TECHNICAL REPORT

ISO/TR 24097-3

First edition
2019-02

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

Part 3: Quality of service

Reference number
ISO/TR 24097-3:2019(E)

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

Published in Switzerland

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

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

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

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

In order to provide high quality ITS services, various types of service coordination are indispensable, e.g. coordination between financial industries in an Electronic Fee Collections service. Service systems are constructed in a heterogeneous platform, e.g. hardware, OS, middleware, and/or application development language. Web services are technologies for heterogeneous distributed systems coordination.

To provide web services in an agile and interoperable manner, the use of standard based metadata was proposed in ISO 24097-1. Web service (WS) metadata is a formal description of a web service. It is expressed by: ***Interface metadata*** and ***QoS (Quality of Service) metadata***. WS metadata is a technical contract between a web service provider and its consumers, so both sides are aware of this interface. This provides the base of interoperability between a service provider's program and a service consumer's program. Because metadata is based on standards, software tools can support the WS lifecycle through design to servicing and upgrading.

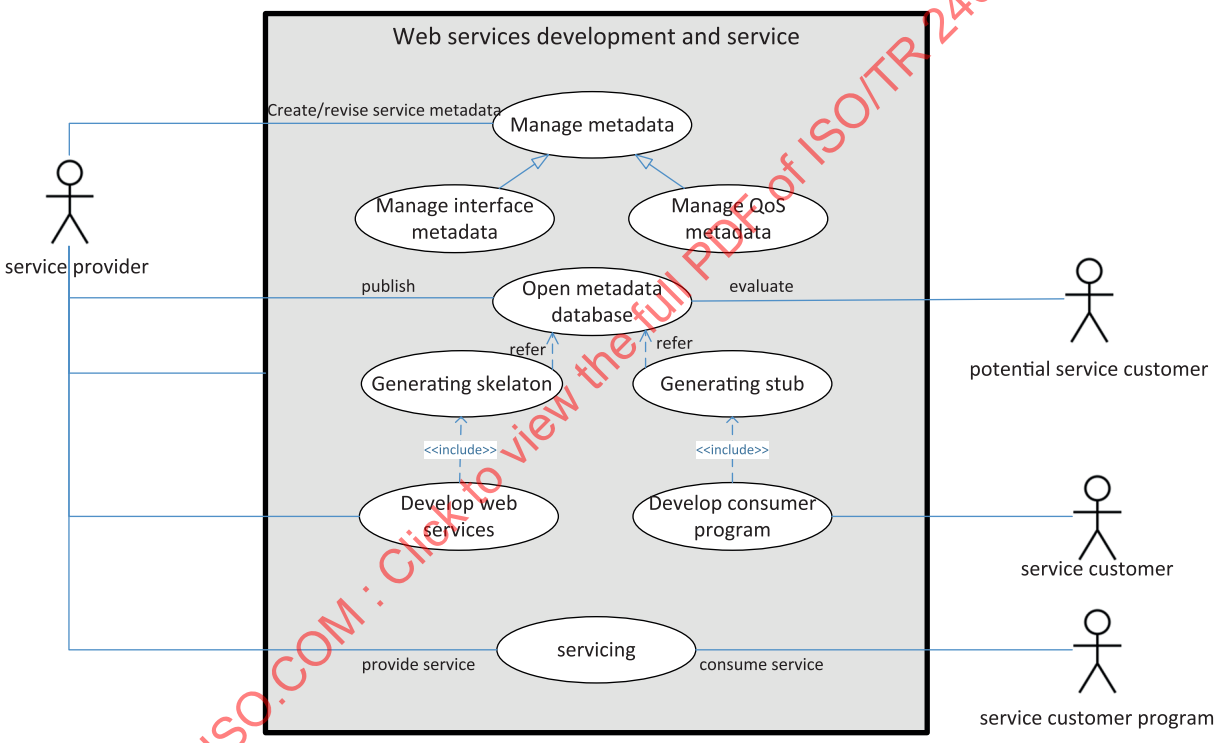


Figure 1 — ITS WS metadata use case

The ***interface metadata*** standard is the WSDL. This topic was covered in ISO/TR 24097-2.

QoS metadata is a combination of domain specific requirements and constraints such as security, reliable messaging, message addressing, and SOAP message transmission optimization.

This document focuses on these QoS topics.

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

Part 3: Quality of service

1 Scope

This document aims to promote ITS web services interoperability. Historically, web services interoperability evolved through activities shown in Figure 2. Applying the first two steps properly is the key to interoperability.

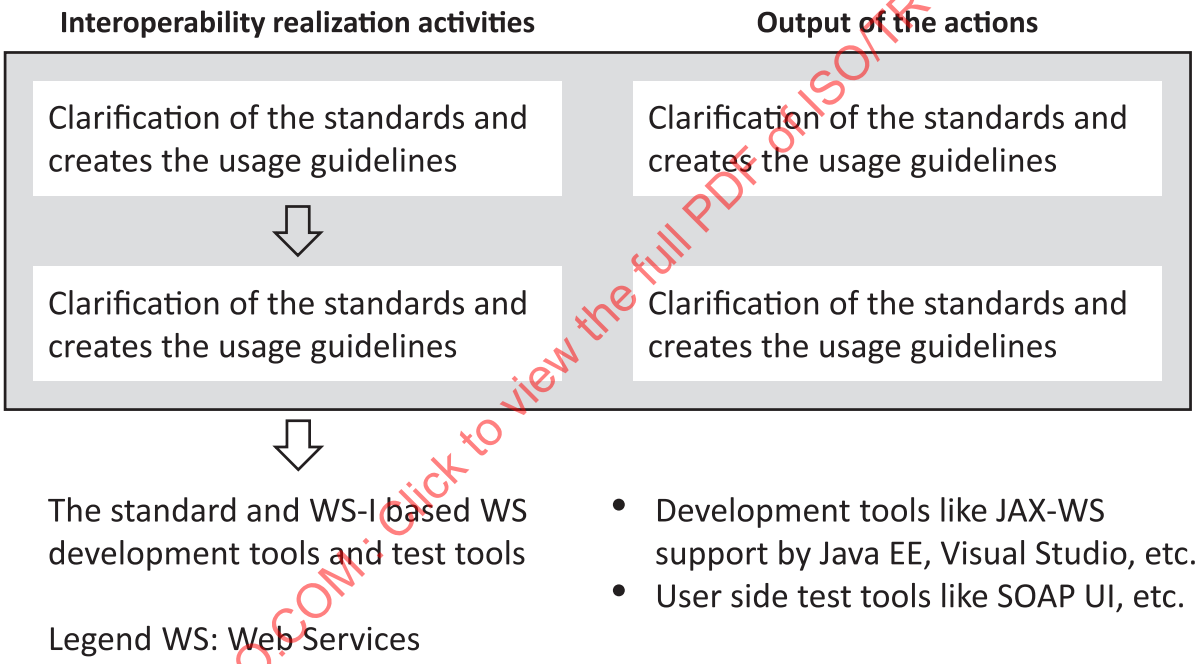


Figure 2 — Evolution of web services developing circumstances

This document focuses on the following topics:

— WS-policy language;

— domain specific policy metadata:

  — WS-Aaddressing policy metadata;

  — WS-ReliableMessaging policy metadata;

  — WS-Security Policy metadata;

  — SOAP Message transmission optimization Policy;

  — other policies.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

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

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

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

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

3.1
claim
declaration made by an entity

EXAMPLE 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 policy applies

EXAMPLE Security or message transmission reliability.

3.4
endpoint
combination of a binding and a network address

3.5
IDE
software that provides comprehensive facilities for application (including web services) development

3.6
initiator
<WS-SecurityPolicy> role sending the initial message in a message exchange

3.7
instance document
XML document that conforms to a schema

EXAMPLE If the schema is WSDL, the XML document is an WSDL instance document.

3.8
metadata
data describing an instance of WS behaviour consisting of interface metadata (WSDL description) and QoS metadata

3.9
policy assertion
<WS-Policy> requirement, capability or other property of a web service

3.10
policy subject
<WS-Policy>entity with which a policy assertion can be associated

EXAMPLE Endpoint, message, resource, operation.

3.11
recipient
<WS-SecurityPolicy> role that processes the initial message in a message exchange

3.12
security binding
<WS-SecurityPolicy> set of properties that provide enough information to secure a given message exchange

3.13
security binding assertion
<WS-SecurityPolicy> policy assertion that identifies the type of security binding being used to secure an exchange of messages

3.14
security binding property
<WS-SecurityPolicy> aspect of securing an exchange of messages

3.15
security token
token
<WS-SecurityPolicy> collection of (one or more) claims

3.16
supporting token
<WS-SecurityPolicy> security token used to provide additional claims

3.17
token assertion
<WS-SecurityPolicy> description of a token requirement

Note 1 to entry: Token assertions defined within a security binding are used to satisfy protection requirements.

3.18
web service policy language
WS-Policy
<WS-SecurityPolicy> language that is used to expresses domain specific policy and/or an instance of a web service requirement and policy attachment to relevant WSDL constructs

4 Abbreviated terms

BP	(WS-I) Basic Profile
BPEL	Business Process Execution Language
ENISA	European Union Agency for Network and Information Security Agency
EPR	End Point Reference
HTTP	Hypertext Transfer Protocol
JAX-WS	Java API for XML Web Services
JSR	Java Service Request
IDE	Integrated Development Environment
MAP	Message Addressing Property
MTOM	SOAP Message Transmission Optimization Mechanism

NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
SOA	Service Oriented Architecture
STS	Security Token Service
TC	Technical Committee
QoS	Quality of Services
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WG	Working Group
WS	Web Service
WS-I	Web Services Interoperability organization
WSSP	Web Services Security Policy
XKMS	XML Key Management Specification
XML	eXtensible Markup Language (SOAP 1.2)

NOTE SOAP is not an abbreviated terms. It is a proper noun.

5 Notation and conventions

5.1 Namespace URI and prefixes used in this document

This document uses many namespace prefixes throughout; they are listed in Table 1.

The choice of any namespace prefix is arbitrary and not semantically significant. However, the prefix shall be unique in any single document. We use the namespace prefixes as shown 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, the namespace prefix identified in Table 2 is used as a 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
wsdl20	http://www.w3.org/ns/wsdl"	WSDL 2.0
wssa	http://www.w3.org/2011/03/ws-sas	SOAP Version assertion policy
xs	http://www.w3.org/2001/XMLSchema	[XML-Schema1], [XML-Schema2]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Structures
wsic	http://ws-i.org/schemas/conformanceClaim	WS-I conformance claim
wssa	http://www.w3.org/2011/03/ws-sas	WS-SOAPAssertions

Table 2 *(continued)*

Prefix	XML namespace URI	Specifications
bp12	http://ws-i.org/profiles/basic-profile/1.2/"	BP 1.0
wsp	http://www.w3.org/2006/07/ws-policy	WS-Policy 1.5
wsam	http://www.w3.org/2007/05/addressing/metadata	WS-Addressing metadata
wsrmp	http://docs.oasis-open.org/ws-rx/wsrmp/200702	WS-ReliableMessaging Policy Assertion
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	Utility schema
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[WS-Security]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy.xsd	WS-SecurityPolicy 1.3
ds	http://www.w3.org/2000/09/xmldsig#	[XML-Signature]
xenc	http://www.w3.org/2001/04/xmlenc#	[XML-Encrypt]
wsc	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512	This specification
mtom	http://www.w3.org/2007/08/soap12-mtom-policy	MTOM Serialization Policy Assertion
tns		The “this namespace” (tns) prefix is used as a convention to refer to the current web service.

