is necessary because each can change independently.

NOTE For the WSDL versioning rule, see 7.2.4. For WS-Policy versioning rule, see 7.2.4.

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 sublayers: Interface description, Policy description and Addressing and shall explicitly identify the standards for: WSDL, WS-Policy and WS-Addressing. Clauses 7 to 10 describe the achievement of interoperable ITS WS using these standards.

7.2 Service description layer — Requirements and recommendations 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 role of WSDL in the ITS field includes:

g) the description of technical conditions agreed between service provider and service consumer;

h) the provision of automatic generation of stub and skeleton using web service developing tools; this reduces the software development load for both.

7.2.2 Multiple WSDL specifications

The current version of WSDL is WSDL 2.0 and this part of ISO 24097 focuses on 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 part of ISO 24097) of WSDL and SOAP are WSDL 2.0 and SOAP 1.2, respectively.

WSDL 1.1 will therefore 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 shall select the version of the specification.

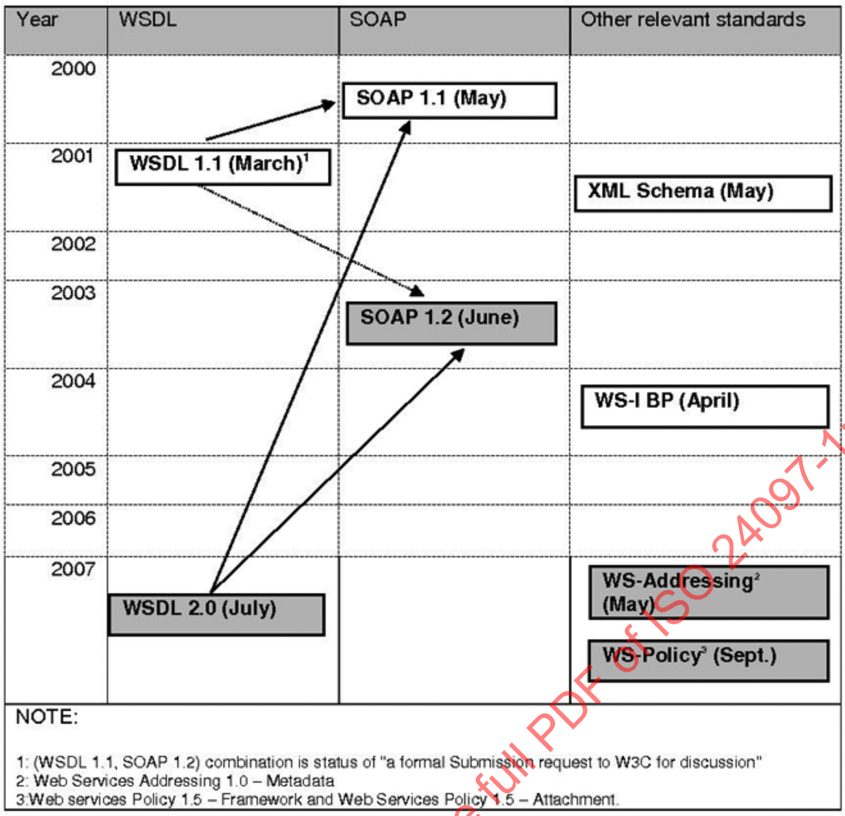
Year
WSDL
SOAP
Other relevant standards
2000
2001
2002
2003
2004
2005
2006
2007
SOAP 1.1 (May)
WSDL 1.1 (March)[1]
XML Schema (May)
SOAP 1.2 (June)
WS-I BP (April)
WSDL 2.0 (July)
WS-Addressing[2] (May)
WS-Policy[3] (Sept.)
NOTE:
1: (WSDL 1.1, SOAP 1.2) combination is status of "a formal Submission request to W3C for discussion"
2: Web Services Addressing 1.0 – Metadata
3:Web services Policy 1.5 – Framework and Web Services Policy 1.5 – Attachment.

Figure 7 — WSDL and SOAP version correct combination

Even if WSDL 2.0 is significantly superior to WSDL 1.1 and WSDL 1.1 users migrate to WSDL 2.0, the phase-out of WSDL 1.1 takes considerable time (see Figure 7), especially because WS are usually effected between different organizations. Furthermore, tools supporting the implementation of WSDL 2.0 continue to be developed beyond mid-2008.

With this uncertain condition, this part of ISO 24097, while focusing on WSDL 2.0, also accommodates version 1 specification (WSDL 1.1 and SOAP 1.1) and version 2 specification (WSDL 2.0 and SOAP 1.2).

