TECHNICAL REPORT

ISO/TR 24097-2

First edition
2015-10-15

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

Part 2: Elaboration of interoperable web services' interfaces

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

Reference number
ISO/TR 24097-2:2015(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 meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is 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*

— *Part 2: Elaboration of interoperable web services' interfaces* [Technical Report]

The following parts are under preparation:

— *Part 3: Quality of service* [Technical Report]

Introduction

ITS services have been evolving from single functional and limited area services, to services in which many systems co-operate to provide effective and efficient services 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 communication with/between traffic management, route guidance systems, security systems, environment protection systems, private freight management systems, and transport-related electronic payment service with banking or credit industry.)

These systems (even those limited to ITS services) are usually deployed in a heterogeneous circumstance, use different hardware, different operating systems (OS), middleware, or development (programing) 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 WS technology for ITS services can significantly simplify and reduce the cost of internet based service provision, which may well affect the level and speed of take up of use of ITS services.

WS require a lot of functionalities, and as a result, architecture is indispensable. WS 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 the ITS services, the following effects are expected.

From a business viewpoint:

— increased service value;

— internationalization;

— expansion to the business automation.

From a system development viewpoint:

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

— WS enables system developer focus on "WHAT" not "HOW". "HOW" is covered by standard-based tools. This enables quick and easy system software development;

— WS standards have a composable structure, and so promote reusability of software by SOA;

— Easy connection to a legacy system.

In the ITS sector, message standardization of many applications have already been completed, are well-advanced, or are determined regionally. Message standardization is intended to improve system coordination, interoperability, and re-use. 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 viewpoint of WS standards evolution, 2007 was an epoch-making year. Web Service Description Language (WSDL) 2.0 became a W3C recommendation. Corresponding with this, relevant WS specifications were standardized by open standard bodies (W3C and OASIS). These standards cover all functional layers. Using these standards, the ITS sector has a sound base for interoperable WS.

ITS service collaboration with other sectors is expected to increase mutual effectiveness. Globalization of economies also requires communication across the domains and jurisdictions. All these collaborations rely on the interoperability of services. Interoperability can be achieved if based on open international standards.

WS was created to use distributed network resources in an interoperable way. However, to realize interoperable WS various functionalities are required.

In most ITS services, availability and quick recovery from a fault is critical. Business process management and monitoring help to realize these requirements. Business Process Execution Language (BPEL) is considered one method. It is based on web services. BPEL also enables various services combination and automatic execution of business process. All these apply to web services.

This Technical Report presents a base of high quality ITS services with easy and quick development as well as a base for further service expansion.

"Using web services (machine-machine delivery) for ITS service delivery" has been developed considering these requirements. ISO 24097 consists of three parts: ISO 24097-1, ISO 24097-2, and ISO 24097-3. ISO 24097-1 is an International Standard. ISO 24097-2 and ISO 24097-3 are Technical Reports.

ISO 24097-1 focused on an approach to realize interoperable ITS WS. ISO 24097-2 and ISO 24097-3 are example based documents that show how to realize interoperable ITS web services that are already described in ISO 24097-1.

Fundamental concept

Metadata, as this term reminds us, may be considered a higher level description of requirements and constraints of a web service. Metadata are, by its nature, declarative. Declarative means one **does not care about how to realize requirement**, but only about **what functionality is wanted in a WS**.

ISO 24097-1 proposed to construct metadata description based on standards. Recap key points of ISO 24097-1 are as follows:

a) To construct interoperable WS, standards-based **metadata description** is mandatory. This also implies a technical contract between service provider and service consumers.

b) WS service description metadata consists of ***interface metadata*** and Quality of Services (***QoS) metadata***. Only by describing both metadata could WS be interoperable. In addition to interoperability, this metadata provides the following:

   1) easy development of WS starting from requirements and constraints (top down approach);

   2) quick and effective and high quality system delivery of service from metadata with support from a software tool called a generator (especially this feature is especially important for a service consumer);

   3) readily realize service evolution and maintenance throughout its lifecycle.

c) ***Interface*** metadata describes the interface between a service program and a service consumer program. Therefore, this metadata represents interface contract between service provider and consumer. This information is published by the service provider and evaluated by potential consumers. This metadata are expressed in Web Service Description Langage (WSDL).

d) ***QoS*** metadata are composite of domain specific requirements and constraints such as security, reliable messaging, message addressing, Simple Object Access Protocol (SOAP) message transmission optimization. ***QoS*** metadata are described using WS-Policy. WS-Policy is constructed by two standards; "WS-Policy 1.5 — Framework" and "WS-Policy 1.5 — Attachment".

Figure 1 depicts web service descriptive language metadata USE Case and its role in web services.