When developing web services, we need to check what version(s) of the standards the software under development should support. In a standards body like W3C and OASIS, new versions are frequently released, but support tools follow behind with some time lag. Therefore, it is up to the developers to determine if it is appropriate to implement the latest versions.

5.2 Web service syntax notation: pseudo-schemas

Every web service language, e.g. WSDL, WS-Policy, has its own schema to validate the user’s instance description. Because web service language is complex, it is helpful to know the grammar at a glance. For that reason, 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 do 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

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

EXAMPLE

```
/wsdl11:definition/wsdl11:message, wsdl11:definition/wsdl11:binding/wsdl11:@name
```

5.4 XML infoset

A WS-Policy document relies on the XML Information Set [XML Information Set]. Information item properties are indicated by the style [infoset property].

5.5 SOA stack name notation

The SOA stack name is represented in bold italics.

EXAMPLE ***messaging***

5.6 Examples

To clarify an explanation, we give examples using "Eclipse" (from Eclipse Foundation) IDE and its plug-in web tool platform (hereafter WTP). This is only an example of an available tool and not a recommendation for Eclipse.

We selected Eclipse because it:

— is open software (not specific vendor software);

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

— supports many high-quality candidate plug-ins;

— has a context-based wizard;

— was developed by a community that included W3C members;

— has reasonable documentation; and

— is similar to many commercial IDEs, which should facilitate other IDE users to understand the information presented in this document.

In this document all examples are informative.

6 Web services overview

The ISO 24097 series enables a plurality of geographically dispersed intelligent transport systems using various platforms to exchange necessary information, thereby realizing a service that is high in quality and reflects user needs. For service providers and service users to realize interoperability, the requirements shown in Table 1 are necessary.

Table 1 — Basic requirements for interoperable ITS web services

Basic requirements	Requirement content	Reference material
To ensure interoperability between the ITS service provider and the service user, the service provider describes and discloses the technical requirements of the ITS service in metadata. The service user creates a program according to the metadata.	Describe the technical requirements with interface and QoS metadata and make it public to the service user in some way (e.g. UDDI and/or endpoint reference). Service users develop user systems under these conditions.	ISO 24097-1:2017
Metadata standards are sometimes generalized, and if you can implement interoperability with restrictions, use WS-I.	When there is a provision in WS-I, use metadata within that range.	ISO 24097-1:2017, A.4.2
As the ITS service may evolve, version control is necessary. When considering new services, consider backward compatibility and protect service users when possible. If backward compatibility cannot be implemented, change the major version number (see the right column), but continue the old version service for a reasonable period.	For each interface and QoS domain metadata, give a version number. Version numbers are given in the following system: Form: m.n.a: m: major version number (`xs:positiveInteger`) n: minor version number (`xs:nonNegativeInteger`) a: draft version number (`xs:NCName`)	ISO 24097-1:2017, 6.2.1.3 ISO 24097-1:2017, 7.2.4
Interface metadata	Use WSDL 1.1.	ISO/TR 24097-2:2015, 6.1
SOAP version	Prioritize SOAP 1.2 over SOAP 1.1.	ISO/TR 24097-2:2015, 7.1
Description of QoS	Define the QoS for each domain. Other than SOAP MTOM, it is described as a policy standardized by the QoS standard. Recommend description of SOAP MTOM based on JSR 224	This document
Enable QoS	Other than SOAP MTOM, based on the WS-PolicyAttachment standard, considering the scope of policy, give it to the correct attachment point.	This document

7 QoS overview

The QoS describes the requirements and constraints of a web service. The following figure shows the whole web services stack. The black shaded parts are relevant to QoS standards.

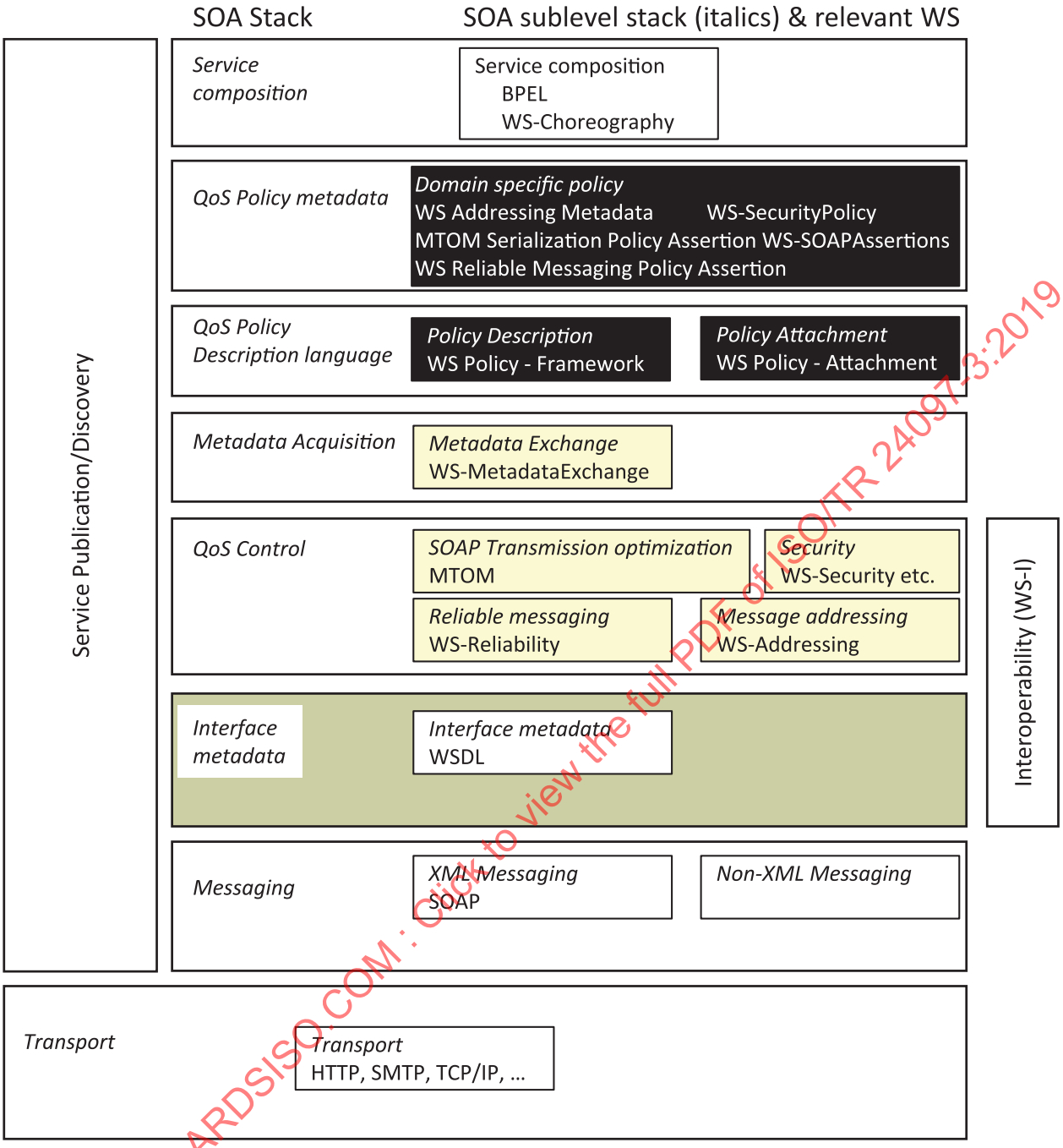


Figure 3 — QoS relevant metadata

8 QoS standards

Web services metadata is a high-level description of a specific web service. It consists of the *Interface metadata* and the *QoS metadata*.

The *Interface metadata* is WSDL 1.1 and/or WSDL 2.0.

The *QoS metadata* is an expression of the required functionality or constraint of a specific web service. A web service requires various types of functionality or constraints such as reliable message transmission, secure messaging, and/or data transmission optimization. Therefore, QoS metadata are divided into domain-specific categories based on SOA principles. This allows domain-specific experts to standardize

content based on their expert knowledge and rich experience. Modular structure standards enable users (service providers and/or service consumers) to select functions and constraints as needed.

To develop domain-specific QoS standards, the W3C prepared a common policy description language known as the WS-Policy. The latest domain-specific QoS standards using WS-Policy are: Web Services Policy 1.5 — Framework and Web Services Policy 1.5 — Attachment.

The Web Services Policy 1.5 — Framework is applied to describe domain-specific standards. All domain-specific policy standards are described by this common syntax (see Figure 4). Common domain-specific policies include standardized Security Policy, Messaging Policy, Reliable Messaging Policy, and SOAP Messaging Policy. If policy users want their specific policy, users can create their policy based on the WS-Policy language.

The Web Services Policy 1.5 — Attachment enables a policy attachment to WSDL and/or UDDI and executes the policy at run time. The user is responsible for adding the attachment as needed. This language provides a combining facility for domain-specific policies, e.g. applying both a security policy and a reliable transmitting policy, which is called a "**merge**" in the standard.

The following figure depicts the policy language structure and the relationship between Interface metadata.

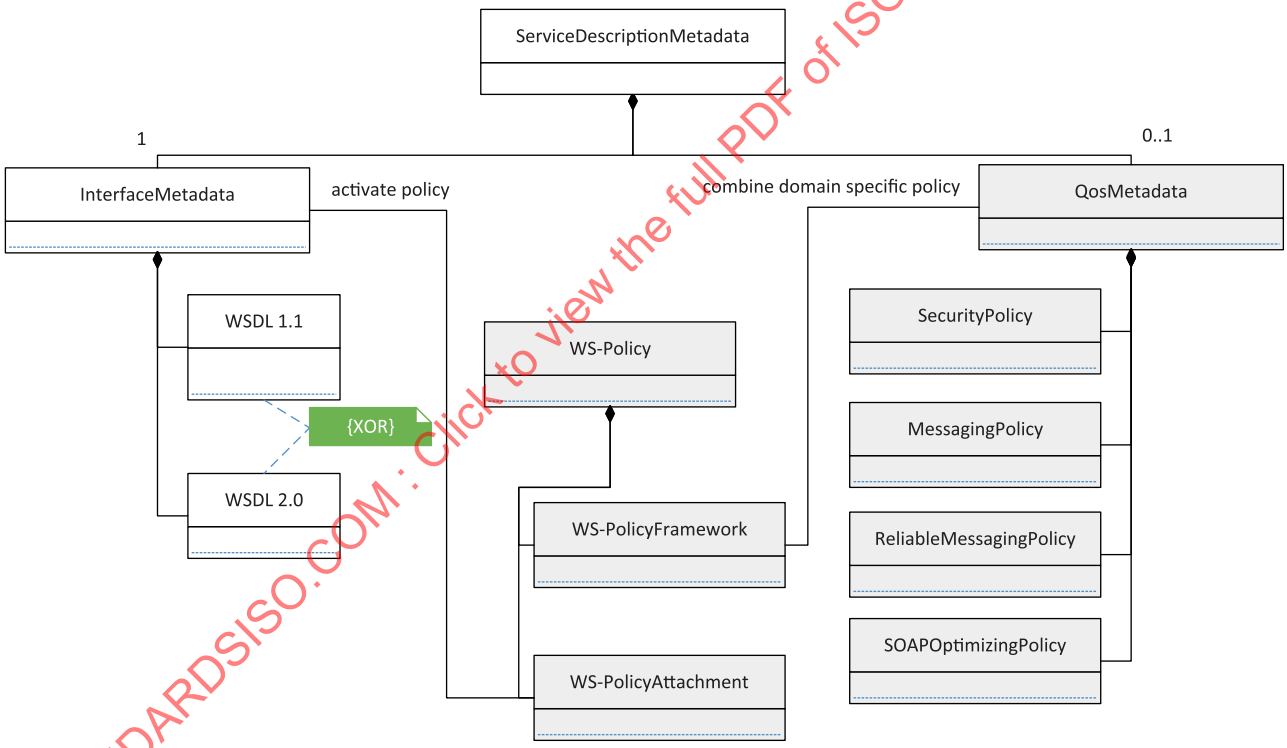


Figure 4 — Web services metadata and their relationships

8.1 WS-Policy language

Table 3 summarizes the WS-Policy language standards and the introductive or guideline documents developed by W3C.

