# Web Services Policy 1.5 - Guidelines for Policy Assertion Authors

## Editors' copy $Date: 2006/11/01 02:40:42 $ @@ @@@@ @@@@

**This version:**
ws-policy-guidelines.html
**Latest version:**
http://dev.w3.org/cvsweb/~checkout~/2006/ws/policy/ws-policy-guidelines.html?content-type=text/html;charset=utf-8
**Editors:**
Maryann Hondo, IBM Corporation
Umit Yalcinalp, SAP AG.



---

## Abstract

*Web Services Policy 1.5 - Guidelines for Policy Assertion Authors* is a guideline for assertion authors that will work with the Web Services Policy 1.5 - Framework [*Web Services Policy Framework*] and Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment*] specifications to create domain specific assertions. The focus of this document is to provide best practices and patterns to follow as well as illustrate the care needed in using WS-Policy to achieve the best possible results for interoperability. It is a complementary guide to using the specifications.

## Status of this Document

**This document is an editors' copy that has no official standing.**

---

## Table of Contents

## Appendices



---

# 1. Introduction

WS-Policy specification defines a policy to be a collection of policy alternatives, where each policy alternative is a collection of policy assertions. *Web Services Policy 1.5 - Guidelines for Policy Assertion Authors* is a resource primarily for assertion authors that provides guidelines on the use of Web Services Policy 1.5 - Framework and Web

Services Policy 1.5 - Attachment specifications to create and use domain specific assertions to enable interoperability.

WS-Policy Assertions are XML expressions that communicate the requirements and capabilities of a web service by adhering to the specification, WS-Policy Framework. It was recognized that in order to enable interoperability of web services, different sets of WS-Policy Assertions needed to be defined by different communities based upon the requirements of the web service.

Some policy assertions specify traditional requirements and capabilities that will ultimately manifest on the wire (e.g., authentication scheme, transport protocol selection). Other policy assertions have no wire manifestation yet are critical to proper service selection and usage (e.g., privacy policy, QoS characteristics). WS-Policy provides a single policy grammar to allow both kinds of assertions to be reasoned about in a consistent manner.

The focus of these guidelines is to capture best practices and usage patterns for practitioners to follow. It is a complementary guide to the Framework and Attachments specifications and primer. It is intended to provide non-normative guidelines for:

- WS-Policy expression authors who need to understand the syntax of the language and understand how to build consistent policy expressions,
- WS-Policy assertion authors who need to know the features of the language and understand the requirements for describing policy assertions.
- Consumers of policy expressions who need to understand the requirements contained in policy assertions,
- Providers of policy expressions who need to understand how to use the assertions authored by policy assertion authors

This document assumes a basic understanding of XML 1.0, Namespaces in XML, WSDL 1.1 and SOAP.

This is a non-normative document and does not provide a definitive specification of the Web Services Policy framework. **B. XML Namespaces** lists all the namespace prefixes that are used in this document. (XML elements without a namespace prefix are from the Web Services Policy XML Namespace.)

## 2. Basic Concepts: What is an Assertion

An assertion is a piece of metadata that describes a capability of a specific WS-Policy domain. Sets of assertions are typically defined in a dedicated specification that describe their semantics, applicability and scoping requirements as well as their data type definition using XML Schema.

There are already many examples in the industry that adhere to this practice, such as *Web Services Reliable Messaging Policy* and *WS-SecurityPolicy*.

Some common characteristics from these documents may be considered as *best practices* for new assertion authors:

- Specify both the syntax and the semantics of the assertions
- If nested or parameterized assertions are defined, be clear about their usage

**Comment [asv1]:** What is an xml expression? Suggest cutting the phrase 'are XML expressions that'.

**Comment [asv2]:** Where are these usage patterns?

**Comment [asv3]:** Shouldn't this be policy assertion authors?

**Comment [asv4]:** ISSUE 3982 - This document is for policy assertion authors. Can't think of a reason why the introduction should identify other target audiences.

**Comment [asv5]:** ISSUE 3980 - Why wouldn't we expect the assertion authors to be familiar with WS-Policy and WS-PolicyAttachment? At a minimum, we should assume that the assertion authors have read the WS-Policy Primer.

**Comment [asv6]:** This section covers guidelines based on existing practice. Why is this section called 'What is an Assertion'?