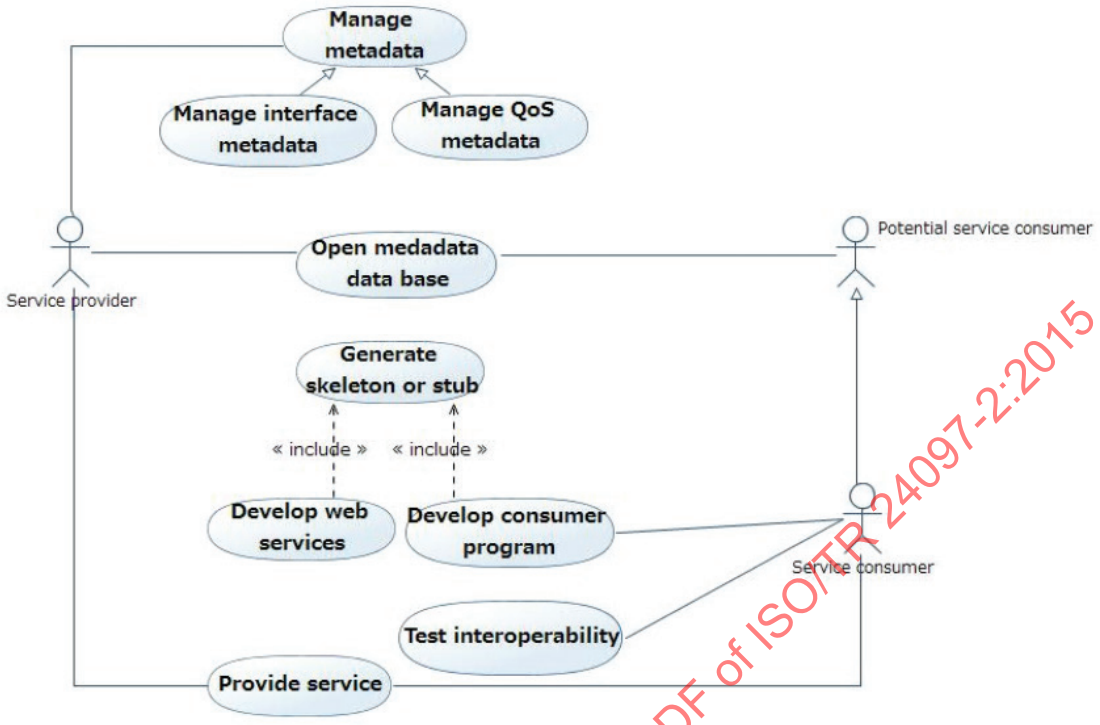


Figure 1 — High level ITS web service metadata role

Table 1 is a high-level use case description of ITS web service metadata's role.

Table 1 — Use case description of ITS WS metadata's role in high level

Use case description (Level 1)	
Name	ITS Web Services metadata role
Actors	Service provider Potential service consumer Service consumer
Description	This use case depicts a high level ITS WS metadata role through an ITS WS lifecycle.
Main scenario	1. Service provider creates standard-based interface and QoS metadata, and gives version number individually. (See Alt-1 for alternative scenario) 2. Service provider publishes metadata through Universal Description, Discovery, and Integration (UDDI), Service end point, web page, etc. 3. Service provider develops web service, including (Generating skeleton or stub). 4. Service provider provides the service. 5. Potential service consumer estimates the service from the view point of business value to him and/or interoperability realization difficulty. 6. If potential user decides to utilize the service, he develops client WS program, including the generation of a skeleton or stub. 7. Service consumer tests his program interoperability to the service program. 8. After successful interoperability test consumer utilizes the service.
Alternative scenario	Alt-1: In the case of the service provider service 1. He changes the metadata and the relevant version number. 2. Publishes revised metadata. 3. Resume.
Use case description (Level 2)	
Name	Generate skeleton or stub
Description	Service provider or service consumer creates skeleton or stub, respectively. Usually this process is done by using WS development tools. This enables stakeholder to quickly develop program and go sooner to interoperability test.
Main scenario	1. A tool reads interface metadata and QoS metadata. 2. It creates the interface part and run time data for required QoS.

e) "WS-Policy 1.5—Framework" acts as a structure for connecting domain specific metadata in one policy document. Each domain specific metadata has its own vocabulary and is identified by its namespace. Joining service side policy and service consumer policy, interoperability can then be tested. If join is not empty, then service provider program and service consumer program are interoperable.

f) "WS-Policy 1.5—Attachment" gives mechanism to attach policy to WSDL that policy is effective at runtime. Figure 2 exhibits WS-Policy's role to connect QoS to ***Interface metadata.***

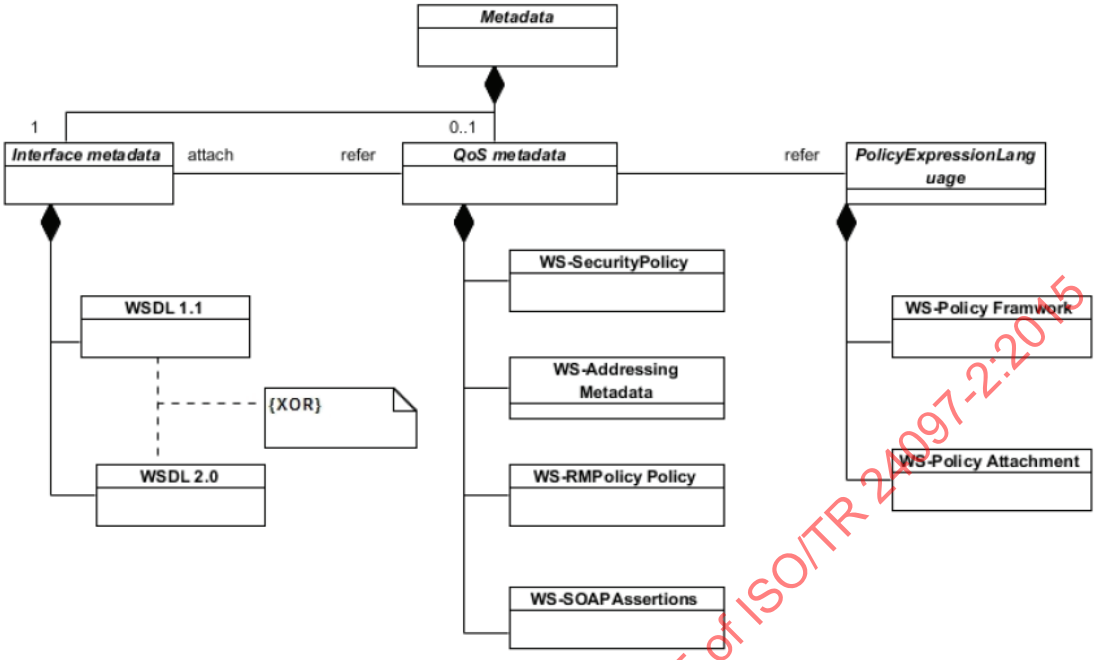


Figure 2 — Web service policy roles

g) For policy reuse, promotion of maintainability, and readability, policy description as an independent document from the interface document is recommended. Then combine both documents in a wsp:PolicyReference element in a WSDL.

h) When WS evolves, backward compatibility is recommended strategy.