For better readability of this part of ISO 24097, WSDL 1.1 aspects are presented in Annex A and WSDL syntax is given in Annex B.

Further information on WSDL 1.1 basics and the evolution from WSDL 1.1 to WSDL 2.0 is provided in Annex A.

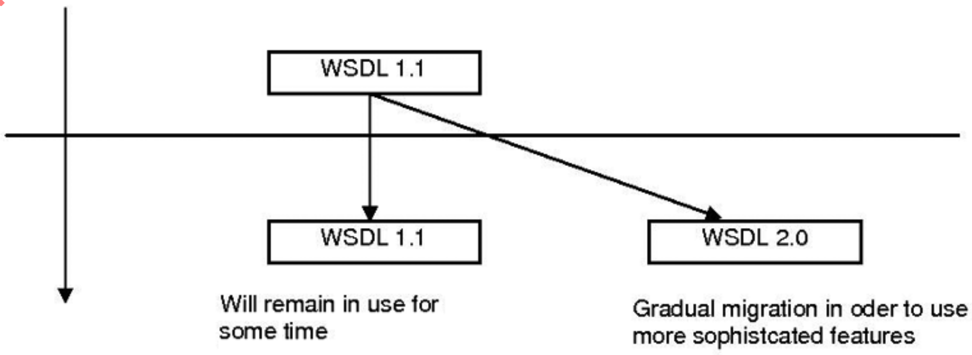
WSDL 1.1
WSDL 1.1
WSDL 2.0
Will remain in use for some time
Gradual migration in oder to use more sophistcated features

Figure 8 — WSDL migration

7.2.3 The relationship between WSDL and SOAP

SOAP is a messaging protocol. Both WSDL 1.1 and WSDL 2.0 can select SOAP protocol (as a syntax both can select other protocols, such as 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 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 versioning rule

a) The ITS web service interface version takes the following form:

   Form: m.n.a

   — m: major version number (`xs:positiveInteger`)

   — n: minor version number (`xs:nonNegativeInteger`)

   — a: draft version number (`xs:NCName`)

b) Version change takes the following form:

   — m: if m version instance cause m+1 version service fault

   — n: if (n version instance is correct to (n+1) version service) & (some (n+1) version instance cause m version service fault)

   — a: draft and semantic is changed

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

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

7.2.5 Requirements and recommendations for applying WSDL 2.0

The requirement for applying WSDL 2.0 is determined in this subclause.

7.2.5.1 WSDL 2.0

WSDL 2.0 is superior to WSDL 1.1 at these points:

a) modular structure for easy WSDL development and reuse;

b) object-oriented style (e.g. inheritance) support;

c) clear definition of extension point (e.g. Web Service Policy is described using this extension point); this enables the enriching functionality of WS.

NOTE At the time of publication of this part of ISO 24097, tool support for WSDL 2.0 is underway (e.g. Apache Axis 2). By comparison, the older WSDL 1.1 has rich tool support. Considering this fact, the most appropriate WSDL version is selected by taking the application requirement for tool support into account.

7.2.5.2 WSDL 2.0 conformance test

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

REASON WSDL 2.0 validation to the WSDL web site of W3C is the minimum condition for interoperable WS.

NOTE It supports online checking.

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 it to WSDL.

REASON Separating service description layer from messaging layer enables modular style development. Maintenance or evolution of systems becomes easy. Most fault messages are common to various applications, therefore reusable. It helps rapid development of ITS WS. Many countries have already completed message standardization. This also supports interoperability and WS rapid development.

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

Where possible, service classification based on ISO 14813-1 should be used in WSDL 2.0 name attribute (strongly recommended).

NOTE Its meaning is clearly understandable.

7.3 Service description layer — Requirements and recommendations for policy description sublayer

WSDL supports formal description of an interface. As previously described, WS-Policy complements other facets of WS facility and requirements (see Figure 4). Describing security, reliable messaging and/or transaction in formal method are indispensable for real world WS. Formal description also supports tooling. The basics of WS-Policy and requirements and recommendations for ITS WS are described in this subclause.

Table 2 gives 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

The basics of WS-Policy (role and syntax) and ITS WS requirements and recommendations are described in this subclause.