**Comment [asv7]:** What practice? (Missing antecedent)

**Comment [asv8]:** What documents? (Missing antecedent)

- Describe the policy subjects the assertions can be attached to.

In this document we will explain why these practices should be followed so that the assertion developers defining such a specification will be well informed and able to adequately specify assertions for their domain.

It is expected that consumers of the metadata specified by the assertion authors will also benefit from understanding these practices as it will help them utilize the assertions in the context of the WS-Policy framework. A result of following the best practices will be an assertion specification that describes a contract for the consumers and providers of the capabilities and constraints of the domain.

## 2.1 Roles and Responsibilities in Utilizing Policy Assertions

In order for the policy framework to enable communities to express their own domain knowledge, it is necessary to provide basic functionality that all domains could exploit and then allow points of extension where authors of the various WS-Policy expressions for a particular domain can provide additional semantics.

Below we capture some of the characteristics of the roles and responsibilities for the authors, consumers and providers.

### 2.1.1 WS-Policy Authors

WS-Policy Domain owners or WS-Policy authors are defined by the WS-Policy Framework to be a community that chooses to exploit the WS-Policy Framework by creating their own specification to define a set of assertions that express the capabilities and constraints of that target domain. The WS-Policy Framework is based on a declarative model, meaning that it is incumbent on the WS-Policy authors to define both the semantics of the assertions as well as the scope of their target domain in their specification. The set of metadata for any particular domain will vary in the granularity of assertion specification required. It is the intent of this document to help communities utilize the framework in such a way that multiple WS-Policy domains can co-exist and consumers and providers can utilize the framework consistently across domains.

In order to take advantage of the WS-Policy Framework, any domain author attempting to define new WS-Policy assertions must adhere to the MUST's and SHOULD's in the specifications and should review the conformance section of the specification.

WS-Policy Domain authors must also specify how to associate the assertions they have defined with the policy subjects identified by the WS-PolicyAttachment specification

An example of a domain specification that includes these properties can be seen in the WS-SecurityPolicy specification [*WS-SecurityPolicy*]. The WS-SecurityPolicy authors have defined their scope as follows:

*"This document [WS-SecurityPolicy] defines a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in WSS: SOAP Message Security, WS-Trust and WS-SecureConversation. This document takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and*

**Comment [asv9]:** Can't see a reason why this document shouldn't assume that the framework and multiple attachment mechanisms exist today.

**Comment [asv10]:** Shouldn't this be assertions?

**Comment [asv11]:** Who are the WS-Policy domain owners?

**Comment [asv12]:** Not aware of any such definition. Can't think of a reason why this term should be defined either.

**Comment [asv13]:** Can't parse.

**Comment [asv14]:** A policy attachment mechanism defines how to associate policy expressions with policy subjects. Why would a policy assertion author go beyond identifying one or more policy subjects for an assertion?

*mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages."*

An example of scoping individual assertions to policy subjects is also provided by the WS-Security Policy specification in Appendix A.

### 2.1.2 Consumers

A consumer of WS-Policy Assertions can be any entity that is capable of parsing a WS-Policy xml element, and selecting one alternative from the policy, that it then uses to govern the creation of a message to send to the subject to which the policy alternative was attached. The WS-Policy Attachment specification defines a set of attachment models for use with common web service subjects: WSDL definitions [*WSDL 1.1*, *WSDL 2.0 Core Language*], UDDI directory entries [*UDDI API 2.0*, *UDDI Data Structure 2.0*, *UDDI 3.0*], and WS-Addressing Endpoint References (EPR) [*WS-Addressing Core*].

In the degenerate case, a human could read the xml and determine if a message could be constructed conformant to the advertised policy.

It is expected that consumers of WS-Policy will include a wide range of client configurations, from stand alone client applications to "active" web service requestors that are capable of adapting to the constraints and capabilities expressed in a WS-Policy document and modifying their own configurations dynamically.

### 2.1.3 Providers

A provider of WS-Policy Assertions can be any web service implementation that can specify its' on-the-wire message behavior as an XML expression that conforms to the WS-PolicyFramework and WS-Policy Attachment specifications. The WS-PolicyAttachment specification has defined a set of subjects and an extensible mechanism for attaching policies to web services subjects. When a web service provider chooses to make its capabilities and constraints available, it may also need to conform to requirements of other policy specifications it utilizes ( i.e., WS-SecurityPolicy).