Table 3 — WS-Policy related documents

Standard	Description	Document location
Web Services Policy 1.5 — Framework	Provides a general-purpose policy model and corresponding syntax to describe the policies.	http://www.w3.org/TR/ws-policy/
Web Services Policy 1.5 XML Schema URI	http://www.w3.org/2007/02/ws-policy.xsd	
Web Services Policy 1.5 — Attachment	Describes a mechanism for attaching policies to WSDL or UDDI.	http://www.w3.org/TR/ws-policy-attach
Introductive or guideline document		
Web Services Policy 1.5 — Primer	Introductive document of explaining how to use WS-Policy 1.5	http://www.w3.org/TR/ws-policy-primer/
Web Services Policy 1.5 — Guidelines for Policy Assertion Authors	Provides guidance for domain specific assertion authors that will work with the Web Services Policy 1.5. Some parts for service providers.	http://www.w3.org/TR/ws-policy-guidelines/

8.2 WS-Policy 1.5 — Framework

The WS-Policy 1.5 Framework is able to model both a single policy (a collection of assertions that acts as one policy, without alternatives) and also multiple policy alternatives. This means that model service providers may express their policies in multiple policy alternatives and thereby allow users to select the policy that suits them best.

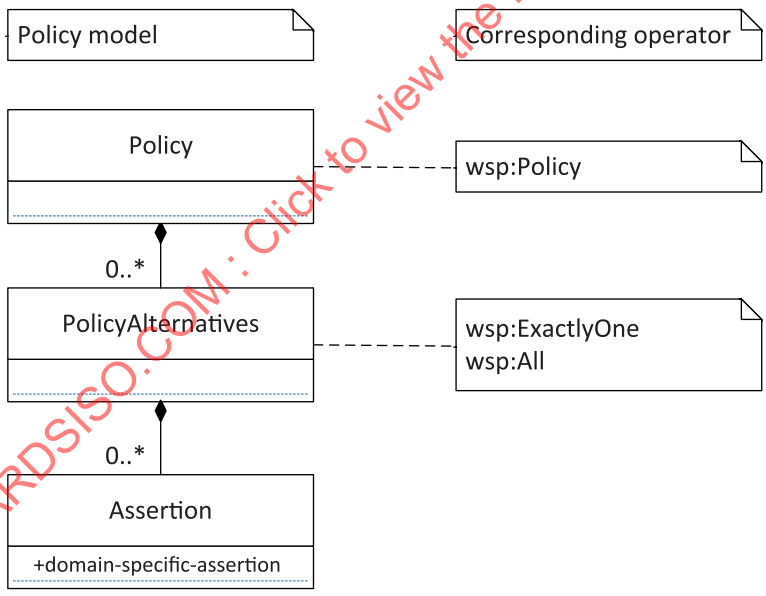


Figure 5 — WS-Policy model

8.2.1 Policy authoring style

W3C's WS-Policy working group organizes policy expressions in two forms: normal form and compact form. The two forms are semantically the same.

The normal form enumerates all possible cases (alternatives) the service supports; each alternative is mutually exclusive. The normal form follows the syntax below:

```
(01) <wsp:Policy … ( Name="xs:anyURI" )?  ( wsu:Id="xs:ID" | xml:id="xs:ID" )?>
(02)   <wsp:ExactlyOne>
(03)      ( <wsp:All> ( <Assertion …> … </Assertion> )* </wsp:All> )*
```



```
(04)    </wsp:ExactlyOne>
(05) </wsp:Policy>
```

A graphical representation of the syntax above is shown in Figure 6.

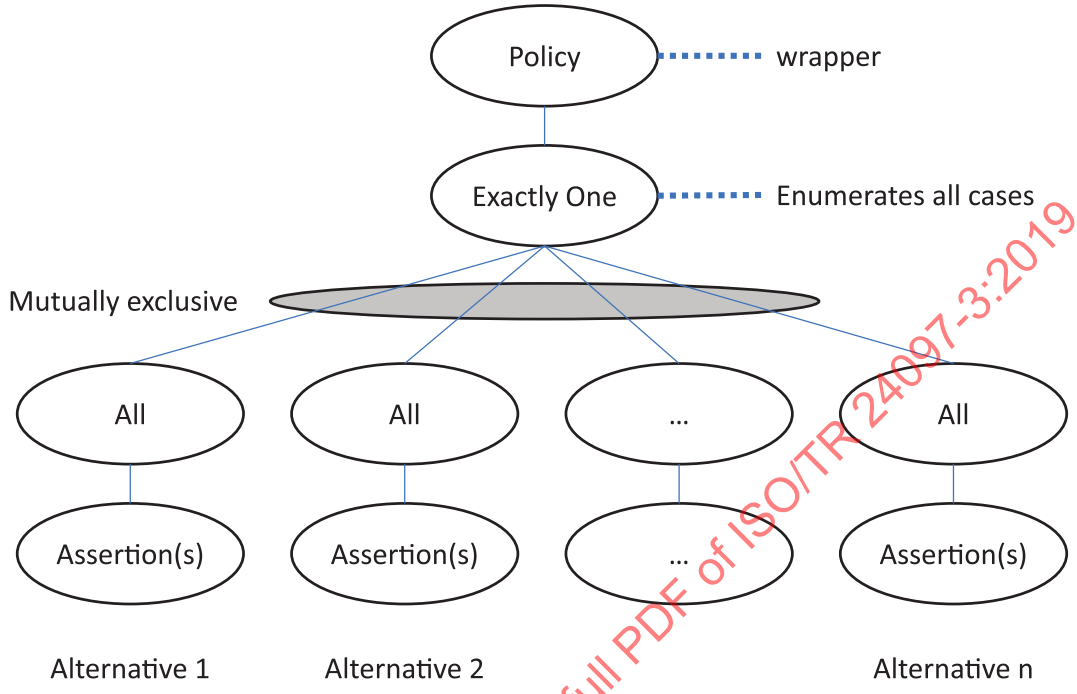


Figure 6 — Normal form expression

The normal form follows the rules below:

1) Each alternative is mutually exclusive.

2) Each nested policy contains at most one policy alternative [due to rule 1)].

This usually causes verbose policy expression compared to the compact form, which is shown below:

1) The compact form syntax:

```
<wsp:Policy … >
  ( <wsp:Policy …>…</wsp:Policy> |
    <wsp:ExactlyOne>…</wsp:ExactlyOne> |
    <wsp:All>…</wsp:All> |
    <wsp:PolicyReference … >…</wsp:PolicyReference> |
    …
  )*
</wsp:Policy>
```

2) No additional constraints;

3) The `wsp:Optional = "xs:boolean"` attribute may be attached to `Assertion` element. Compact form `<Assertion … wsp:optional = "true"…>` is equivalent to normal form;

```
<wsp:ExactlyOne>
  <wsp:All>
    <Assertion> … </Assertion>
  </wsp:All>
  <wsp:All >
    <Assertion />
  </wsp:All>
</wsp:ExactlyOne>
```

The following is an example from the WS-Policy 1.5 Framework document, which is useful to compare both forms. The compact form is:

```
(01) <wsp:Policy xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)    <sp:TransportBinding>
(03)      <wsp:Policy>
(04)        <sp:AlgorithmSuite>
(05)          <wsp:Policy>
(06)            <wsp:ExactlyOne>
(07)              <sp:Basic256Rsa15 />
(08)              <sp:TripleDesRsa15 />
(09)            </wsp:ExactlyOne>
(10)          </wsp:Policy>
(11)        </sp:AlgorithmSuite>
(12)        <sp:TransportToken>
(13)          <wsp:Policy>
                  <sp:HttpsToken>
                    <wsp:Policy/>
                      </sp:HttpsToken>
(15)             </wsp:Policy>
(16)        </sp:TransportToken>
            <!-- Details omitted for readability -->
(17)       </wsp:Policy>
(18)     </sp:TransportBinding>
(19) </wsp:Policy>
```

The equivalent normal form of the expression would be:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)    <wsp:ExactlyOne>
(03)      <wsp:All>
(04)        <sp:TransportBinding>
(05)          <wsp:Policy>
(06)            <sp:AlgorithmSuite>
(07)              <wsp:Policy>
(08)                <sp:Basic256Rsa15 />
(09)              </wsp:Policy>
(10)            </sp:AlgorithmSuite>
(11)            <sp:TransportToken>
(12)              <wsp:Policy>
                    <sp:HttpsToken>
                      <wsp:Policy/>
                    </sp:HttpsToken>
(14)              </wsp:Policy>
(15)            </sp:TransportToken>
                <!-- Details omitted for readability -->
(16)          </wsp:Policy>
(17)        </sp:TransportBinding>
(18)      </wsp:All>
(19)      <wsp:All>
(20)        <sp:TransportBinding>
(21)          <wsp:Policy>
(22)            <sp:AlgorithmSuite>
(23)              <wsp:Policy>
(24)                <sp:TripleDesRsa15 />
(25)              </wsp:Policy>
(26)            </sp:AlgorithmSuite>
(27)            <sp:TransportToken>
(28)              <wsp:Policy>
                    <sp:HttpsToken>
                      <wsp:Policy/>
                    </sp:HttpsToken>
(30)              </wsp:Policy>
(31)            </sp:TransportToken>
                <!-- Details omitted for readability -->
(32)          </wsp:Policy>
(33)        </sp:TransportBinding>
```