7.3.1.1 WS-Policy — Framework

As previously described, WS requirements and constraints relate to topics such as security, reliability, transaction and messaging. Treating all the topics in one document helps understanding and simplifies software tool development. How, then, are these topics (security, reliability, etc.) organized in a single policy document? The W3C standard, “Web services policy 1.5 - Framework” provides a mechanism for consistently assembling policy topics in a document (see Figure 9).

NOTE Web Services-Policy 1.5 WS-Policy were previously developed by vendor groups. The final specification version was version 1.4. It moved away to W3C and its version is now 1.5.

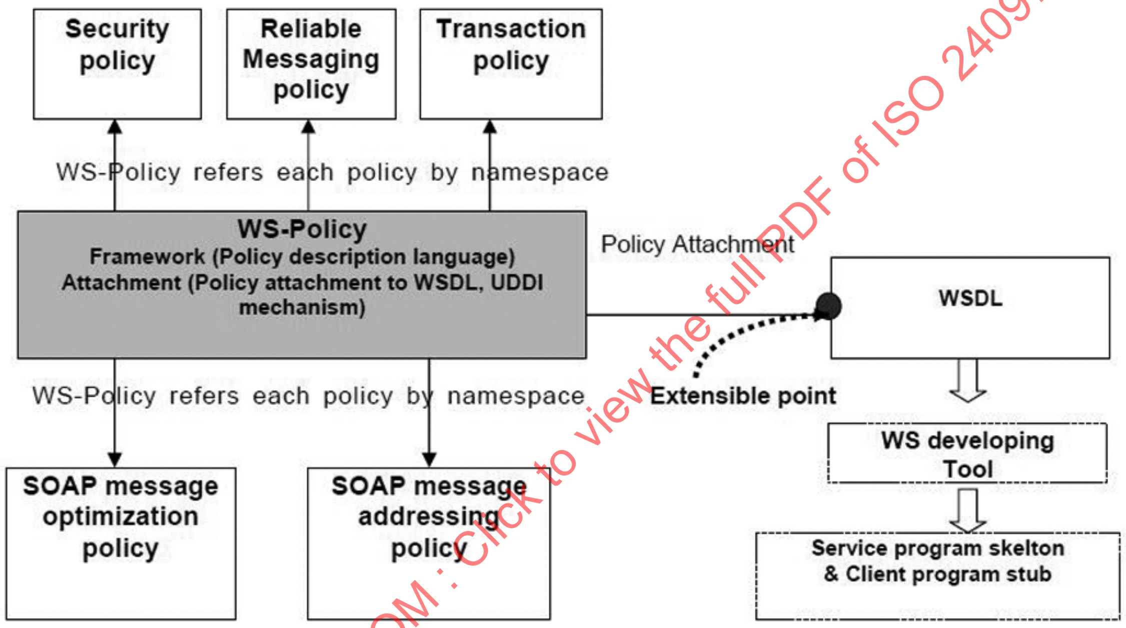
Security policy
Reliable Messaging policy
Transaction policy
WS-Policy refers each policy by namespace
WS-Policy
Framework (Policy description language)
Attachment (Policy attachment to WSDL, UDDI mechanism)
Policy Attachment
WSDL
Extensible point
WS developing Tool
Service program skelton & Client program stub
WS-Policy refers each policy by namespace
SOAP message optimization policy
SOAP message addressing policy

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, namely do something to domain specific topics like security, reliability, transaction and messaging. Attributes give some property to a policy.

The following is 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 its functionality is defined in WS-Policy. Other prefixed elements and attributes mean these constructs belong to other policy-related vocabularies. The `mtom:OptimizedMimeSerialization` element is a direction to apply SOAP message optimization. The "sp" prefix indicates the WS-Security vocabulary. Each prefixed construct functionality is defined in its policy-related standards.

WS-Policy prepares the selection operator. In this 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 shall support both alternatives.