When deploying services with policies it is uesful for providers to anticipate how to evolve their services capabilities over time. If forward compatibility is a concern in order to accommodate compatibility with different and potentially new clients, providers should refer to **5.9 Lifecycle of Assertions** and *Web Services Policy Primer* that describes service and policy assertion evolution.

# 3. General Guidelines for WS-Policy Assertion Authors

## 3.1 Assertions and Their Target Use

**Comment [asv15]:** What is the best practice behind the quoted example?

**Comment [asv16]:** Why should the consumer role be limited to policy expression parsers?

**Comment [asv17]:** Why should the consumer role be limited to an entity that creates wire messages?

**Comment [asv18]:** WS-PolicyAttachment doesn't define an attachment mechanism for EPRs.

**Comment [asv19]:** What is the value added by this sentence? Suggest dropping this sentence.

**Comment [asv20]:** Don't understand what does the phrase 'provider of WS-Policy Assertions' mean.

**Comment [asv21]:** Shouldn't this be a policy expression?

**Comment [asv22]:** Shouldn't we use the official titles?

**Comment [asv23]:** Who does the 'it' refers to?

**Comment [asv24]:** What is the basis for this 'conform' statement? As a principle, we should cross reference normative text in WS-Policy, WS-PolicyAttachment and relevant assertion specifications, as appropriate.

**Comment [asv25]:** Other policy specifications or policy assertion specifications?

**Comment [asv26]:** Does the phrase 'lifecycle of assertions' mean 'versioning policy assertions'? If so, why shouldn't we use the phrase 'versioning policy assertions'?

WS-Policy authors need to have some definition of what the goal is for the assertions they author. WS-Policy authors should also understand the functionality the WS-Policy framework provides and apply the knowledge of framework processing when defining the set of assertions.

Assertions can be simple or they can be complex. WS-Policy authors may choose to specify one or two assertions or they may choose to specify many assertion elements that require parameters or other nested assertions. There are advantages to simplifying the set of assertions. The ultimate goal of policy assertions is to enable interoperability and assertions that define behavior for consumers and providers that can be clearly understood will most likely be consumed by a broad audience.

If a domain has a wide range of behaviors, the ability to combine individual assertions may also need to be considered.

The number of different subjects to which an assertion can be attached is also a factor when defining an assertion. Determining the appropriate policy subjects can sometimes involve understanding the requirements of wide range of client configurations, from stand alone client applications to "active" web service requestors that are capable of modifying their own configurations dynamically.

Once the range of policy subjects are identified, there are choices for ways of attaching multiple instances of a simple policy assertion to multiple subjects. One way is to utilize the referencing mechanism that is present in the framework. By defining the assertion once and using it in different policies (e.g. with different alternatives) via reference, a policy assertion may be associated with different alternatives and subjects. A referencing mechanism is very useful in a tooling environment; when creating a domain specific attachment in multiple WSDL files or Endpoint References in which the same set of policies are expected to be applied.

In summary, WS-Policy domain authors should consider the following factors when composing the set of domain assertions as it may make sense to factor out some assertions that can be used by multiple subjects:

- *The definition of the assertion*. Does the assertion pertain to a specific behavior that is tied to a context or is the behavior different in each possible context? For example an assertion that may have different semantics for a single message exchange or for all message exchanges that pertain to an endpoint would probably be represented by separate assertions.
- *Scoping of the assertion*. Where does the assertion apply? Is the assertion about a specific description in WSDL? Is the assertion about a specific aspect of protocol? Is the assertion about a specific coordination between parties? Determination of the subject of an assertion is helpful in specifying the different scopes and subjects that it applies to.
- *Composition*. If the assertion is used with other assertions in a context, it is necessary to consider how the assertion may or may not affect the composition. For example, if an assertion applies to the SOAP protocol, it would be necessary to consider how its presence must interact with other policy assertions that are defined for security.

**Comment [asv27]:** There seems to be a target audience confusion: policy author or policy assertion author?

**Comment [asv28]:** What are these advantages? Here is one: simple assertion may simplify an assertion implementation.

**Comment [asv29]:** Why should a policy assertion author worry about client configurations, standalone client applications or active clients? Suggest dropping this sentence.