NOTE WS-Policy related issues are to be elaborated in ISO 24097-3.

The second phase standards of web services metadata were delivered through 2007 to 2014 from standardization organizations like W3C and OASIS, and continue to evolve. There have been, to date, few documents that explain how to apply new metadata standards in a consistent and comprehensive manner. Without such assistance it is not easy to use relevant standards in a consistent manner.

In addition, making WS secure is essential, but realizing secure WS requires quite vast cryptography technologies in the background, such as basic XML signature and XML encryption. So applying these fundamentals could become a major hurdle to overcome.

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

Part 2:
Elaboration of interoperable web services' interfaces

1 Scope

This part of ISO/TR 24097 elaborates on ISO 24097-1 by discussing ***Interface metadata***.

This part of ISO/TR 24097 covers the following:

— interface metadata standard version selection (WSDL 1.1 or WSDL 2.0);

— SOAP version selection (SOAP 1.1 or SOAP 1.2);

— WSDL 1.1 SOAP 1.2 binding;

— WS-I conformant WS development.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 24097-1, *Intelligent transport systems — Using web services (machine-machine delivery) for ITS service delivery — Part 1: Realization of interoperable web services*

3 Terms and definitions

For the purposes of this document, the terms and definitions in ISO 24097-1 and the following apply.

3.1
claim
declaration made by an entity

Note 1 to entry: Entity is a name, identity, key, group, privilege, capability.

3.2
claim confirmation
process of verifying that a claim applies to an entity

3.3
domain
specific area to which a policy applies

EXAMPLE Security, message transmission reliability.

3.4
integrated development environment
software that provides comprehensive facilities for application development

3.5
instance document
XML document that conforms to a particular schema

Note 1 to entry: If schema is WSDL, XML document is WSDL instance document.

3.6
metadata
meta data
descriptive data of an instance WS behavior

Note 1 to entry: WS consists of *interface metadata* (WSDL description) and *QoS metadata*.

3.7
policy assertion
requirement, capability, or other property of a web service (WS-Policy)

3.8
policy subject
entity with which a policy assertion can be associated (WS-Policy)

EXAMPLE An endpoint, message, resource, operation.

4 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

BP	(WS-I) Basic Profile
BPEL	Business Process Execution Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
OASIS	Organization for the Advancement of Structured Information Standard
QoS	Quality of Services
rpc	remote procedure call
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol (SOAP 1.1)
URI	Uniform Resource Identifier
URL	Universal Resource Locator
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WS	Web Service
WS-I	Web Services Interoperability Organization
WSDL	Web Services Description Language
WTP	Web Tool Platform

XML eXtensible Markup Language

5 Notation and conventions

5.1 Namespace URI and prefixes used in this specification

This part of ISO/TR 24097 uses a number of namespace prefixes throughout; these are listed in Table 2.

The choice of any namespace prefix is arbitrary and not semantically significant. However, the prefix shall be unique in any single document. This part of ISO/TR 24097 uses a namespace prefix as shown in Table 2.

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

Table 2 — Namespace and prefix convention

Prefix	XML Namespace URI	Specifications
`s`	Either of `s11` or `s12`	
`s11`	http://schemas.xmlsoap.org/wsdl/soap/	SOAP 1.1
`s12`	http://schemas.xmlsoap.org/wsdl/soap12/	SOAP 1.2
`wsdl`	http://schemas.xmlsoap.org/wsdl/	[WSDL 1.1]
`wssa`	http://www.w3.org/2011/03/ws-sas	SOAP Version assertion policy
`xs`	http://www.w3.org/2001/XMLSchema	[XML-Schema1], [XML-Schema2]
`wsic`	http://ws-i.org/schemas/conformanceClaim	WS-I conformance claim
`bp20`	http://ws-i.org/profiles/basic-profile/2.0	BP 2.0
`bp12`	http://ws-i.org/profiles/basic-profile/1.2/"	BP 1.2
`tns`		The "this namespace" (tns) prefix is used as a convention to refer to the current web service.

Standard bodies, such as W3C and OASIS, make updated versions on a periodic basis. However, support tools follow with some time lag. So at the design stage it is necessary to check which International Standard was supported by which software tools.

5.2 Web service syntax notation: pseudo-schemas

Every web services language (e.g. WSDL, WS-Policy) has its own schema to validate a user's instance description. Because web services schema is complex, it is helpful to know the grammar at a glance, therefore pseudo-schemas (or shorthand schemas) are used to represent schema syntax. In this representation

— The syntax appears as an XML instance, but values 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 (...) 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 understand an extension, the receiver SHOULD ignore the extension.

5.3 XPath 1.0 expression

XPath 1.0 expression is used to specify an XML element and attribute.

EXAMPLE /wsdl:definition/wsdl:message, wsdl:definition/wsdl:binding/wsdl:@name

5.4 SOA stack name notation

SOA stack (layer) name is represented by bold italics.

EXAMPLE ***messaging***

5.5 The context of the terms WSDL, SOAP, and BP

These terms (WSDL, SOAP, and BP) are used if they do not distinguish the versions. For example, WSDL means both WSDL 1.1 and WSDL 2.0, which does not distinguish version. SOAP and BP are used same way.

5.6 Examples

To clarify explanation examples have been included using the IDE “Eclipse” (from Eclipse Foundation) and its plug-in web tool platform (hereafter WTP). This is used only as an example of an available tool and not a recommendation to use.

Eclipse has been selected for following reasons:

— open software (not specific vendor software);

— provision of integrated development environment (from design to deployment, based on latest software technology);

— selectable of the best plug-ins from many candidate plug-ins;

— has context based wizard;

— W3C members participated in the development of the Eclipse;

— reasonable quality of the document;

— eclipse influences many commercial IDE, so the users of other IDE should be able to understand it;

— all examples of this part of ISO/TR 24097 are informative.