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

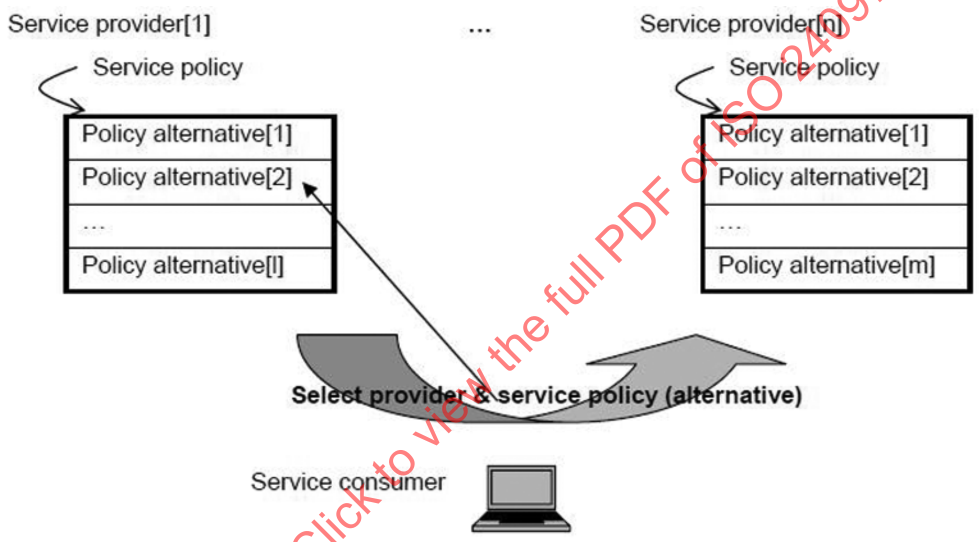

Figure 10 — Service provider and policy alternative selection

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 the extensible point, policy can be attached to a WSDL document and UDDI by the service provider. "Attachment" defines semantics as well as attachment syntax. Figure 11 depicts a policy attachment mechanism.

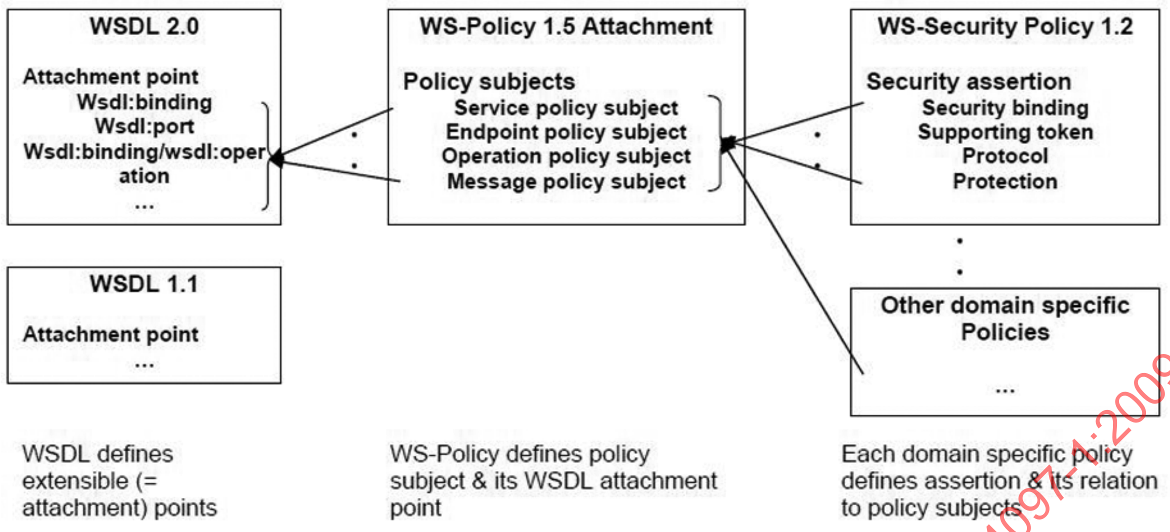

Figure 11 — WSDL, WS-Policy and domain specific policy

7.3.2 Requirements and recommendations for policy description

The WS-Policy requirements and recommendations for ITS WS are described in this subclause.

7.3.2.1 Explicit policy declaration of all applied domain specific policy

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

REASON WS-Policy has a rule that if a web service does not declare policy explicitly, the service consumer should not suppose anything about non-declared domain. If a service program uses non-declared domain, it might cause incompatibility. Therefore, the policy employed shall be explicitly declared.

NOTE Usually it is the service provider who is supposed to develop policy, such as the following process:

a) analyse and list the necessary capability and restriction;

b) decide each policy item as mandatory, optional or ignorable;

c) describe policy based on WS-Policy;

d) attach policy to the WSDL;

e) develop policy.

Comprehensive policy domain is covered and explicitly declared in steps a) to d).

7.3.2.2 WSDL and policy description separation

WSDL and policy description separation is strongly recommended.

REASON One reason is separation increases readability. If WSDL and WS-Policy description are not separated, they will become complex and difficult to understand. Separation increases readability. Another reason is to increase maintainability.