**Comment [asv30]:** Can't think of a reason why this is relevant to policy assertion design. Suggest dropping this sentence.

We will cover these aspects in more detail with respect to the type of assertions being used in the subsequent sections.

## 4. Authoring Styles

Comment [asv31]: ISSUE 3981 - Section 4 is irrelevant to policy assertion design.

WS-Policy supports 2 different authoring styles. A compact form is one in which an expression consists of three constructs: an attribute to decorate an assertion (to indicate whether it is required or optional), semantics for recursively nested policy operators, and a policy reference/inclusion mechanism.

```
<wsp:Policy xmlns:wsp='http://www.w3.org/@@@@/@@/ws-policy'
xmlns:sp='http://schemas.xmlsoap.org/ws/2005/07/securitypolicy'

                  xmlns:wsrmp='http://docs.oasis-open.org/ws-
rx/wsrmp/200608'>

 <wsrmp:RMAssertion wsp:Optional="true"/>

 <wsp:ExactlyOne>

  <wsp:All>

   <sp:TransportBinding>

    <wsp:Policy>

     <sp:TransportToken>

      <wsp:Policy>

       <sp:HttpsToken RequireClientCertificate='xs:boolean' />

      </wsp:Policy>

     </sp:TransportToken>

   </sp:TransportBinding>

  </wsp:All>

 </wsp:ExactlyOne>

</wsp:Policy>
```

An alternative style is a "normal form" in which the optional attribute is replaced by the expression of the alternatives allowed by the set of policy assertions.

Comment [asv32]: This statement is incorrect. Policy normalization is more than processing wsp:Optional attributes.

```
<wsp:Policy xmlns:wsp='http://www.w3.org/@@@@/@@/ws-policy'
xmlns:sp='http://schemas.xmlsoap.org/ws/2005/07/securitypolicy'

 xmlns:wsrmp='http://docs.oasis-open.org/ws-rx/wsrmp/200608'>

 <wsp:ExactlyOne>
```

```
 <wsp:All>


  <wsrmp:RMAssertion>
     <sp:TransportBinding>
      <wsp:Policy>
         <sp:TransportToken>
           <wsp:Policy>
                 <sp:HttpsToken RequireClientCertificate='true' />
           </wsp:Policy>
         </sp:TransportToken>


</wsp:Policy>
    </sp:TransportBinding>
 </wsp:All>


 <wsp:All>
     <sp:TransportBinding>
      <wsp:Policy>
         <sp:TransportToken>
           <wsp:Policy>
                 <sp:HttpsToken RequireClientCertificate='true' />
           </wsp:Policy>
         </sp:TransportToken>
      </wsp:Policy>
     </sp:TransportBinding>
 </wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```

Note that both authoring styles are equivalent, however there may be reasons to choose one form over another depending on the use. For example, when multiple alternatives are present in a policy, the normal form may express the choices more explicitly. On the other hand, the compact form is more readable for humans when an assertion is marked as optional using the `@optional` attribute as our example illustrates above.

# 5. Guidelines for Modeling New Assertions

## 5.1 New Policy Domains

When creating a new policy domain, it is important to understand how policy expressions are used by the framework. Some WS-Policy domains represent infrastructure efforts like WS-SecurityPolicy and WS-RM Policy. We also anticipate that individual web services applications and deployments may wish to reuse the WS-Policy framework in order to enable dynamic negotiation of business requirements and capabilities at runtime.

New policy authors should not try to overload assertions. A single assertion indicates a single behavior. Sets of assertions can by grouped by an operator "all". This indicates that there is a relationship between the assertions and they now constitute a policy alternative. Choices between alternatives can be indicated by an "exactly one" operator. This basic set of operators allows authors a wide range of options for expressing the possible combinations of assertions within their domain.

It requires a good deal of effort to evaluate the capabilities of a domain and capture them in a way that reflects the options of the domain if the domain has a lot of assertions to define. Interoperability testing of new policy domains is recommended to ensure that consumers and providers are able to use the new domain assertions.

New authors are encouraged to look at *Web Services Reliable Messaging Policy* to see an example of a relatively simple domain that has defined 3 assertions. Domain authors are encouraged to look at *WS-SecurityPolicy* to see an example of a complex domain that has been decomposed into a set of policy expressions.

## 5.2 Single Domains

When considering the creation of a new domain of policy assertions, it is important to identify whether or not the domain is self-contained or at least if a subset of the domain can be well defined. A domain that expresses a broad set of capabilities will also need to have community supporting implementations to provide value to the consumers. Ultimately it is the consumers and providers that will determine whether a particular set of assertions correctly characterize a domain. A new community should avoid duplicating assertions that have already been defined as this will create ambiguity not clarification. New WS-Policy authors should focus on creating assertions for those specific constraints and capabilities that do not overlap with other domains but that communicate new functionality.

**Comment [asv33]:** ISSUE 3989 - Suggested format for guidelines:

Identify a scenario, articulate the problem statement at the assertion design level, enumerate the factors that need to be considered and highlight best practices. The Architecture of the WWW document is a good model to follow.

Further more, providing the readers with a list of good practice statements upfront (right after the table of contents) would be very useful. For instance, http://www.w3.org/TR/webarch/

**Comment [asv34]:** What is the basis for any relationship between assertions when they are combined using policy operators?

**Comment [asv35]:** ISSUE 3983 – 'New Policy Authors', 'New Authors', 'Domain Authors', 'WS-Policy Authors', 'Assertion Authors' and 'Practitioners' – to avoid confusion, suggest using one phrase instead of multiple phrases. Suggestion: 'Assertion Author (s)'.

The model advocated for new assertion development is a cooperative marketplace [some might say its an "opt-in" model]. The providers of services need to find value in the set of assertions or they will not include the assertions in their service descriptions.

A review by a broad community is the best way to ensure that the granularity of a set of domain assertions is appropriate.

## 5.3 Comparison of Nested and Parameterized Assertions

There are two different ways to provide additional information in an assertion beyond its type. We cover these two cases below followed by a comparison of these approaches targeting when to use either of the approach.

### 5.3.1 Assertions with Parameters

The framework allows WS-Policy domain authors to define name value pairs, for example, to qualify an assertion. For some domains it will be appropriate to specify these parameters instead of nesting assertion elements.

Note that parameters of assertions include the following:

- Complex elements with element children that cannot be policy assertions.
- Elements that have attributes

In the example below, `sp:Body` and `sp:Header` elements are the two assertion parameters of the `sp:SignedParts` policy assertion (this assertion requires the parts of a message to be protected).

*Example 5-1. Policy Assertion with Assertion Parameters*