```
(34)      </wsp:All>
(35)    </wsp:ExactlyOne>
(36) </wsp:Policy>
```

Comparing both expressions, the compact form is easier to read and to understand.

W3C Web Services Policy 1 [11] subclause 4.1 states "to simplify processing and improve interoperability, the normal form of a policy expression SHOULD be used where practical". However, service users might not have their own described policy. Moreover, some potential service users might not have software to calculate compatibility (see paragraph below). Thus, in this case, the normal form cannot be used to calculate compatibility. Considering this, the compact form may be the better choice.

If we encrypt one field message, are the policy alternatives exclusive? They are not exclusive even though they look like they would be exclusive. This requires a semantic check, which might not be easy to do.

8.2.2 A policy description by combining domain specific policies

The WS-Policy enables the web services to be developed by dividing policy into modular domain-specific policies. As a result, service providers are able to construct their own policy by combining domain-specific policies. This results in not only higher productivity but also easier readability and maintainability.

The compact form `wsp:PolicyReference` lets us facilitate outer policy reference. The syntax is as follows:

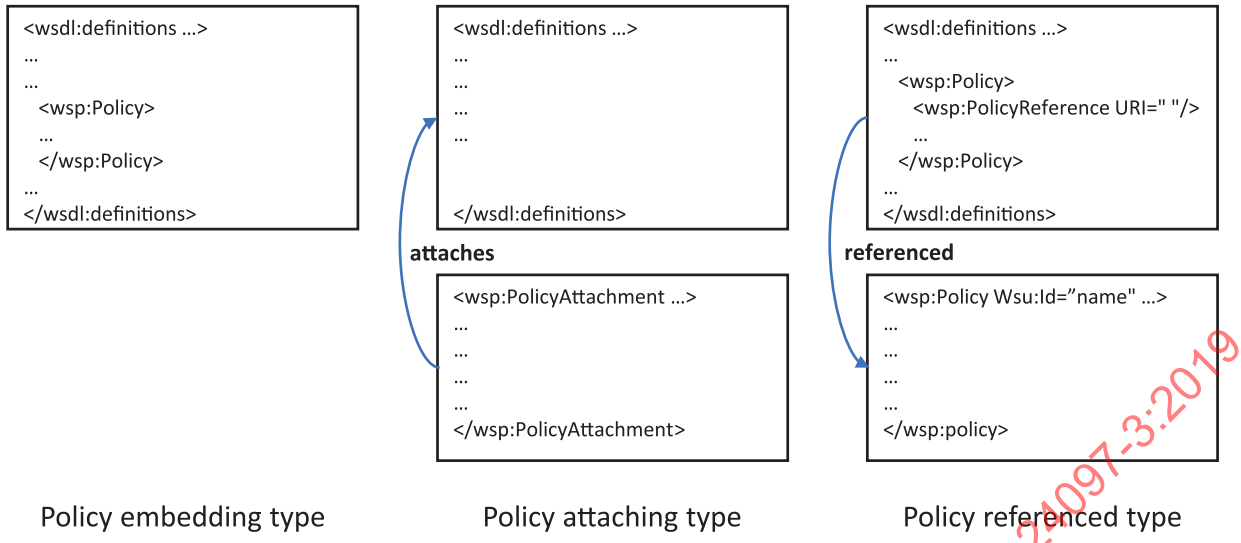
```
(01) <wsp:PolicyReference
(02)      URI="xs:anyURI"
(03)    ( Digest="xs:base64Binary" ( DigestAlgorithm="xs:anyURI" )? )?
(04)      … >
(05)     …
(06) </wsp:PolicyReference>
```

The default digest algorithm is the "SHA1 hash", athough additional algorithms can be expressed. The digest value to check is `xs:base64Binary`.

NOTE As shown in the compact form, psuedo schema, we can use `wsp:PolicyRerence` as many times as desired. This means we can create domain specific policies independently and attach them to a WSDL attach point independently. This offers an easier way to create a policy and promotes readability.

8.3 WS-Policy 1.5 — Attachment

The WS-Policy language is independent from WSDL; a policy can be attached to WSDL in one of three ways: embedding a policy in WSDL directly, associating an outer policy to the WSDL, or creating an outer policy reference from the WSDL.



NOTE: `wsp:Policy` attribute plays same functionality as `wsp:PolicyReference` element.

Figure 7 — Policy association styles

"Embedding type" is an expression that captures both ***interface metadata*** and ***QoS metadata*** in one document. It consists of numerous statements, and reuse of the ***QoS metadata*** outside of the WSDL is impossible. This style may be the best option for a service provider who has a small number of web services.

"Policy attaching type" is another recommended way by which we can expect the following benefits:

— Independent policy development from ***interface metadata***. This allows designers to concentrate on policy, and in some situations, policy can be developed by domain specific professionals;

— Increase of reusability. If a service provider has many WSs, the policy requirements among the services may be similar, in which case the policy becomes reusable; and

— Ease of maintenance.

When using the `wsp:PolicyReference` element, a service consumer will know what kind of QoS is required with the WSDL.

Outer policy reference is done using the `wsp:PolicyReference`. Using `wsp:PolicyAttachment` it may be difficult for a consumer to know what kind of policy is required. It may be useful for a service provider to govern the policies.

The syntax of `wsp:PolicyReference` is:

```
<wsp:PolicyReference
     URI="xs:anyURI"
    ( Digest="xs:base64Binary" ( DigestAlgorithm="xs:anyURI" )? )?
     … >
     …
</wsp:PolicyReference>
```

The `wsp:PolicyReference/@Digest` and `wsp:PolicyReference/@DigestAlgorithm` provide the WS consumer the referenced policy integrity.

The following listing is an example of `wsp:PolicyReference` usage. In this example `wsdl:definition` refers to the outer policy of `/wsp:policy/@common`.

```
 <wsdl:definition …>
    …
```

```
    <wsp:PolicyReference URI = http://www.example.org/tentative#common xmlns:wsp =
    "http://www.w3.org/ns/ws-policy" />
    …
 </wsdl:definition>
```

A referenced policy needs the `wsu:Id` attribute. In this case, the common policy document is located at http://www.example.org/tentative.

```
<wsp:Policy wsu:Id="common">
   <mtom:OptimizedMimeSerialization wsp:Optional="true"
  xmlns:mtom="http://www.w3.org/2007/08/soap12-mtom-policy" />
   <wsam:Addressing>…</wsam:Addressing>
</wsp:Policy>
```

NOTE There are three methods to identify policy: `wsu:Id`, `xml:id`, and `wsp:Name`. The schema outline for attributes to associate an IRI is as follows: `<wsp:Policy ( Name="xs:anyURI" )? ( wsu:Id="xs:ID" | xml:id="xs:ID" )?.>`.

The W3C Policy standard recommends using the `wsu:Id` rather than `xml:id` because `xml:id` can cause incompatibility problems when calculating a digital signature.

8.3.1 Combining multiple policies

It is considered common to combine multiple domain specific policies simultaneously, e.g. security policy and reliability policy, because it is easier to consider and to manage independent policies. As a result, multiple policies may be attached to an identical policy subject. In this case, all independent policies are combined in the `wsp:All` element (see EXAMPLE below).

EXAMPLE

```
<wsdl:definition …>
 …
  <wsdl:binding…>
     <wsp:policyReference #policy1…>
     <wsp:policyReference #policy2…>
…
…
</wsdl:definition>
```

The system software would then interpret the policy as follows: (this process is called a **merge**):