5.7 The term {service, user}

In this part of ISO/TR 24097, the term {service, user} means {ITS service, ITS user} respectively.

6 Interoperable version selection of *interface description metadata*

There are many web services related standards and usually multiple versions for each standard. Further some standards are still evolving. This situation presents a significant challenge to potential users..

In the case of an ***interface description metadata*** standard, there are two standards; WSDL 1.1 and WSDL 2.0. Comparing these two standards, WSDL 1.1 is more proven in the market and has richer

software vendor support. Conversely, a software vendor that supports WSDL 2.0 is rare at the writing of this part of ISO/TR 24097. WSDL 1.1 is supported by almost all web services software vendors.

As a result, almost 100 % web services provider use WSDL 1.1. This also means almost 100 % web service consumers use WSDL 1.1.

Vendor software usually enables the formulation of an interface using ***interface description metadata.*** A service provider's business objective is usually quick market penetration and user acquisition. Interface program generation through use of a vendor tool brings benefits both service providers and consumers. Both can concentrate their efforts to business logic. As described in ISO/TR 24097-1, some ITS web services interact with other sector services. This responds to user seamless service request. For this reason, using well proven ***'interface metadata'*** is desirable. WS- I (explained later) targets only WSDL 1.1.

Therefore, the rest of this part of ISO/TR 24097 focuses on WSDL 1.1 usage scenario.

7 SOAP version selection

7.1 SOAP 1.1 or SOAP 1. 2?

SOAP is primarily accepted as a ***messaging*** standard. There are two SOAP versions (SOAP 1.1 and SOAP 1.2). SOAP 1.2 has two versions (First edition and Second edition).

W3C XML Protocol Working Group describes SOAP 1.2 as follows:

"*.... after SOAP 1.1 note published, the* XML Protocol Working Group *was chartered in September 2000 to design an XML-based protocol. Through 2.5 year-long work, the XML Protocol Working Group discovered and addressed around 400 issues in order to make the SOAP Version 1.2 and its processing model robust and unambiguous.*"

In the SOAP 1.2, two documents are added for further promote interoperability: One is "SOAP Version 1.2 Specification Assertions and Test Collection" mainly for software vendors, and the other is "SOAP Version 1.2 Part 0: Primer" mainly for a user.

Table 3 summarizes SOAP 1.1 and SOAP 1.2.

Table 3 — Comparison of SOAP 1.1 and SOAP 1.2

	SOAP 1.1	SOAP 1.2
Status	W3C note	W3C recommendation
Published	08 May 2000	First edition: 24 June 2003 Second edition: 27 April 2007
Standard(s)	Simple Object Access Protocol (SOAP) 1.1.	— SOAP Version 1.2 Part 1: Messaging Framework — SOAP Version 1.2 Part 2: Adjuncts
Tutorial document	—	SOAP Version 1.2 Part 0: Primer (W3C)
Test support	—	SOAP Version 1.2 'Specification Assertions and Test Collection'

Considering these facts, except for supporting existing legacy services, applying SOAP 1.2 for new services seems reasonable.

7.2 Creating a SOAP 1.2 web service

How can a SOAP 1.2 message be created? This creation focuses on how to designate SOAP 1.2 usage through WSDL 1.1. Figure 3 shows a WSDL 1.1 structure.

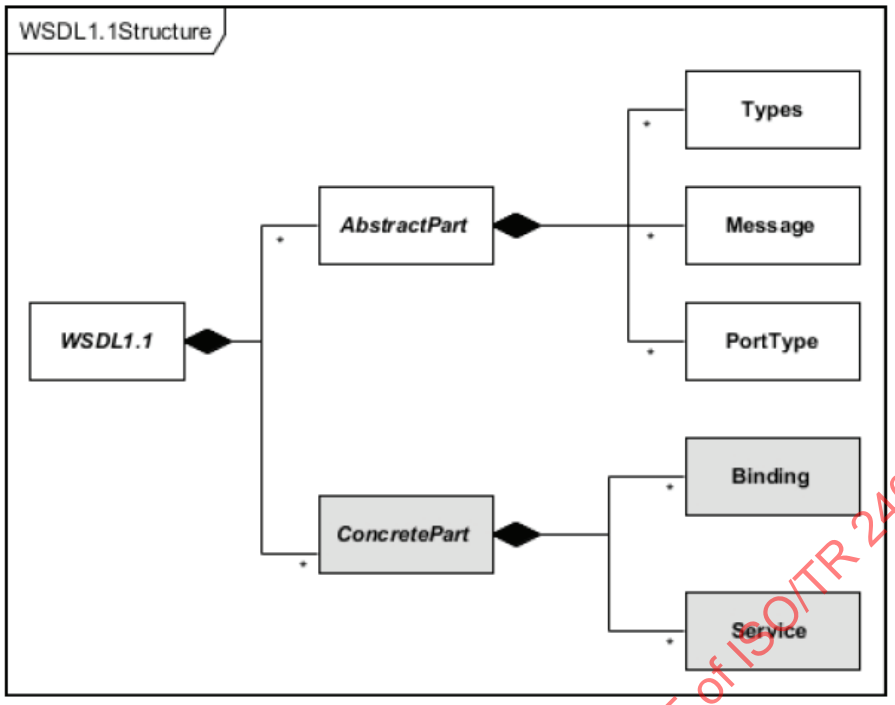


Figure 3 — WSDL 1.1 structure

As represented in Figure 3, WSDL 1.1 is constructed from an *'AbstractPart'* and a *'ConcretePart'*. The *'AbstractPart'* is communication protocol independent and the *'ConcretePart'* is the real protocol dependent part. SOAP 1.1 and SOAP 1.2 are different protocols so in order to support SOAP 1.2, a W3C member submitted the new document titled "WSDL 1.1 Binding Extension for SOAP 1.2". This document describes only the *'ConcretePart'*. This implies that the *'AbstractPart'* is common to SOAP 1.1. The full schema including *'AbstractPart'* can be downloaded from http://schemas.xmlsoap.org/wsdl/soap12/wsdl11soap12.xsd.