```
<Policy>
  <sp:SignedParts>
    <sp:Body />
    <sp:Header />
  </sp:SignedParts>
</Policy>
```

### 5.3.2 Nested Assertions

The framework provides the ability to "nest" policy assertions. For domains with a complex set of options, nesting provides one way to indicate dependent elements within a behavior. The granularity of assertions is determined by the authors and it is recommended that care be taken when defining nested policies to ensure that the options provided appropriately specify policy alternatives within a specific behavior.

We will use the WS-SecurityPolicy to illustrate the use of nested assertions.

**Comment [asv36]:** ISSUE 3984 - Does the framework allow assertion authors to define assertion parameters or name value pairs?

**Comment [asv37]:** ISSUE 3986 - Why is the parameter versus nested policy decision made at the domain level?

**Comment [asv38]:** ISSUE 3985 - This list is confusing. In the XML representation, the (non-policy) child elements and attributes of an assertion are the assertion parameters.

**Comment [asv39]:** ISSUE To Be Filed - What is the best practice for leveraging nested policy assertions?

Securing messages is a complex usage scenario. The WS-SecurityPolicy authors have defined the `sp:TransportBinding` policy assertion to indicate the use of transport-level security for protecting messages. Just indicating the use of transport-level security for protecting messages is not sufficient. To successfully interact with a Web service, the consumer must know not only that transport-level security is required, but also the transport token to use, the secure transport to use, the algorithm suite to use for performing cryptographic operations, etc. The `sp:TransportBinding` policy assertion can represent these dependent behaviors.

A policy assertion like the `sp:TransportBinding` identifies a visible behavior that is a requirement. A nested policy expression can be used to enumerate the dependent behaviors on the Transport binding. A nested policy expression is a policy expression that is a child element of another policy assertion element. A nested policy expression further qualifies the behavior of its parent policy assertion.

In the example below, the child Policy element is a nested policy expression and further qualifies the behavior of the `sp:TransportBinding` policy assertion. The `sp:TransportToken` is a nested policy assertion of the `sp:TransportBinding` policy assertion. The `sp:TransportToken` assertion requires the use of a specific transport token and further qualifies the behavior of the `sp:TransportBinding` policy assertion (which already requires the use of transport-level security for protecting messages).

*Example 5-2. Transport Security Policy Assertion*