7.3.2.3 Policy change

If changing current WS 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 one, the new assertion shall be added with `wsp:Optional="true"`. Otherwise, if adding new policy assertion has exclusive relation to the current one:

— the new and current assertion shall be enclosed with `wsp:ExactlyOne`, and

— the new assertion shall be added as a child element of `wsp:ExactlyOne` tag.

7.3.2.3.2 Changing the current assertion to the new one

Fundamentally, changing the current assertion to the new one is not allowed because of the successive support requirement for current service consumer. However, if a minor improvement or correction is required, the new assertion rule shall be applied.

REASON The rule is necessary to ensure successive service support.

NOTE If policy change is achieved with physically the same 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, even where policy declaration is non-standardized, that the policy declaration be made explicitly.

REASON To avoid conflict with the user. Logging can endanger privacy, for example. In this case, the service provider can declare taking a log.

NOTE Taking a log is not standardized as a policy. `wsp:Ignorable` is predefined vocabulary of `wsp:Policy`. In such a case, using 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 the following example). "http://www.example.com/***tentative***" can include 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 Versioning of policy

a) The following versioning rule is strongly recommended.

Form: m.n.a

— m: major version number (`xs:positiveInteger`)

— n: minor version number (`xs:nonNegativeInteger`)

— a: draft version number (`xs:NCName`)

b) A version change takes the following form:

— m: if m version instance cause m+1 version service fault

— n: if (n version instance is correct to (n+1) version service) & (some (n+1) version instance cause n version service fault)

— a: draft and semantic is changed

REASON Versioning clarifies a change of policy. This rule implies changing semantics.

NOTE WS-Policy semantics has no explicit versioning attribute. The syntax of `wsp:Policy` element is `<wsp:Policy …>`. Character "…" means policy extension point. Therefore, the version can be added as in 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 — Requirements and recommendations for addressing sublayer

WS-Addressing for ITS applications shall provide transport-neutral mechanisms to address ITS web service messages (see a future part 2 of ISO 24097 for examples).

8 Quality of service layer

8.1 Quality of service layer — Requirements and recommendations for reliable messaging sublayer

8.1.1 Requirements and recommendations for reliable messaging policy description

A facility to control message reliability shall be identified. RM (reliable messaging) 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 (policy) shall be described with various domain-specific policies (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 gives 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 shows how `ExactlyOne`, `AtLeastOne`, `AtMostOne` and `InOrder` affect messaging reliability. Both designations of `ExactlyOne` and `InOrder` keep no loss, duplication or order. The choice of appropriate assertion depends on an application requirement.

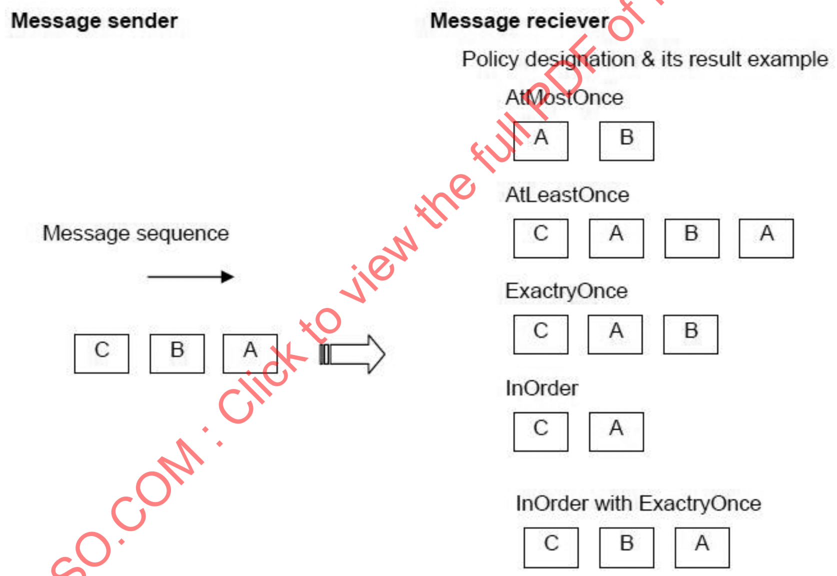

Figure 12 — WS-RM designation and its result example

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

8.2 Quality of service layer — Requirements and recommendations for security sublayer

WS are machine-machine service deliveries on insecure Internet (if security is not applied). Securing web service is acheived by applying *WS-SecurityPolicy*. *WS-SecurityPolicy* relates to many standards. Table 4 gives the relevant standards.