While this document ("WSDL 1.1 Binding Extension for SOAP 1.2") is a W3C members' submission, industry accepts it as if it were a standard. So, almost all IDEs support SOAP 1.2 generation. Therefore, there is a choice of using software vendor's SOAP 1.2 support IDE or generating a document from open software.

NOTE WSDL 1.1 instance can be created using above schema. In the case of using above schema, it is necessary to check conformance to WS-I Basic Profile 2.0 (see 8.3.2).

7.3 SOAP 1.2 usage indication

There are two methods to indicate SOAP 1.2 usage in a web service.

One method is using an implicit header. The other is an explicit way to express usage by WS-Policy. Example 1 shows an implicit method. Example 2 exhibits a policy declaration method.

EXAMPLE 1

```
<?xml version = "1.0" encoding = "utf-8" ? >
<wsdl:definitions
   targetNamespace = http://example.com
   xmlns:tns = http://example.com
   xmlns:wsoap12 = "http://schemas.xmlsoap.org/wsdl/soap12/"
   xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:xs = "http://www.w3.org/2001/XMLSchema">
…
<wsdl:types ….
```


In this method, applying SOAP 1.2 is indicated implicitly by xmlns:wsoap12 = http://schemas.xmlsoap.org/wsdl/soap12/.

EXAMPLE 2

```
<wsp:Policy>
   <wssa:SOAP12/>
</wsp:Policy>
```

If a service provider wants to use only SOAP 1.2, he/she can designate use of SOAP 1.2 in this manner. This policy assertion has End point Policy Subject.

NOTE Web service policy describes the mandatory conditions for both service provider and service consumer. This policy in WSDL 1.1 forces the consumer to use SOAP 1.2. wssa prefix as a WS-SOAP assertion. See Table 2.

If a service provider wants the current SOAP 1.1 service to be upgraded to SOAP 1.2, he/she can write policy as presented in Example 3

EXAMPLE 3

```
<wsp:Policy>
<wsp:ExactlyOne>
   <wssa:SOAP11 />
   <xs:annotation>
     <xs:documentation>SOAP1.1 ONLY backward compatibility
        for user who has already used SOAP1.1.
        It is strongly recommend to use SOAP1.2 for new users.
     </xs:documentation>
   </xs:annotation>
   <wssa:SOAP11/>
</wsp:ExactlyOne>
</wsp:Policy>
```

NOTE As WS-Policy 1.5 Framework says it does not support preferences expression, in this case prefer SOAP 1.2 to SOAP 1.1. According to the no preference support, xs:annotation is used for human reader to show SOAP 1.2 preference. For a more detailed explanation, related WS-Policy is described in ISO 24097-3[1)].

8 WS-I basic profile conformance

8.1 What is WS-I?

WS-I is an industry body, established in February 2002, which aims to make sure web services work with each other. Founded at the instigation of Microsoft and IBM, WS-I is an alliance of technology vendors and other interested parties that works alongside standards bodies.

November 10, 2010 – After nearly a decade of work and industry cooperation, the WS-I successfully concluded its charter to document best practices for Web services interoperability across multiple platforms, operating systems and programming languages.

WS-I Basic Profile (BP) products are best practice guidelines and implementation advice to help developers create web services that will live up to the vision of platform-independent interoperability, and testing tools.

A basic profile document defines the set of non-proprietary Web services specifications, along with clarifications, refinements, retentions and amplifications of those specifications which promote interoperability.

1) To be published.

Testing tools contain a set of executable test assertions for assessing the conformance to the profile and utilities. Those are the following:

— The 'Monitoring' component that captures a message trace as generated by a web service instance and a consumer;

— The 'Test Analyser' component assesses the formatted capture produced by the 'Monitoring' component and produces a test report in HTML.

Because WS is generally message exchange between different organizations, the 'Monitoring' component is expected to be useful for troubleshooting.

Current BP documents are BP 1.2 and BP 2.0. BP 1.2 is a profile of SOAP 1.1 and BP 2.0 is that of SOAP 1.2, respectively. Since the SOAP 1.2 has been selected (see 7.1), WSDL1.1 should conform to BP2.0.

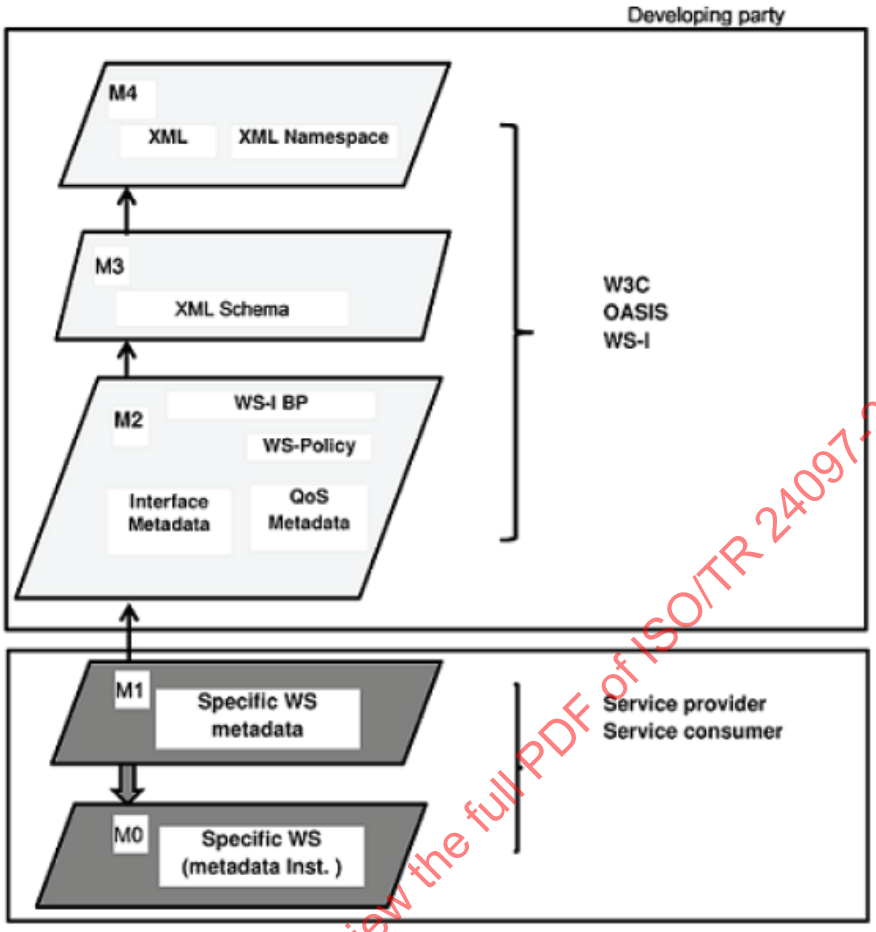
NOTE Annex C summarizes BP 1.2 and BP2.0 coverage items.