```
<wsdl:definition …>
 …
   <wsdl11:binding…>
     <wsp:Policy …>
        <wsp:All>
          <wsp:policyReference #policy1…>
          <wsp:policyReference #policy2…>
        </wsp:All>
     <wsp:Policy >
   </wsdl11:binding>
…
</wsdl:definition>
```

8.3.2 Policy attachment points, policy subjects, and policy scope

To implement a policy as intended, the receiver needs to know the basic behaviour of the policy attachment. Figure 8 shows the WSDL 1.1 structure, policy attachment points, and the policy subject. The policy attachment points are the places where we can attach and associate policies. In Figure 8, the attachment points are: Service, Port, Binding, Binding/Operation/input, Binding/Operation/output, Binding/Operation/fault, Porttype, Porttype/Operation/input, Porttype/Operation/output, Porttype/Operation/fault, and Message element. The policy subjects are the grouping concept of the similar policy attachment points.

NOTE Some domain specific policies specify the attachment point(s). In that case, we follow the specification.

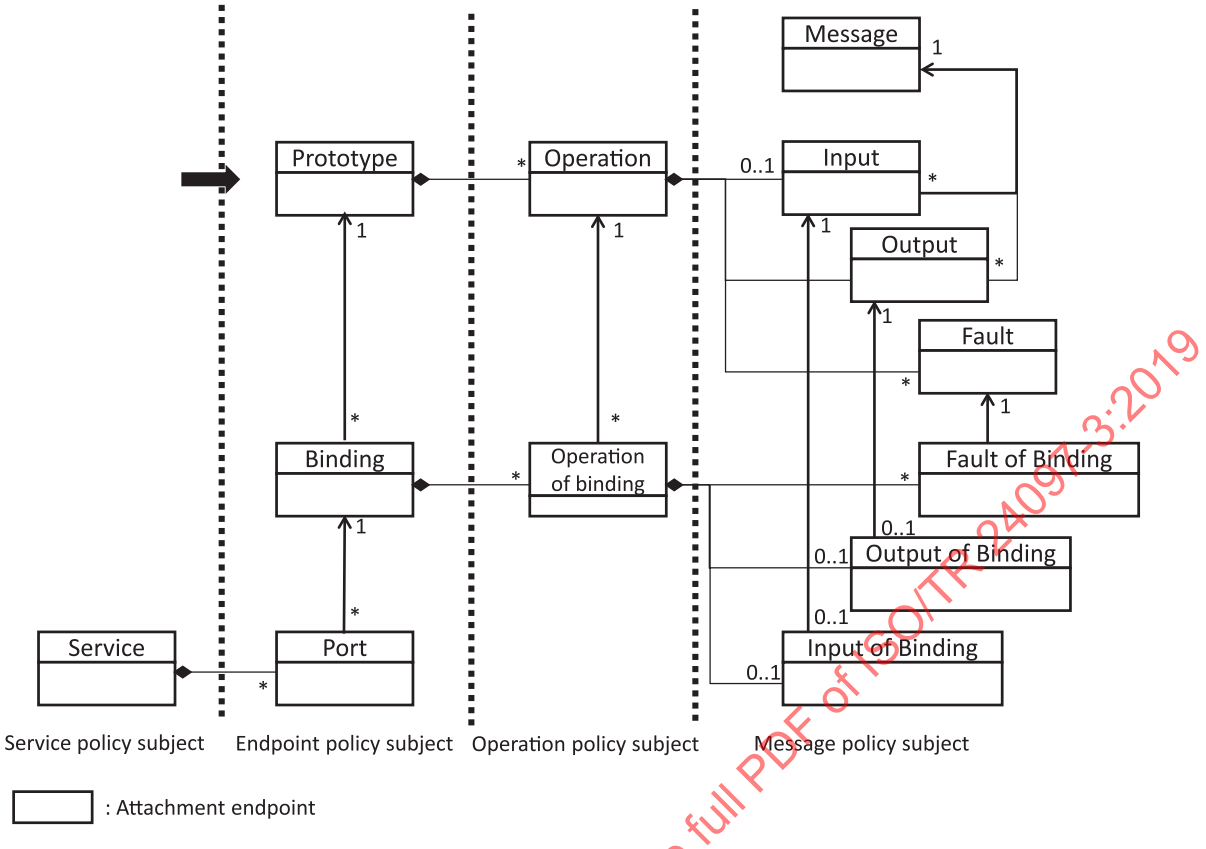


Figure 8 — Policy attachment points and policy subjects in WSDL 1.1

W3C defines the policy scope as a collection of policy subjects to which a policy can apply. When we attach a policy expression to an attaching point, it propagates to an association relationship policy attachment. Figure 9 shows how to propagate the policy expression.

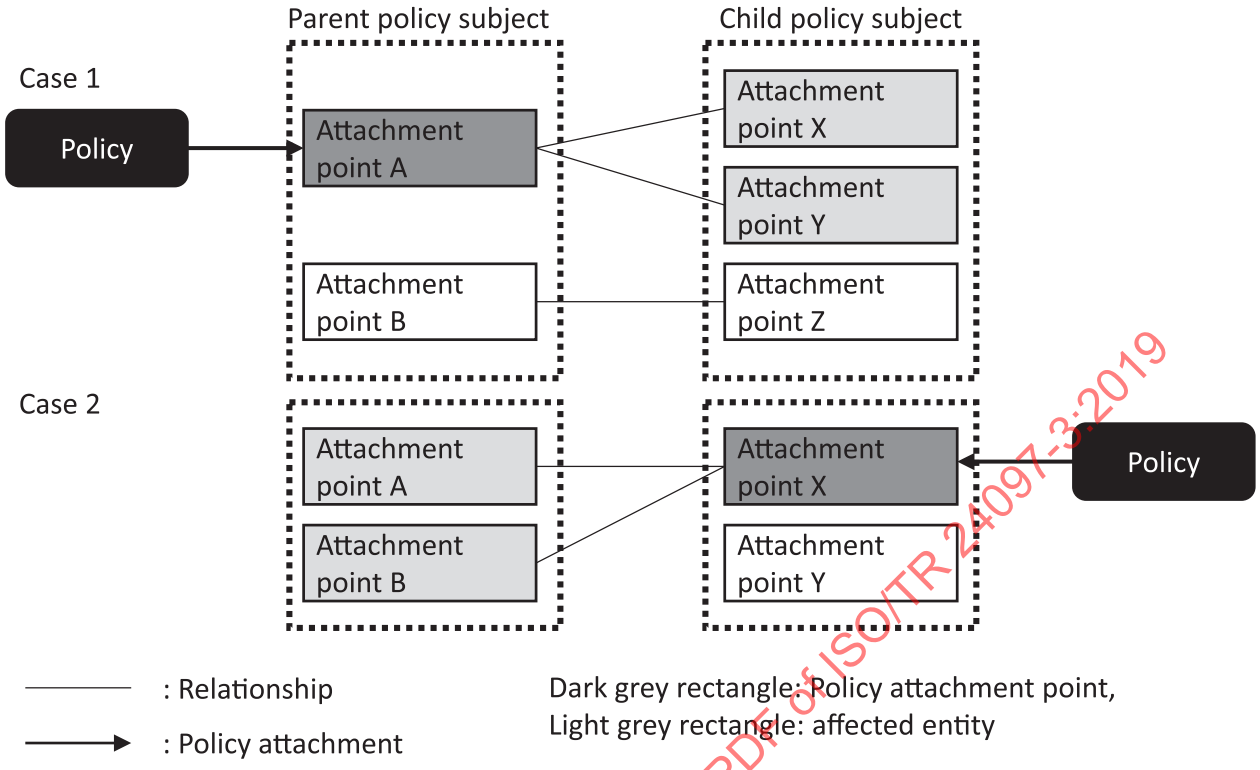


Figure 9 — Attached policy scope example

If we attach a policy expression to an attachment point A, it propagates to the associating child policy subject attachment point X and the attachment point Y (inheritance), and vice versa. Then the scope of the attaching policy is Attachment point A, X, and Y, ... (Case 1). In contrast, if we attach a policy expression to a child policy attachment point, it affects to the parent policy attachment point (Case 2). When we attach a policy expression to some point, we are required to see the policy scope.

9 Domain specific policy overview

Web services are developed two ways:

— Top-down approach, or

— Bottom-up approach.

The top-down approach starts with **interface metadata** and attaches **QoS metadata** to the interface metadata.

The bottom-up approach starts from a production program to create **interface metadata** and **QoS metadata**. Both metadata are used to evaluate a service from a technical viewpoint and if a service consumer decides to use the service, he/she develops the program using the metadata. In this document, only the top-down approach is considered. The standards for this are shown in Table 4.

Table 4 — Domain specific policy standards

Standardization domain	Current standard[a]
Web service security	WS-SecurityPolicy 1.3
Reliable messaging	Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.2
Sophisticated messaging	Web Services Addressing 1.0 — Metadata
SOAP Message Transmission Optimization	MTOM Serialization Policy Assertion 1.1

[a] If a version number exists, e.g. "WS-SecurityPolicy *1.3*", it is the latest version.

NOTE Though the WS-Policy language supports the definition and use of user specific policy, it is out of the scope of this document.

9.1 *Messaging metadata* (WS-Addressing metadata)

9.1.1 WS-Addressing standard

Table 5 summarises the standards related to WS-addressing:

Table 5 — WS-Addressing standards

Standard name	Description	Document URI (the first) and the schema location (the second)
Web Services Addressing 1.0 — Core (2006-05-09)	Definition of abstract WS-Addressing information items.	`http://www.w3.org/TR/ws-addr-core/` `http://www.w3.org/2005/08/addressing.xsd`
Web Services Addressing 1.0 — SOAP Binding (2006-05-09)	WS-Addressing information item's mapping to SOAP	`http://www.w3.org/TR/ws-addr-soap/` `The schema is included in the upper one.`
Web Service Addressing 1.0 — Metadata (2007-09-04)	Definition of how to include abstract WS-Addressing information in WSDL (1)	`http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/` `http://www.w3.org/2007/05/addressing/metadata.xsd`
Test tool	**Document URI**	
Web Services Addressing 1.0 Test Suite	`http://dev.w3.org/cvsweb/~checkout~/2004/ws/addressing/testsuite/Overview.html?content-type=text/html;%20charset=iso-8859-1`	
Web Services Addressing 1.0 Metadata Test Suite	`http://dev.w3.org/cvsweb/~checkout~/2004/ws/addressing/testsuitewsdl/Overview.html?content-type=text/html;%20charset=iso-8859-1`	

Note that Old "Web Services Addressing 1.0 — WSDL Binding" is replaced by the new "WS Addressing 1.0 — Metadata". However, some software vendors still support this old version, which can cause confusion.

NOTE In the Web Service Addressing 1.0 — Metadata, terms are based on WSDL 2.0. Therefore, when using WSDL 1.1 rewording the WSDL 2.0 based terms to the WSDL 1.1 terms: `Endpoint` ⇒ `wsdl:port`, `InterfaceName` ⇒ `wsdl:portType`, and `ServiceName` ⇒ `wsdl:service`.

In this document when we use the term WS-Addressing, it implies WS-Addressing 1.0 — Core, Web Services Addressing 1.0 — SOAP Binding, and Web Service Addressing 1.0 — Metadata. The WS-Addressing standards and their roles are depicted in Figure 10.

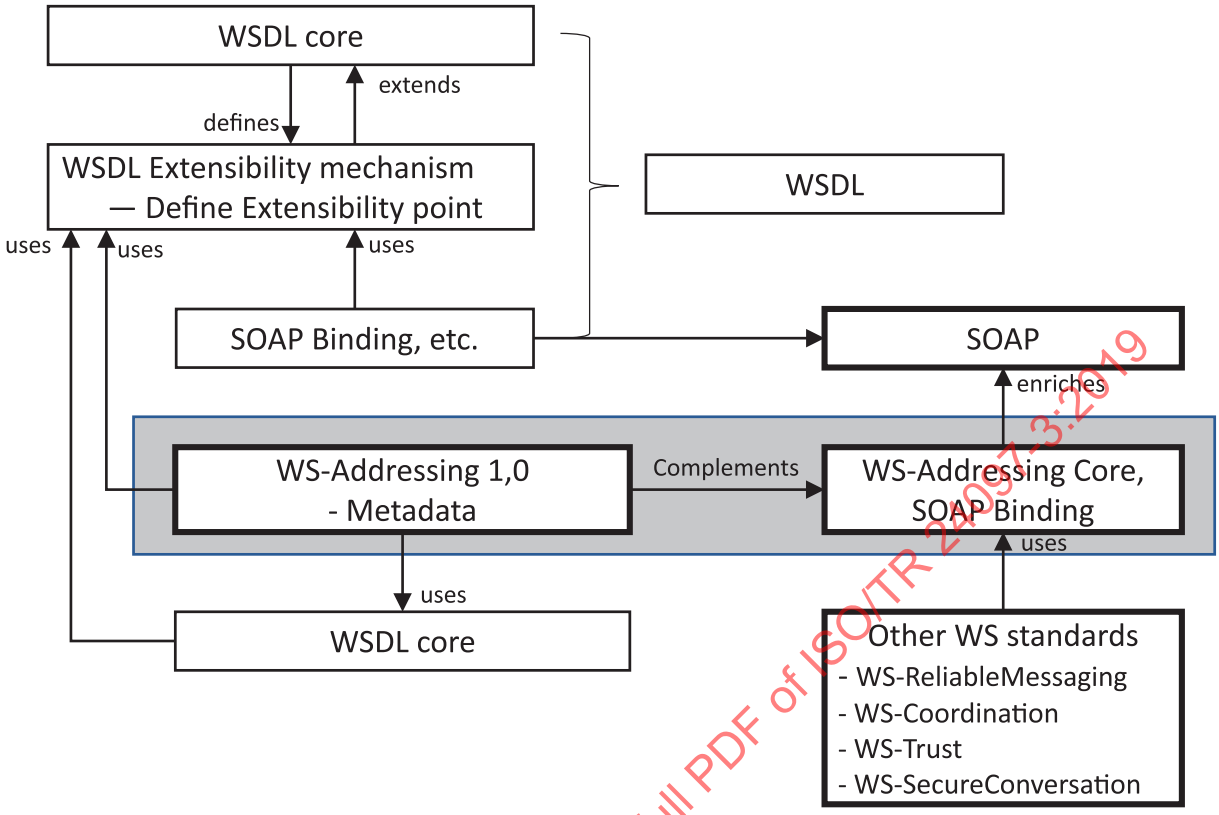


Figure 10 — The WS-Addressing standard and its role

9.1.2 WS-Addressing 1.0 — Core and Web Services Addressing 1.0 — SOAP Binding

The WS standards are constructed to be combinable. One standard does not cover all functions, but rather prepares an extensibility mechanism. WS-Addressing is an example of a SOAP extension.

SOAP is intentionally designed as a simple and lightweight messaging protocol. That is WSDL + SOAP cover web services' basic features.

The standards are still used even after WS-Addressing 1.0 — Metadata. So, we have to use the core and SOAP binding standard for effective use of the WS-Addressing.

The core and SOAP binding extend SOAP basic features:

— Endpoint reference (EPR), and

— Messaging addressing property (MAP) extension to deliver, route, identify, and correlate messages.

Pseudo schema of the EPR is as follows:

```
<wsa:EndpointReference>
    <wsa:Address>xs:anyURI</wsa:Address>
    <wsa:ReferenceParameters>xs:any*</wsa:ReferenceParameters> ?
    <wsa:Metadata>xs:any*</wsa:Metadata>?
</wsa:EndpointReference>
```

Also, pseudo schema of the MAP is as follows:

```
<wsa:To>xs:anyURI</wsa:To> ?
<wsa:From>wsa:EndpointReferenceType</wsa:From> ?
<wsa:ReplyTo>wsa:EndpointReferenceType</wsa:ReplyTo> ?
<wsa:FaultTo>wsa:EndpointReferenceType</wsa:FaultTo> ?
<wsa:Action>xs:anyURI</wsa:Action>
<wsa:MessageID>xs:anyURI</wsa:MessageID> ?
```

```
<wsa:RelatesTo RelationshipType="xs:anyURI"?>xs:anyURI</wsa:RelatesTo> *
<wsa:ReferenceParameters>xs:any*</wsa:ReferenceParameters> ?
```

Both pseudo schema are cited from "Web Services Addressing 1.0 — Core (2006-05-09)". The functionality of the constructs names may be self-descriptive.

9.1.3 WS-Addressing 1.0 — Metadata

This standard conforms to the WS-Policy (Web Services Policy 1.5 — Framework and Web Services Policy 1.5 — Attachment). Table 6 lists the whole constructs and their function.

Table 6 — Metadata constructs and their functions

Metadata constructs	Function
EPR relevant metadata	
`wsam:InterfaceName`	This QName identifies the set of endpoints at which a web service is deployed.
`wsam:ServiceName`	This QName identifies the set of endpoints at which a web service is deployed.
`wsam:Address`	This QName identifies WS-Addressing is required to communicate with the subject.
MAP relevant metadata	
`wsam:AnonymousResponse`	This QName identifies the endpoint requires request messages to use response endpoint EPRs that contain the anonymous URI (`"http://www.w3.org/2005/08/addressing/anonymous"`) as the value of [address]. In other words, the endpoint requires the use of anonymous responses. This is used as nested policy of `wsam:Address`.
`wsam:NonanonymousResponse`	This QName indicates that the endpoint expresses require request messages to use response endpoint EPRs that contain something other than the anonymous URI as the value of [address]. In other words, the endpoint requires the use of non-anonymous responses. This is also used as nested policy of `wsam:Address`.
`wsam:@Action`	This QName is used to explicitly define the value of the [action] property for messages in a WSDL description.

9.1.4 Elaboration of WS-AddressingMetadata

An example of the WS-AddressingMetadata is the realisation of an asynchronous response web service. Using an asynchronous response provides effective resource usage for both a web service consumer and a service provider, this is preferable when a service takes a long time. In addition, when we use a web service like a BPEL process, we can expect higher throughput for enterprise applications.

The following is an example:

```
<wsdl:definitions xmlns:wsdl = " http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsam = " http://www.w3.org/2007/05/addressing/metadata" …>
 …
 <wsdl:binding>
   <operation …>
     <wsp:Policy wsdl:required="true">
       <wsam:Addressing>
          <wsp:Policy>
             <wsam:NonAnonymousResponses/>
          </wsp:Policy>
       </wsam:Addressing>
     </wsp:Policy>
     <input …/>
     <output …/>
   </operation>
 </wsdl:binding>
```

```
    …
</wsdl:definitions>
```

NOTE Web Service Addressing 1.0 — Metadata (2007-09-04) describes that policy attachment subject is the service policy subject but it seems that the attachment policy subjects are service policy subject **and binding subject.**

9.2 WS-SecurityPolicy (WSSP)

Web services entail an interaction between a service program and a consumer program over an unsafe Internet. This process requires security for protecting both a service consumer and a service provider.

NIST 800-95 [29]describes that web services security should be based on several important concepts:

— **Identification and Authentication:** Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system,

— **Authorization.** The permission to use a computer resource, granted, directly or indirectly, by an application or system owner,

— **Integrity.** The property that data has not been altered in an unauthorized manner while in storage, during processing, or in transit,

— **Non-repudiation.** Assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the information,

— **Confidentiality**. Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information, and

— **Privacy**. Restricting access to subscriber or relying party information in accordance with Federal law and organization policy.

To realise these concepts, many web services security standards have been developed. Annex A provides a sample listing of these. For successful secure web services, two things are required: 1) ease of development and administration, and 2) interoperability of security solutions.

WS-SecurityPolicy is declarative (describes what we want to do) and eliminates details of how to develop web services security. The "how" is then provided by standards-based interoperable software tools. In addition, policy-based security will be published as a WSDL + QoS metadata, then a potential user can evaluate a service and realise interoperability.

9.2.1 WSSP standard

Web services security is an indispensable component for real world web services and a pillar of other security related standards such as WS-SecureConversation. The web services security standards are developed mainly by OASIS and some by W3C. Web service security is carried out only when both sides (client and service) support the same standard for a WS. For interoperability, WSSP is opened as a part of ***QoS*** metadata along with ***interface*** for a WS requestor. OASIS standardized the "WS-SecurityPolicy 1.2" in 2007 corresponding to the new W3C's WS-Policy 1.5 standardization. In 2009 WS-SecurityPolicy 1.3 was developed as a successor of WS-SecurityPolicy 1.2.

WSSP is a family language of WS-Policy, so its grammar is constructed in a consistent manner to WS-Policy.

WSSP specific vocabulary is defined using a prefix wssp.

Table 7 — WSSP standards, document URI and the Schema URI

Standard	Document URI	XML Schema URI
WS-SecurityPolicy 1.3[a] (Feb. 2009)	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.pdf	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy-1.3.xsd
Related document		**Zip file location**
OASIS, "WS-SecurityPolicy Examples" (included in WS-SX-ExamplesDocAndInteropMessages-cd-01-06.zip)		http://www.oasis-open.org/committees/documents.php?wg_abbrev=ws-sx

[a] WS-SecurityPolicy v1.2 was standardized in July 2007 before the completion of WS-Policy 1.5 standardization (September 2007). Therefore, using WS-SecurityPolicy 1.3 (February 2009) is preferable.

9.2.2 WSSP scope

WSSP is metadata for WSS, but it does not cover all of web services security (as shown in Annex A).

Per the WS-SecurityPolicy 1.3 scope, it covers WSS, WSSC (Web Services Secure Conversation), and WS-Trust. WS-Security is a part of information system security, namely it covers only wired data protection. Therefore, authorization (access control), XACML (eXtensible Access Control Markup Language) etc. are not included in this standard, i.e. other web services security standards are excluded (see Figure 11). It may be well understandable that this standard narrows down the communication options in order to ensure communication security.

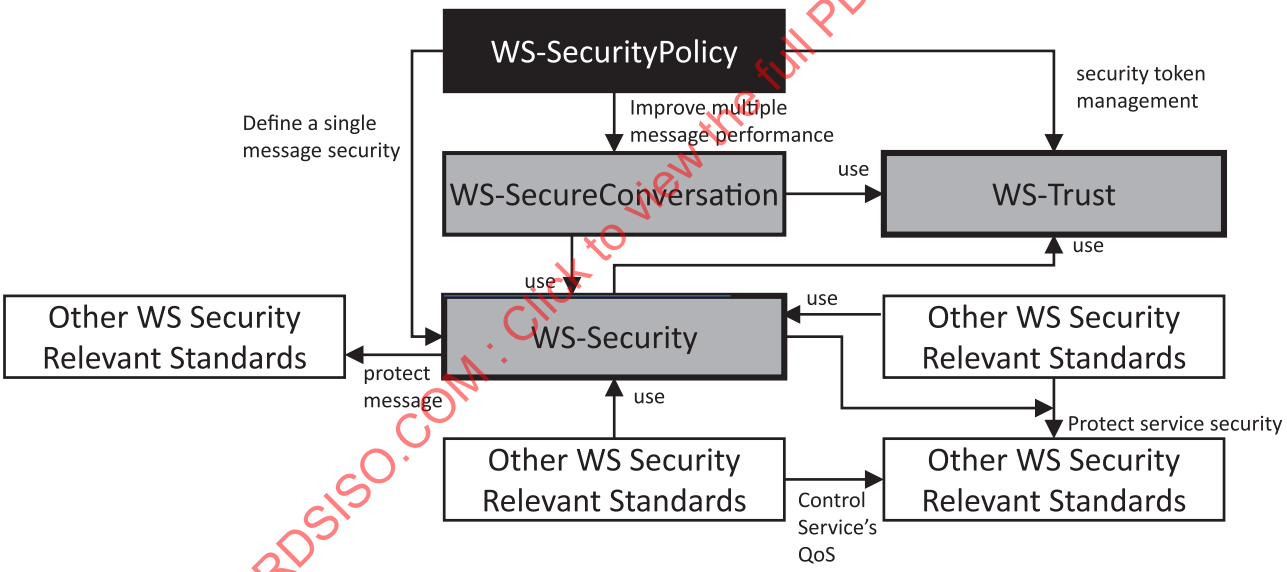


Figure 11 — WSSP scope

From a processing perspective, WSSP is represented as Figure 12.

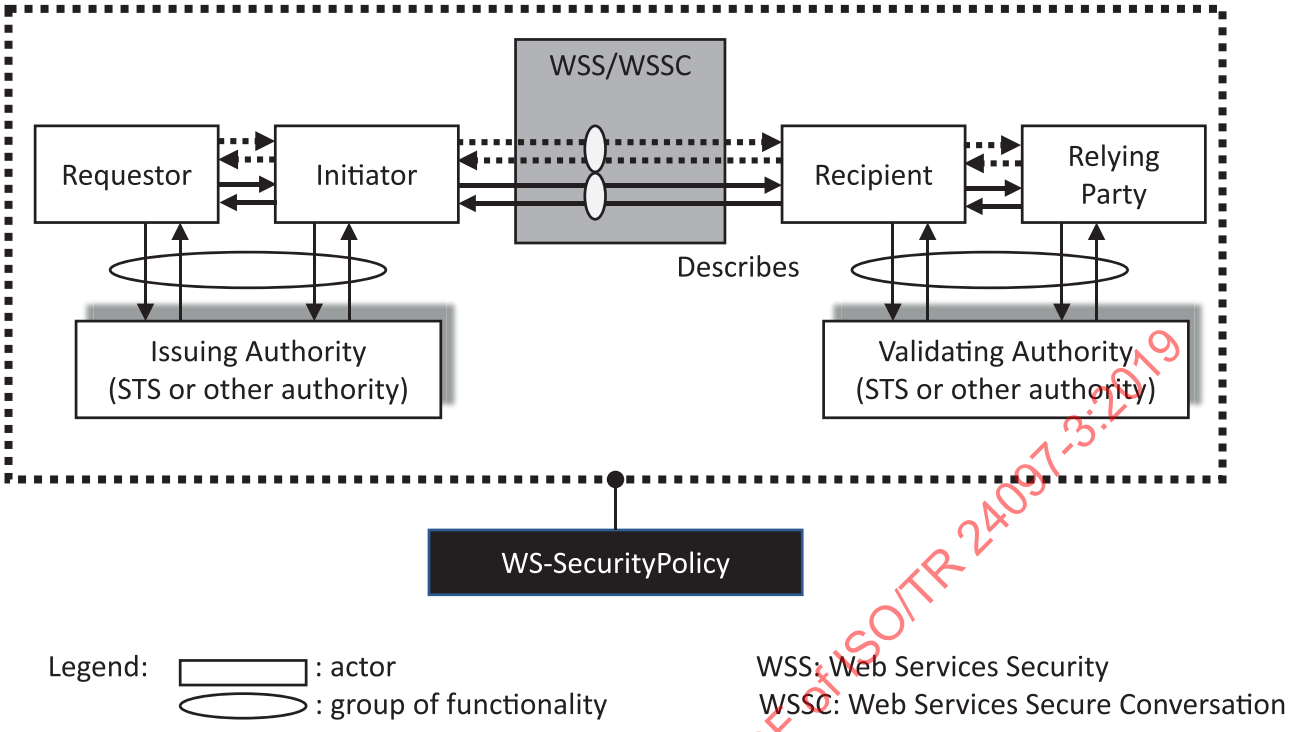


Figure 12 — WSSP reference model

In Figure 12, a thick dotted square represents the WSSP reference model. A web services security and/or web service secure conversation need(s) STS (Security Token Service). For this purpose WS-Trust is supported in WS-SecurityPolicy.

9.2.3 WS-SecurityPolicy fundamental

As described previously, the latest version of the WSSP is version 1.3. The version 1.3 schema imports the version 1.2 schema (see list below). The beginning part of the v1.3 is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--  … -->
   <xs:schema blockDefault="#all" elementFormDefault="qualified"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:wsa="http://www.w3.org/2005/08/addressing"
   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:tns="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802"
   targetNamespace="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802">

   <xs:import schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"
        namespace="http://www.w3.org/2005/08/addressing" />
   <xs:import
        schemaLocation=
   "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2.xsd"
        namespace="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702" />
   …
   </xs:schema>
```

Within the WS-SecurityPolicy v1.2 schema, there are over 100 global elements. For a web services designer or a developer this amount is a heavy burden. The WSSP categorised over one hundred assertions into some groups and show principal categories. Table 8 shows these.

Table 8 — Principal WSSP assertions

Main policy assertion	Description
Protection Assertions	what parts of a message are being secured
Conditional Assertions	general aspects or pre-conditions of the security
Security Binding Assertions	the security mechanism (Security Binding Assertions) that is used to provide the security
Supporting Token Assertions	the token types and usage patterns (Supporting Token Assertions) used to provide additional claims
WSS and Trust Assertions	token referencing and trust options (WSS and Trust Assertions)

When deciding on a WS policy, a service provider should take into consideration the user's requirements as well as their own service security. WS-Policy language supports many alternative descriptions and/or options. If a service provider presents an alternative or optional facility, the service provider supports both conditions.