Table 4 — Security-related standards

Standard title	Function	Standardization 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)	Metadata specification of WS 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 called *WS-Security Policy* on 1 July 2007. Using this standard, the service provider and the service consumer can communicate safely in a compatible manner. As the name implies, the standard works under general control of WS-Policy. As a result, security requirement and constraint can cooperate with other requirements and constraints, such as messaging reliability and/or SOAP MTOM. MTOM can be used to transmit binary security token effectively.

Security policy assertion is attached to WSDL. The kinds of security policy assertion that can be attached to certain WSDL attachment points are specified in *WS-Security Policy*. For attaching security assertion, security assertion shall be described as a part of WS-Policy. The application of security policy is not simple (see a future part 2 of ISO 24097 for an example of how to apply security policy).

8.3 Quality of service layer — Requirements and recommendations for transaction sublayer

This sublayer relates strongly to the service composition layer. Therefore, this subject will form the subject of a future document.

9 Messaging layer

9.1 Messaging layer — Requirements and recommendations for XML messaging

9.1.1 The role of SOAP

SOAP defines the protocol between service provider and service consumer programs (including communication to UDDI as WS.

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 actual services requirement and the 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 machine-machine communication, thus different from web applications. In the case of a web application, if fault happens, the web browser operator is notified and he/she recovers the fault manually. On the contrary, in a web service the fault is processed by the client program (machine-machine communication). As a result, it shall be necessary to specify the kind of fault and the structure through which it is notified. Therefore, fault standardization is critical to interoperable service.

9.1.3 The relationship of SOAP 1.2 to WSDL 1.2

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

Table 5 specifies the description of SOAP behaviour using 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 policy

SOAP messages often become large. MTOM supports SOAP message transmission optimization (compact form of serialization). 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 WS can apply this feature.

NOTE 1 MTOM Serialization Policy Assertion 1.1 is a “Last Call Working Draft”.

NOTE 2 The namespace URI of the specification is http://www.w3.org/2007/08/soap12-mtom-policy. The conventional prefix and element are `wsoma:MTOM`. The syntax of MTOM is <`wsoma:MTOM wsp:Optional?` …/>.

10 Service publication/discovery layer

10.1 Service publication/discovery layer — Requirements and recommendations for universal description, discovery and integration

10.1.1 The role of the UDDI

The UDDI is used to discover the services needed by the users and provides technical conditions for the use of services (see Figure 1). The UDDI is a kind of directory service in various usage scenarios, for example: a public worldwide directory service in which there is no limitation of access, such as B2C (business to consumer), or private or semi-private forms (access is limited to inside an enterprise or consortium). In WS, service providers assume that the users will develop programs for the use of WSDL. Therefore, data concerning technical information for the use of services [(IRIs) of WSDL with policy and explanatory documents] shall be provided, either through the UDDI or the EPR of WS (see Clause 7). The EPR includes only metadata, therefore business information is excluded. This part of ISO 24097 is for the impromptu development of client programs.

10.1.2 The components of the UDDI

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 use of service;

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

Item a) is a white-page equivalent, while items b) and c) concern business in a broad sense (yellow-page equivalent). The information in items d) and e) is technical information for the use of the service (green-page equivalent).

The UDDI is a type of web service and accessing the 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

The Public UDDI is a registry on which any subscriber can publish his/her service and to which authorized users can have access without 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 can be replaced by governmental, standardization organization or consortium-based web pages. For ITS-related organizations, neither white-page nor yellow-page information is likely to be necessary. It is highly possible for ISO/TC 204 working groups and regional or national ITS-relevant organizations (e.g. on web pages) to publish WS.

10.1.3.1 The UDDI and ISO 14817 DD/DR

The most important feature of the Service publication and discovery stack is the use of interface information (WSDL) and the reuse of data concepts. Either an ITS data registry (ISO 14817-based) or a UDDI shall be used. Table 6 compares the two registries.

At an abstract level, ISO 14817 data registry and UDDI have almost the same functionality. In this part of ISO 24097, use of the UDDI is not considered an essential condition. However, it is possible to use the UDDI for the WS directory.

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 on ISO level Some country services and some country trials	Deployed
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 methods. 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 is possible)	Possible
Explanation of data concept		Explained by description metadata	Possible
Acquisition of business information		Not the intention of the DR	Possible
Retrieval of service		Not the 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 specifies auto notification or notification at access
Retrieve		Browser	Program/browser(optional)

Reasons for giving preference to the use of an ISO 14817 data registry are:

a) some countries already have ISO 14817-based registries;

b) in general, a database is only effective if it includes almost all required information (see Figure 13).