8.2 Specific WS metadata and WS metadata relationships

Figure 4 shows the relationships of WS standards and a specific WS. Here, "specific WS" means a real WS, like electronic payment WS.

The M2 layer is based on WS metadata standards developed by standard bodies such as W3C, OASIS, or WS-I. The M0 layer (specific WS) is a result of M1 layer (specific WS metadata). M0 can be generated using tools referring to M1 layer.

Specific WG metadata need to be published for potential users to evaluate. In addition, specific WS metadata are a starting point for WS creation for both the service provider and the service consumer. Therefore, specific WS metadata should conform to WS metadata standards.



Key

↑ refers

⇩ generate

Figure 4 — Specific WS metadata, specific WS, and XML standards relationship

8.3 Creating a basic profile conformant service

A basic profile consists of many requirements, such that manual validation is impractical. Therefore, to create a BP conformant service, utilizing software is the preferable way to achieve this.

Some software packages provide a batch check facility after creation of WSDL 1.1. This is a case like utilize other party developed WSDL (e.g. consumer use provider's WSDL.) This validation is done using the editor (e.g. import and save process).

When creating a new specific WS WSDL, it is typical to use an editor. At input stage, BP conformance checking is preferable way. In addition, recently developed XML editors have a facility that helps us context aware input advice. This means that the editor provides some correct candidate for user to input seeing the input context. When it ends input validation end. It promotes easy and quick development of BP conformant WSDL.

There are two methods to create BP-I conformant WSDL at the editing stage:

a using WSDL editor;

b using XML editor.

8.3.1 and 8.3.2 elaborate each method.

8.3.1 Using a WSDL editor

The following example shows BP compliant WSDL creation using the Eclipse WSDL editor. Eclipse provide three options to WS-I conformance.

a ignore compliance;

b suggest compliance;

c require compliance.

The option selected is dictated by the provider's business policy and circumstances. One possible reason to ignore compliance is that the enterprise closed or not interoperating outside its boundaries. Requiring compliance seems to fit the other cases. In the ITS WS case, considering the future need to coordinate with non-ITS business sector, choosing a BP conformant seems preferable.

In the Eclipse WSDL editor, WS-I preferences can be set at the workspace level (this means all WS) by opening the Preferences notebook, or at the project level by right-clicking on a project and selecting Preferences.

Figure 5 shows an example of the setting that all WS require WS-I compliance.

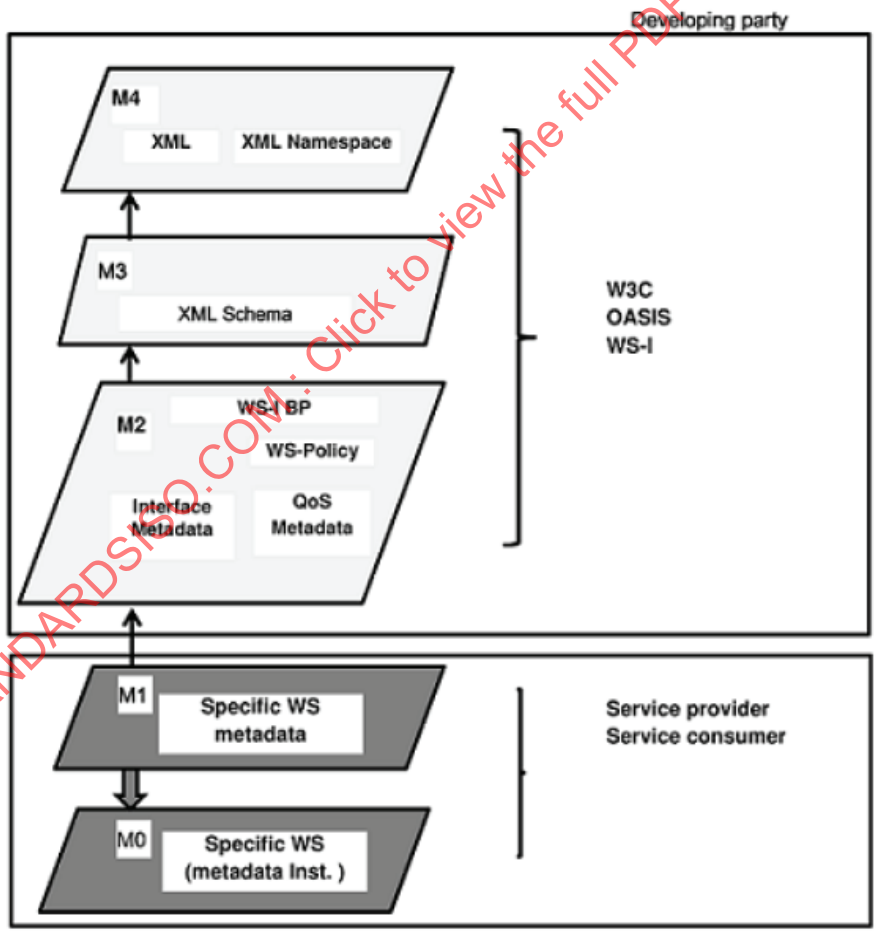


Figure 5 — WS-I preference setting

This setting requires any web services to be WS-I compliant. As shown in Figure 6, if "rpc (remote procedure call) encoded" for SOAP Binding Options is selected, the IDE reports "WS-I conformance

error: binding cannot be rpc encoded". Selecting "document literal" allows conformance to be checked by the editor.