9.2.4 Cryptographic algorithms and key length

After a WS security requirement analysis, we decide what kind of security is required for the WS, and then describe corresponding security policy.

WS security is built on the basis of four key technologies, namely "XML signature", "XML Encryption", "XML Canonicalization", and "ciphered data encoding".

When deciding security algorithms and key length, the following documents are helpful.

— Wikipedia, "Key size" (https://en.wikipedia.org/wiki/Key_size)

— NIST, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Length" (January 13, 2011), https://www.nist.gov/node/563406

— ENISA, "Algorithms, key size and parameters report — 2014" (November 2014), https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014

9.2.5 WSSP use case

In Clause 7 we introduced the idea of policy type categorization. OASIS Web Services Secure Exchange (WS-SX) Technical Committee produced the document "WS-SecurityPolicy Examples Version 1.0". It includes the WSSP example.

It is not easy to grasp the whole structure of the WSSP due to the number of security technologies that are continuously evolving. The WSSP attempts to cover a wide range of use cases by binding various necessary component security properties.

The WSSP categorizes the typical security usage patterns in three categories:

a) Transport layer security-based system,

b) Symmetric key based security system, and

c) Asymmetric key based security system.

It is not that one type of security system is superior to the others, rather the alternatives allow a user to select the security system that best fits the application requirement. Table 9 summarizes the security system characteristics.

Table 9 — WS-Security types and their features

WS security system type →	***Transport layer based security system***	***Symmetric key based security system***	***Asymmetric key based securit system***
System example	TLS based system	AES, 3DES based system. Kerberos	RSA based system
Binding name of WS-SecurityPolicy	Transport Binding	Symmetric Binding	Asymmetric Binding
Pseudo Schema description of WS-SecuritvPolicv *1	Sub-clause 7.3	Sub-clause 7.4	Sub-clause 7.5
Example description of WS-SecurityPolicy *1	Sub-clause C.1.1	Sub-clause C.2.1	Sub-clause C.3.1
Attaching policy subject	Endpoint	Endpoint/Operation	Endpoint/Operation
Security element realization method			
Identification and Authentication	Trusted party's digital certificate (mutual authentication possible)	{user name, password}, Kerberos ticket, SAML token	Trusted party's digital certificate
Authorization	not applicable	SAML token may be used for authorization decision	SAML token may be used for authorization decision
Integrity	Message Authentication Code (MAC)	Message Authentication Code (MAC)	Digital signature
Non-repudiation	Unrealizable	Unrealizable	Digital signature
Confidentiality	Encryption	Encryption	Encryption
Availability	Timestamp	Timestamp	Timestamp
Privacy protection	Not cryptographic matter (Business application design and management)	Not cryptographic matter (Business application design and management)	Not cryptographic matter (Business application design and management)
Pros and cons	Does not work with intermediaries	Needs secure key distribution method	Takes the most processing time

TLS: Transport Layer Security, SAML:
*1: Sub-clause of WS-SecurityPolicy 1.3

9.2.5.1 Effective and Secure multiple messages exchange web service realization

WS-Security provides a base of a secure web services. Its scope is a single SOAP message. SOAP is a stateless messaging protocol. Therefore, if a web service needs multiple messages exchanged, a different secure and effective method is required that extends WS-Security. The WS-SecureConversation standard addresses this need.

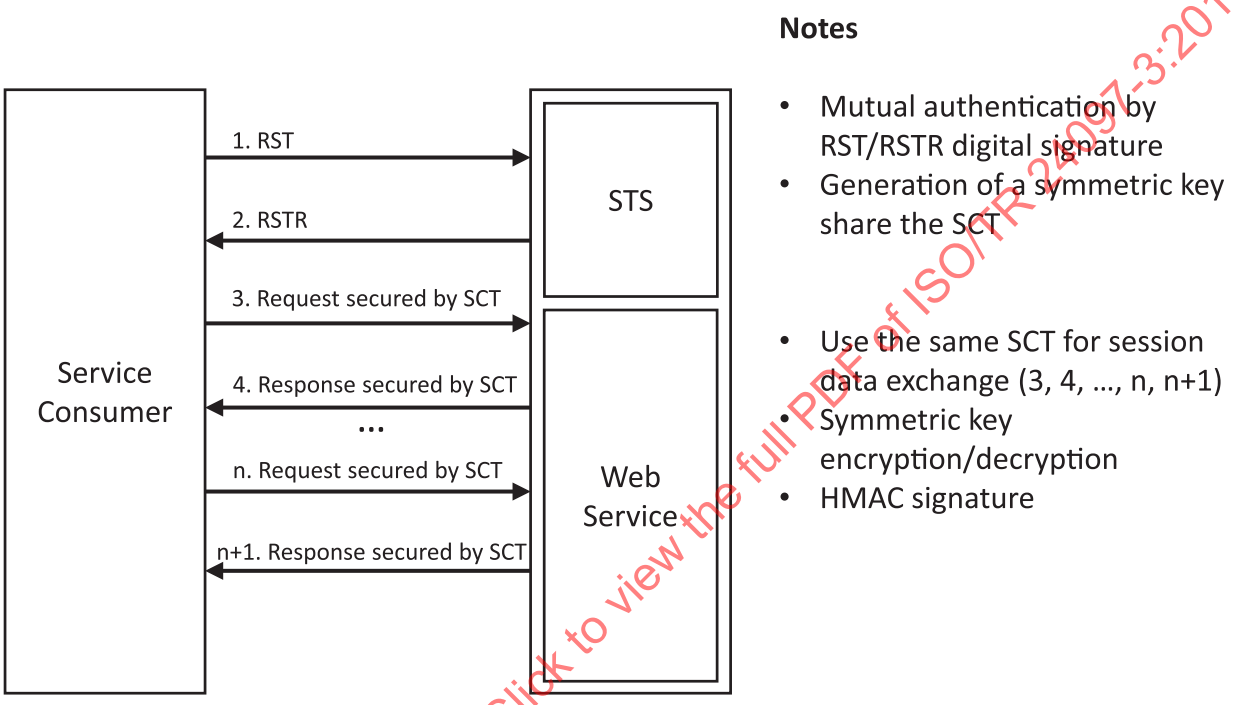
The term conversation means the series of exchanges from the initial message to the terminating message. The term is a similar concept of the HTTP session. WS-SecureConversation defines the SecurityContextToken (SCT) that secures the web services through conversation.

The Security Token Service (STS) is a web service. This supports:

— issuing, renewing, and cancelling of secure conversation tokens;

— allowing symmetric encryption to eliminate the overhead that asymmetric encryption adds.

STS may collocate in the web service endpoint or may be as a trusted third party. Figure 13 shows how a secure conversation is processed.



RST: Request Security Token, RSTR: Request Security Token Response, SCT: Secure Contect Token, STS: Secure Token Service, HMAC: Hash-based Message Authentication Code

Figure 13 — Secure web service conversation

The service consumer first sends RST to STS to issue a SecureConversationToken. STS authenticates the service consumer by its digital signature. If STS acknowledges the requestor as a valid user, it issues RSTR with its digital signature. The service consumer authenticates the web service. Through this process, mutual authentication is achieved. This process is like TLS mutual authentication.

If mutual authentication has succeeded, a symmetric ephemeral key is generated based on SCT. While symmetric key signature is generally considered imperfect authentication, in this case symmetric key is generated with SCT that is based on mutual authentication. Therefore, it is considered equivalent to asymmetric key signature.

Using symmetric key, data exchanges become more effective than asymmetric key data exchange.

9.2.5.2 WS-SecureConversation policy example

The following example targets the security policy of a multiple message exchange web service. Fundamental assumptions are:

— use of secure conversation;

— both parties have a X.509 certificate. Mutual authentication will be done by referring their private key signature;

— after success of mutual authentication, an ephemeral symmetric key will be created;

— as a secure messaging we adopt "Basic 256" algorithm, i.e. Sha1 for signature and AES256 for encryption, respectively.

The security policy consists of one endpoint subject policy (WSS10SecureConversation_policy) and two message subject policies (attached wsdl:input and wsdl:output respectively).

The following listing is quoted from OASIS's document, "WS-SecurityPolicy Examples Version 1.0", 4 November 2010".

```
(P001)    <wsp:Policy wsu:Id="WSS10SecureConversation_policy">
(P002)      <wsp:ExactlyOne>
(P003)        <wsp:All>
(P004)          <sp:SymmetricBinding
                  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
(P005)            <wsp:Policy>
(P006)              <sp:ProtectionToken>
(P007)                <wsp:Policy>
(P008)                  <sp:SecureConversationToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/
AlwaysToRecipient">
(P009)                    <wsp:Policy>
(P010)                      <sp:RequireDerivedKeys/>
(P011)                      <sp:BootstrapPolicy>
(P012)                        <wsp:Policy>
(P013)                          <wsp:ExactlyOne>
(P014)                            <wsp:All>
(P015)                              <sp:AsymmetricBinding
                                      xmlns:sp="http://docs.oasis-open.org/ws-sx/
ws-securitypolicy/200702">
(P016)                                <wsp:Policy>
(P017)                                  <sp:InitiatorToken>
(P018)                                    <wsp:Policy>
(P019)                                      <sp:X509Token
                                              sp:IncludeToken=
 "http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/AlwaysToRecipient">
(P020)                                        <wsp:Policy>
(P021)                                          <sp:WssX509V3Token10/>
(P022)                                        </wsp:Policy>
(P023)                                      </sp:X509Token>
(P024)                                    </wsp:Policy>
(P025)                                  </sp:InitiatorToken>
(P026)                                  <sp:RecipientToken>
(P027)                                    <wsp:Policy>
(P028)                                      <sp:X509Token
                                             sp:IncludeToken=
    "http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/AlwaysToInitiator">
(P029)                                        <wsp:Policy>
(P030)                                          <sp:WssX509V3Token10/>
(P031)                                        </wsp:Policy>
(P032)                                      </sp:X509Token>
(P033)                                    </wsp:Policy>
(P034)                                  </sp:RecipientToken>
(P035)                                  <sp:AlgorithmSuite>
(P036)                                    <wsp:Policy>
(P037)                                      <sp:TripleDesRsa15/>
(P038)                                    </wsp:Policy>
(P039)                                  </sp:AlgorithmSuite>
(P040)                                  <sp:Layout>
(P041)                                    <wsp:Policy>
(P042)                                      <sp:Strict/>
(P043)                                    </wsp:Policy>
(P044)                                  </sp:Layout>
(P045)                                  <sp:IncludeTimestamp/>
(P046)                                  <sp:OnlySignEntireHeadersAndBody/>
```

```
(P047)                              </wsp:Policy>
(P048)                            </sp:AsymmetricBinding>
(P049)                            <sp:Wss10>
(P050)                              <wsp:Policy>
(P051)                                <sp:MustSupportRefKeyIdentifier/>
(P052)                              </wsp:Policy>
(P053)                            </sp:Wss10>
(P054)                            <sp:SignedParts>
(P055)                              <sp:Body/>
(P056)                              <sp:Header Name="Action"
                                      Namespace="http://www.w3.org/2005/08/addressing"/>
(P057)                            </sp:SignedParts>
(P058)                            <sp:EncryptedParts>
(P059)                              <sp:Body/>
(P060)                            </sp:EncryptedParts>
(P061)                          </wsp:All>
(P062)                        </wsp:ExactlyOne>
(P063)                      </wsp:Policy>
(P064)                    </sp:BootstrapPolicy>
(P065)                  </wsp:Policy>
(P066)                </sp:SecureConversationToken>
(P067)              </wsp:Policy>
(P068)            </sp:ProtectionToken>
(P069)            <sp:AlgorithmSuite>
(P070)              <wsp:Policy>
(P071)                <sp:Basic256/>
(P072)              </wsp:Policy>
(P073)            </sp:AlgorithmSuite>
(P074)            <sp:Layout>
(P075)              <wsp:Policy>
(P076)                <sp:Strict/>
(P077)              </wsp:Policy>
(P078)            </sp:Layout>
(P079)            <sp:IncludeTimestamp/>
(P080)            <sp:OnlySignEntireHeadersAndBody/>
(P081)          </wsp:Policy>
(P082)        </sp:SymmetricBinding>
(P083)        <sp:Trust13>
(P084)          <wsp:Policy>
(P085)            <sp:RequireClientEntropy/>
(P086)            <sp:RequireServerEntropy/>
(P087)          </wsp:Policy>
(P088)        </sp:Trust13>
(P089)
(P090)      </wsp:All>
(P091)    </wsp:ExactlyOne>
(P092)  </wsp:Policy>

(P093)  <wsp:Policy wsu:Id="WSS10SecureConversation_input_policy">
(P094)    <wsp:ExactlyOne>
(P095)      <wsp:All>
(P096)        <sp:SignedParts>
(P097)          <sp:Header Name="Action"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P098)          <sp:Header Name="To"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P099)          <sp:Header Name="MessageID"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P0100)   <sp:Body/>
(P0101) </sp:SignedParts>
          </wsp:All>
(P0102)  </wsp:ExactlyOne>
(P0103) </wsp:Policy>
(P0104) <wsp:Policy wsu:Id="WSS10SecureConversation_output_policy">
(P0105)   <wsp:ExactlyOne>
(P0106)     <wsp:All>
(P0107)       <sp:SignedParts>
(P0108)         <sp:Header Name="Action"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P0109)         <sp:Header Name="To"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
```

```
(P0110)          <sp:Header Name="MessageID"
                            Namespace="http://www.w3.org/2005/08/addressing"/>
(P0111)          <sp:Header Name="RelatesTo"
                            Namespace="http://www.w3.org/2005/08/addressing"/>
(P0112)          <sp:Body/>
(P0113)        </sp:SignedParts>
(P0114)      </wsp:All>
(P0115)    </wsp:ExactlyOne>
(P0116) </wsp:Policy>
```