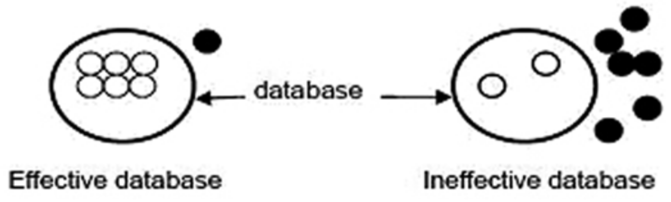

Figure 13 — Effective database

NOTE Nonetheless, this part of ISO 24097 does not restrict usage of UDDI. It is a matter of choice by ITS stakeholders.

If an ITS stakeholder uses ISO 14817 DD/DR, the conformant conditions to this part of ISO 24097 are described in 10.1.4.

10.1.4 Requirements and recommendations 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 to an ISO 14817-compliant data registry or UDDI or 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 For the purpose of reuse of the message.

10.1.4.3 Registration of fault

Fault shall be registered in an ISO 14817-based registry.

REASON For the purpose of reuse of the message.

10.1.4.4 Object identifier

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

REASON Message and fault are described by XML Schema. ISO 24531 defines how to give the OID for Schema and its constructs. Therefore, following the upper rule, all constructs can maintain consistency.

Annex A
(normative)

Principles and evolution of WSDL from Version 1.1 to 2.0

A.1 General

This annex provides information about the basics of WSDL and its evolution from WSDL 1.1 to WSDL 2.0. The provisions in this annex are normative where WSDL 1.1 is used.

A.2 WS-I basic profile

First-step WS standards (SOAP 1.1, WSDL 1.1 and UDDI v.x) contain some ambiguity and conflict. One reason for this is that WSDL 1.1 and SOAP 1.1 were developed before the XML Schema recommendation. XML Schema is a base component of the WS standard. In order to promote WS interoperability across vendors' software, WS-I, vendors and end-user organizations jointly developed basic profile (BP). The WS-I BP provides the requirements to which WSDL and SOAP shall comply.

Figure A.1 shows the relationships among the specification of standardization bodies, WS-I BP and this part of ISO 24097.

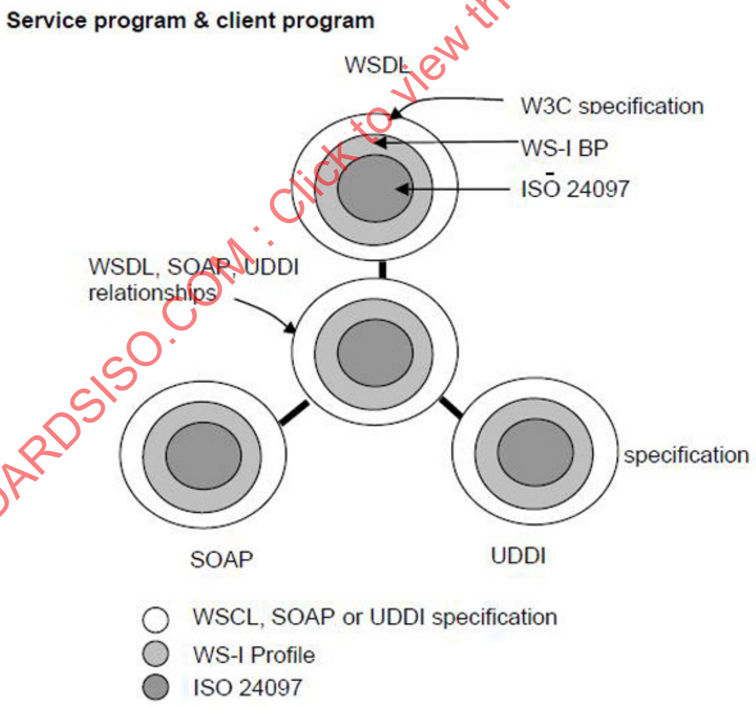
Service program & client program
WSDL
W3C specification
WS-I BP
ISO 24097
WSDL, SOAP, UDDI relationships
specification
SOAP
UDDI
WSCL, SOAP or UDDI specification
WS-I Profile
ISO 24097

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

The relationship is (W3C specification) > WS-I BP > ISO 24097. This means that WS-I BP constrains this part of ISO 24097, which follows WS-I BP.

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

A.3 WSDL 1.1

The WSDL 1.1 basics and requirements for WSDL 1.1 in ITS service provision are described in this clause. 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 SHALL be available to an authorized consumer upon request.”