Checking each WS with BP conformance a WS-I complaint WSDL can finally be created.

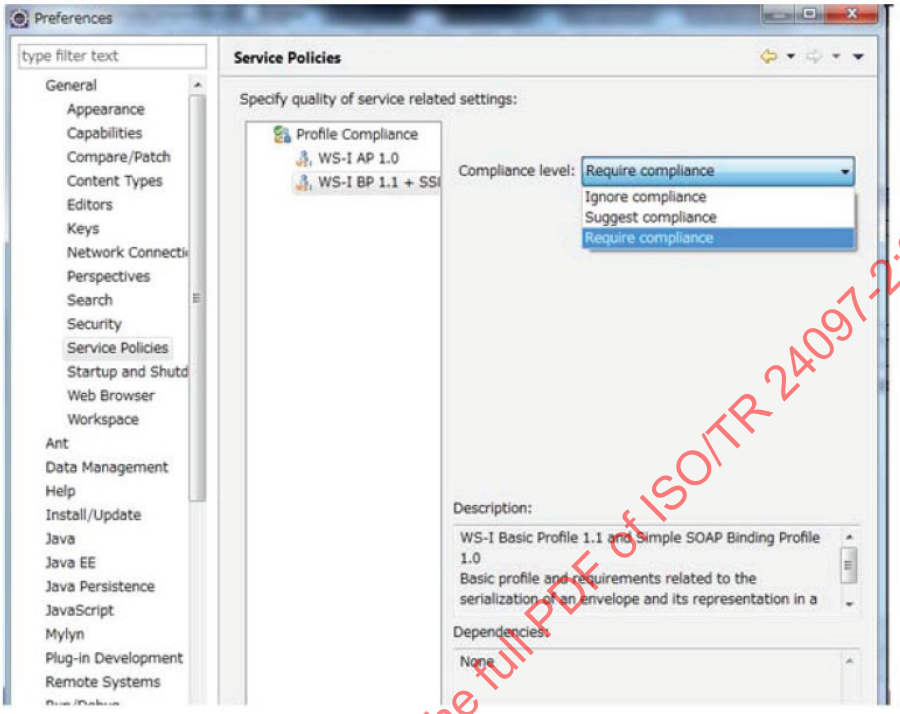


Figure 6 — WS-I conformant specific WS Interface metadata developing example

8.3.2 Using an XML editor

The previous method is applied for BP 1.1. However, BP has four versions: BP 1.0, BP 1.1, BP 1.2, and BP 2.0. If a service provider decides upon BP 2.0, then the previous method cannot be applied.

Figure 7 demonstrates the creation of a specific WS WSDL 1.1 (M1 layer of Figure 4) from WSDL 1.1 standard (M2 layer of Figure 4). This method is the most versatile. In this case (see Figure 7), "WS-IDefinedWSDL1.1Schema.xml" file is created referring to "WS-IDefinedWSDL1.1Schema.xsd".

NOTE WS-I redefined WSDL 1.1 Schema from the original W3C WSDL 1.1 Schema to increase interoperability. WS-IDefinedWSDL1.1Schema.xsd (line 6) is this schema.

As shown in Figure 7, an XML editor lists the correct input candidates depending on the cursor position (again this is a content assist) relative to the displayed text. In this example, when inputting "< " (line 9) the editor has indicated that <wsdl:binding> or <wsdl:documentation>... are the correct input candidates. In this way, one can develop WSDL efficiently.

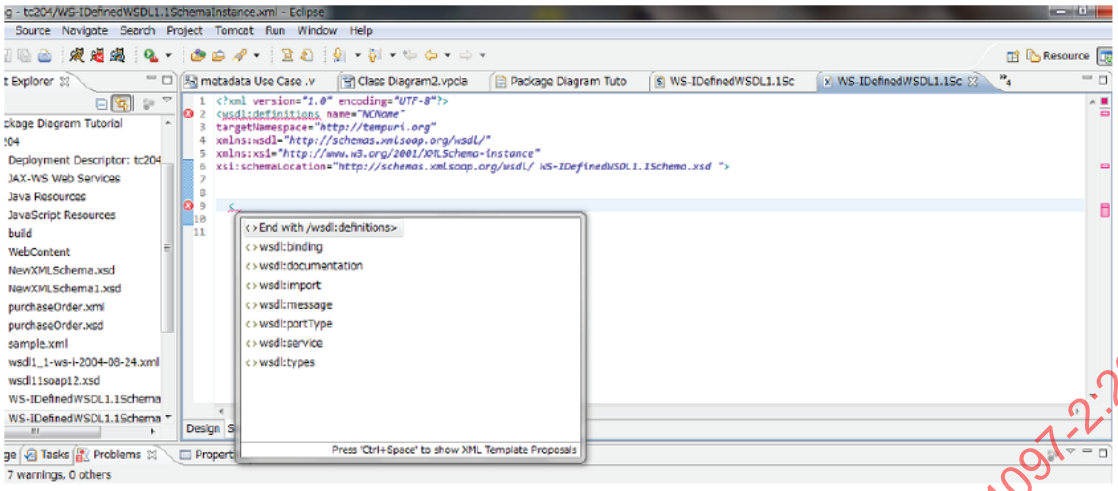


Figure 7 — Specific WS interface metadata creation example

This method makes it easier for the editor to create specific WS metadata, and when the document creation is finished, it ensures the validity of the document.