9.2.6 Validation of WS-SecurityPolicy document

A valid WS-SecurityPolicy document is required. We can generate and validate WS-SecurityPolicy descriptions using the WS-SecurityPolicy schema. It can be downloaded from `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.3.xsd`.

NOTE The software vendor can provide specific tools. For example, it can provide a JAX-WS based tool. Giving annotation to Java program, the tool generates corresponding WSDL and policy and support to publish it.

9.3 Web Services Reliable Messaging Policy Assertion

When sending multiple messages on the Internet, a quality assurance policy as a set is required. In WS — RMPolicy, the quality is assured with a combination of AtLeastOnce, AtMostOnce, ExactlyOnce, InOrder.

Almost all multiple message exchanges need reliability. The Web Services Reliable Messaging Policy Assertion defines how to apply the Web Services Reliable Messaging standard to secure reliable messaging.

In this protocol, a message sent from a source endpoint (or client Web service) to a destination endpoint (or Web service) is guaranteed either to be delivered according to one or more delivery assurances, or to raise an exception if an error occurs. Figure 14 indicates reliable messaging concept [quoted from "Web Services Reliable Messaging (WSReliableMessaging) Version 1.2"].

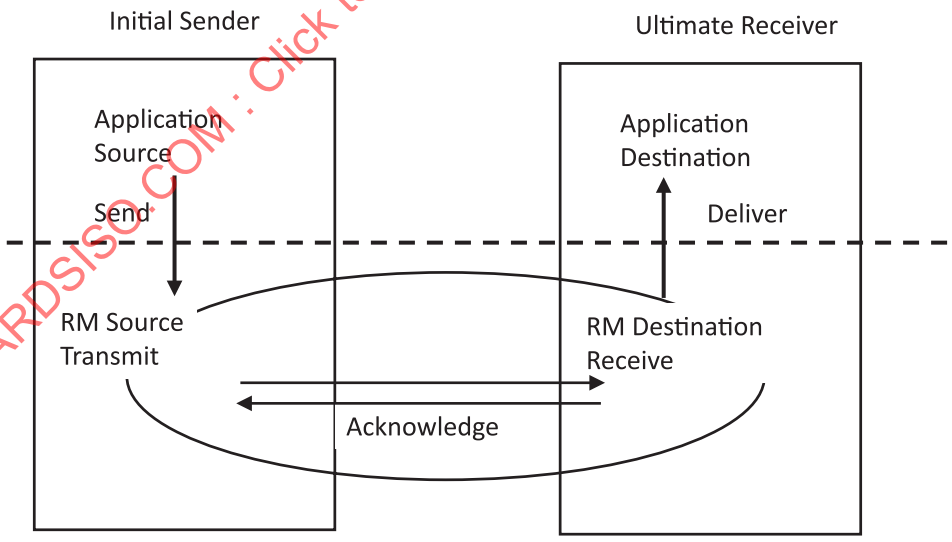


Figure 14 — Concept of reliable messaging

The reliable messaging standards are listed in Table 10.

Table 10 — Reliable message related standards

Standard	Document URl	XML Schema URl
Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.2	http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.2-spec-os.pdf	http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-schema-200702.xsd
Web Services Reliable Messaging (WSReliableMessaging) Version 1.2	http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.pdf	http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-schema-200702.xsd
Reliable Secure Profile Version 1.0 Final Material 2010-11-09 (WSI)	http://www.ws-i.org/Profiles/ReliableSecureProfile-1.0-2010-11-09.html	—

9.3.1 RM Policy Assertions

The following list gives the pseudo RM Policy Assertion schema:

```
<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>
```

The WS-ReliableMessaging specification is aware of security threats in a sequence, i.e. integrity threats or sequence hijacking, and therefore proposes countermeasures to address these problems. The `[<wsrmp:SequenceSTR> | <wsrmp:SequenceTransportSecurity>]?` indicates the selection of countermeasures. The `<wsrmp:SequenceSTR>` specified in runtime will use the `wsse:SecurityTokenReference` that is referenced in the `CreateSequence` message. On the other hand, the `<wsrmp:SequenceTransportSecurity>` specifies that the runtime will use the SSL/TLS transport session. This assertion is used in conjunction with the `sp:TransportBinding` assertion (for example, `sp:HttpsToken`). We can only specify one security assertion, i.e. we can specify `wsrmp:SequenceSTR` or `wsrmp:SequenceTransportSecurity`, but not both.

As described in ISO 24097-1, there are four types of messaging quality requirement; AtLeastOnce, AtMostOnce, ExactlyOnce, and InOrder.

```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <wsrmp:RMAssertion
          xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702">
        <wsrmp:SequenceSTR/>
        <wsrmp:DeliveryAssurance>
          <wsp:Policy>
            <wsrmp:ExactlyOnce/>
            <wsrmp:InOrder/>
          </wsp:Policy>
        </wsrmp:DeliveryAssurance>
    </wsrmp:RMAssertion>
</wsp:Policy>
```

9.4 *MTOM policy* (MTOM Serialization Policy Assertion 1.1)

SOAP Message Transmission Optimization Mechanism (MTOM) specifies an optimized method for sending binary data as part of a SOAP message.

"MTOM Serialization Policy Assertion 1.1" reached W3C Working Draft stage on 18 September 2007, but it is not a full W3C Recommendation. The W3C homepage describes its status as follows: 2007-09-18 MTOM Serialization Policy Assertion 1.1 Retired (`http://www.w3.org/TR/#tr_SOAP`).

However, JSR 224 (Java™ API for XML-Based Web Services (JAX-WS) 2.0) supports this feature based on "SOAP Message Transmission Optimization Mechanism. Recommendation, W3C, January 2005. `http://www.w3.org/TR/soap12-mtom/`.

Further information about applying MTOM is available in JSR 224.

Reading documents about the effect of MTOM before applying it is recommended.

9.5 *SOAP usage policy* (Web Services SOAP Assertions)

This policy declares SOAP message usage.

Syntax

SOAP 1.1 use

```
<wssa:SOAP11 ...> ... <wssa:SOAP11>
```

SOAP 1.2 use

```
<wssa:SOAP12 ...> ... <wssa:SOAP12>
```

EXAMPLE

```
<wsp:Policy>
<wsp:ExactlyOne>
<wssa:SOAP11/>
<wssa:SOAP12/>
</wsp:ExactlyOne>
</wsp:Policy
```

Policy Attachment Subject

```
wsdl:port
```

10 Metadata versioning

Version updates is a normal practice in the software world. It is the result of the service evolution. Evolving services in a backward compatible manner (i.e. Type 2 in Figure 15) is recommended for consumer's service continuity.

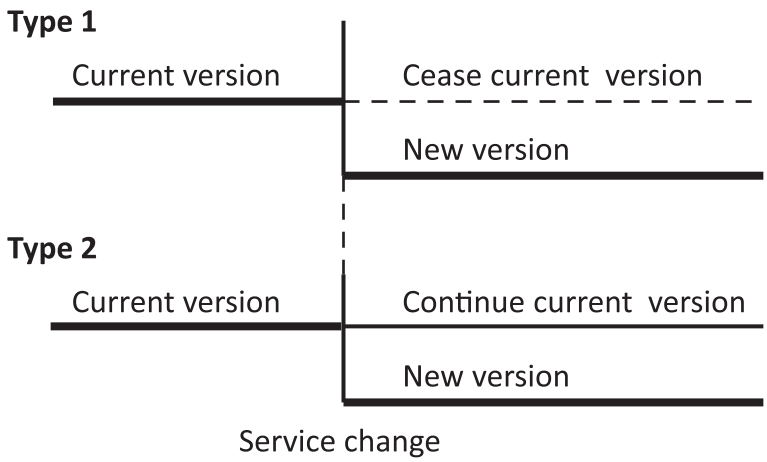


Figure 15 — Recommended service evolution manner

ISO 24097-1 describes versioning rules as follows:

— The version number is attached to WSDL and QoS metadata independently.

— The version is represented by the following common form in both WSDL and QoS metadata:

  Form: m.n.a

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

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

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

— Version change rule:

  — m: the major version number is changed to m+1 when the change from the previous version of the service will cause existing service to fail;

  — n: the minor version number is changed to n+1 when the change to the service will result in all existing services continuing to validate. However, some new service will fail against the old version, and

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

The following is an example.

We assume the WSDL definitions schema is changed as follows:

```
<wsdl:definitions name="sample1" serviceVersion="1.0" interfaceVersion="1.0" …>
…

  <wsp:Policy policyVersion = "1.0" >

    <wsam:Addressing AddressingMetadataVersion = "1.0" wsp:optional="true"/>
    …
  </wsp:policy>
</wsdl:definitions>
```

If we change the `wsam:Addressing` option to false (this means `wsam:Addressing` mandatory), some users who do not use the Addressing may experience an execution fault. Thus, by the version change rule, we update AddressingMetadataVersion to 2.0.

The following is a change result of `wsdl:definitions`.

```
<wsdl:definitions name="sample1" serviceVersion="2.0" interfaceVersion="1.0" …>
…

  <wsp:Policy policyVersion = "2.0" >

    <wsam:Addressing AddressingMetadataVersion = "2.0" wsp:optional="false"/>
    …
  </wsp:policy>
</wsdl:definitions>
```

11 Security considerations

Every QoS standard should have policies and assertions signed to prevent tampering.

It is recommended that policies should not be accepted unless they are signed and have an associated security token that specifies the signer has proper claims for the given policy. A relying party should not rely on a policy unless the policy is signed and presented with sufficient claims to pass the relying parties acceptance criteria.