A.3.1 WSDL 1.1 structure

Figure A.2 shows the structure of WSDL 1.1. As seen in Figure A.2, WSDL is a collection of building blocks, which are “types”, “message”, “portType”, “binding” and “service”.

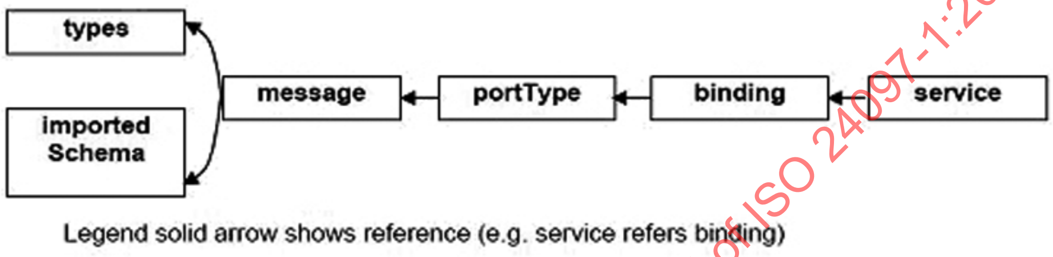
types
imported Schema
message
portType
binding
service
Legend solid arrow shows reference (e.g. service refers binding)

Figure A.2 — The structure of WSDL 1.1

All the building blocks have a name attribute which is referred to by another building block (except imported schema. In this case, namespace plays the same role as name.

“Building blocks” are constructed by abstract parts (types, message and portType) and concrete parts (binding and service).

A named “Building block” enables flexibility, reusability and extensibility for WS, it provides the following:

— flexibility: one can select a best method from multiple methods (e.g. one can select SOAP/HTTP binding or HTTP binding);

— reusability: one can reuse building blocks;

— 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.4 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 element 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
                            without error
                        </xs:documentation>
                    </xs:annotation>
                </xs:any>
                <xs:element name="fault"
                    type="soapenv:FaultStructure">
                </xs:element>
            </xs:choice>
        </xs:complexType>
    </xs:element>

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


    <xs:complexType name="Faultstring">
        <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 meanings 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) shall decide soap `envelope:detail` structure and its contents.	If body element fault happens, mandatory

The meanings 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 understood or not obeyed	Mainly client side
Client	The message (body) was incorrect 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.4.1 Fault code

As a `soapenv:faultcode` contents limitation of only "`soapenv:VersionMismatch`", "`soapenv:MustUnderstand`", "`soapenv:Client`" and "`soapenv:Server`" is strongly recommended.

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

NOTE 1 SOAP 1.1 note does not give SOAP schema explicitly. WS-I defines the schema to clarify this structure; the schema location is http://schemas.xmlsoap.org/soap/envelope/ . This part of ISO 24097 refers 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>
            Envelope element and Header element are ommitted
        </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:FaultStructure">
                </xs:element>
            </xs:choice>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="FaultStructure">
        <xs:sequence>
            <xs:element name="faultcode" type="soapenv:faultcodeValue" />
            <xs:element name="faultstring"
                type="soapenv:Faultstring" />
            <xs:element name="faultactor" type="xs:anyURI" />
            <xs:element name="detail"
                type="soapenv:detail" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Faultstring">
        <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.4.1.1 Simplified ITS fault

The following fault messages are recommended.

Table A.3 — Recommended fault message

soapenv: faultcode	soapenv: faultstring	soapenv: fault actor	soapenv: detail	Note
soapenv:VersionMismatch	Your input SOAP version is mismatched 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 is incorrect	Null	Each message is decided and standardized by ISO/TC 204 working group or developer	

If a service provider detects a fault, the service shall be stopped immediately and the service consumer shall be informed. File processes (e.g. write and update) shall not be processed. The important thing before retrying is to give sufficient information about who (service requestor 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-machine communication.

EXAMPLE 1 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="faultstring"
                type="cf:Faultstring"
                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="faultstring"
                type="cf:Faultstring"
                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="faultstring"
                type="cf:Faultstring"
                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="Faultstring">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute ref="xml:lang" use="optional" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

</xs:schema>
```

EXAMPLE 2   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="ApFaultStructure">
        <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="Faultstring"
                type="af:Faultstring" fixed="InputDataError"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="Faultstring">
        <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 3 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:ApFaultStructure"/>

</xs:schema>

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