NOTE In the upper right pane, “xmlns:xsi” attribute and “xsi:schemaLocation” attribute are included in a <definitions> element. WSDL 1.1 grammar permits extension attribute (and element) if they do not contradict the grammar. See Listing 1.

Listing 1 wsdl:definitions element

```
<?xml version=”1.0” encoding=”UTF-8”?>
<wsdl:definitions name=”NCName”
targetNamespace=”http://tempuri.org”
xmlns:wsdl=”http://schemas.xmlsoap.org/wsdl/”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=”http://schemas.xmlsoap.org/wsdl/WS-IDefinedWSDL1.1Schema.xsd”>


 </wsdl:definitions>
```

XML editor validation of the listing above does not cause any grammatical errors. However, the advice provided by WS-I about extension construct is: “Development tools that consume a WSDL description and generate software for a Web service instance might not have built-in understanding of an unknown WSDL extension. Hence, use of required WSDL extensions should be avoided”. Because WS of service provision and the service consumer generally belong to different organizations, it should be safer to leave out these extension attribute after creation.

Listing 2 Eliminating extension attributes (published service provider’s WSDL)

```
<?xml version=”1.0” encoding=”UTF-8”?>
<wsdl:definitions name=”NCName”
targetNamespace=”http://tempuri.org”
xmlns:wsdl=http://schemas.xmlsoap.org/wsdl/>


</wsdl:definitions>
```

In addition in Figure 7, the file extension of instance WSDL 1.1 is ".xml" (see Project explorer pane of Figure 7). But changing it to ".wsdl" seems preferable.

NOTE In some IDEs, specialized editors such as "WSDL editor" supports the creation of a WSDL instance document. However, WS has many standards like WSDL, WS-Policy, WS-Addressing, WS-Security, etc. so the use of such application specialized editors seem not practical. The above mentioned method is generic one to be applicable in any case.

8.4 BP conformance claim

As described in the previous section, BP conformance is vital for both service providers and service consumers. BP conformance claim can be expressed in two ways: one by attaching claim in WSDL 1.1, the other is to declare conformance by WS-Policy.

The Basic Profile Version 2.0/1.2 (Final Material) says "the use of WS-Policy is RECOMMENDED over the use of the Conformance Claim Attachment Mechanisms". This is because WS policy is a mandatory condition for both service provider and service consumer, so this policy enforces even for service consumers to be WS-I conformant.

EXAMPLE 1 Conformance Claim for BP 1.2.

```
<wsdl:definitions xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl"
  xmlns:tns = "http://example.org/myservice"
  xmlns:soapbind = "http://schemas.xmlsoap.org/wsdl/soap"
  xmlns:wsi = "http://ws-i.org/schemas/conformanceClaim/"
  targetNamespace = "http://example.org/myservice">
 <wsdl:portType name = "MyPortType" >
      ...
 </wsdl:portType >
 <wsdl:binding name = "MyBinding" portType = "MyPortType" >
      ...
 </wsdl:binding >
 <wsdl:service name = "MyService" >
   <wsdl:port name = "MyPort" binding = "tns:MyBinding" >
     <wsdl:documentation >
       <wsi:Claim
         conformsTo = "http://ws-i.org/profiles/basic/1.2" / >
     </wsdl:documentation >
     <soapbind:address
       location = "http://example.org/myservice/myport" / >
   </wsdl:port >
 </wsdl:service >
</wsdl:definitions >
```

EXAMPLE 2 Policy Assertion request to BP 2.0 Conformance.

```
<wsp:Policy xmlns:bp20 = "http://ws-i.org/profiles/basic-profile/2.0/"
            xmlns:wsp = "http://www.w3.org/ns/ws-policy">
 <bp20:Conformant/>
</wsp:Policy>
```

In this EXAMPLE 2 `<bp20:Conformant` is used in order to declare BP 2.0 conformance. The `bp20:Conformant` policy assertion attached to the end point policy subject.

NOTE 1 Policy subject is described in detail in ISO/TR 24097-3[1)].

NOTE 2 Application to end point policy subject designates service is conformant to BP.

IMPORTANT — WS-I BP policy assertion Schema location

Basic Profile Version 2.0/1.2 (Final Material) says policy assertion to follow BP Version 2.0/1.0 exist at http://ws-i.org/profiles/basic-profiles/2.0/bp20.xsd or http://ws-i.org/profiles/basic-profiles/1.2/bp12.xsd, schema at that URL. Basic Profile Version 2.0/1.2 (Final Material) has a copy of schema in its Annex. For convenience, the original schema is copied in Annex D.

Annex A
(informative)

Pseudo WSDL 1.1 expression

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>

    <import namespace="uri" location="uri"/>*

    <wsdl:documentation .... /> ?

    <wsdl:types> ?
        <wsdl:documentation .... />?
        <xs:schema .... />*
        <- extensibility element -> *
    </wsdl:types>

    <wsdl:message name="nmtoken"> *
        <wsdl:documentation .... />?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>

    <wsdl:portType name="nmtoken">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <wsdl:input name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:input>
           <wsdl:output name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:output>
           <wsdl:fault name="nmtoken" message="qname"> *
               <wsdl:documentation .... /> ?
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="nmtoken" type="qname">*
        <wsdl:documentation .... />?
        <- extensibility element -> *
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <- extensibility element -> *
           <wsdl:input> ?
               <wsdl:documentation .... /> ?
               <- extensibility element ->
           <wsdl:input>
           <wsdl:output> ?
               <wsdl:documentation .... /> ?
               <- extensibility element -> *
           </wsdl:output>
           <wsdl:fault name="nmtoken"> *
               <wsdl:documentation .... /> ?
               <- extensibility element -> *
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="nmtoken"> *
        <wsdl:documentation .... />?
        <wsdl:port name="nmtoken" binding="qname"> *
           <wsdl:documentation .... /> ?
           <- extensibility element ->
        </wsdl:port>
```

```
        <- extensibility element ->
    </wsdl:service>

    <- extensibility element -> *

</wsdl:definitions>
```