```
<sp:TransportBinding>
  <Policy>
    <sp:TransportToken>
      <Policy>
        <sp:HttpsToken RequireClientCertificate="false" />
      </Policy>
    </sp:TransportToken>
    <sp:AlgorithmSuite>
      <Policy>
        <sp:Basic256Rsa15/>
      </Policy>
    </sp:AlgorithmSuite>
  </Policy>
</sp:TransportBinding>
```

The `sp:AlgorithmSuite` is a nested policy assertion of the `sp:TransportBinding` policy assertion. The`sp:AlgorithmSuite` assertion requires the use of the algorithm suite

identified by its nested policy assertion (`sp:Basic256Rsa15` *in the example above*) and further qualifies the behavior of the `sp:TransportBinding` policy assertion.

Setting aside the details of using transport-level security,, a policy-aware client that recognizes this policy assertion can engage transport-level security and its dependent behaviors automatically. That is, the complexity of security usage is absorbed by a policy-aware client and hidden from these Web service developers.

### 5.3.3 Considerations for choosing parameters vs nesting

The main consideration for selecting parameters or nesting of assertions, is that *the framework intersection algorithm processes nested alternatives, but does not consider parameters in its algorithm*.

Domain authors should recognize that the framework can yield multiple assertions of the same type. The *QName* of the assertion is the only vehicle for the framework to match a specific assertion, NOT the contents of the element. If the assertion is a parameterized assertion the authors must understand that this type of assertion will require additional processing by consumers in order to disambiguate the assertions or to understand the semantics of the name value pairs, complex content, attribute values contribution to the processing. Therefore, if the domain authors want to delegate the processing to the framework, utilizing nesting should be considered. Otherwise, domain specific comparison algorithms would need to be devised and be delegated to the specific domain handlers that are not visible to the WS-Policy framework. The tradeoff is the generality vs. the flexibility and complexity of the comparison expected for a domain.

**Comment [asv40]:** 'would need' is too strong and there isn't any basis to support this statement. Suggest toning down to 'may need'.

## 5.4 Self Describing Messages

WS-Policy is intended to communicate the requirements, capabilities, preferences and behaviors of nodes that provide the message's path, not specifically to declare properties of the message semantics. One of the advantages of Web services is that an XML message can be stored and later examined (e.g. as a record of a business transaction) or interpreted by an intermediary; however, if information that is necessary to understand a message is not available, these capabilities suffer.

**Comment [asv41]:** 'preferences' cannot be expressed using WS-Policy 1.5.

Policy assertions should not be used to express the semantics of a message. Rather, if a property is required to understand a message, it should be communicated in the message, or be made available by some other means (e.g., being referenced by a URI in the message) instead of being communicated as a policy element.

For example, if the details of a message's encryption ( e.g., the cipher used, etc) are expressed in policy that isn't attached to the message, it isn't possible to later decipher it. This is very different from expressing, in policy, what ciphers (and so forth) are supported by a particular endpoint, or those that are required in a particular message; the latter are the intended uses of the WS-Policy framework.

As a result, the assertion authors should take into account that the following important concepts when designing assertions and documenting the semantics of the assertion types. Firstly, an assertion type indicates a *runtime* behavior. Secondly, an assertion should also indicate how the assertion type can be inferred or indicated from a message

at runtime. If there is a need for the behavior to be represented in a persistent way or if there is a need for additional data or metadata that is present in a message to be persisted, it should be incorporated into the assertion design or in the message itself. In essence, the assertion authors should consider how to make messages self describing when utilizing their assertions by specifying additional properties, headers, etc. that must be present in a message as part of their assertion design.

If the messages could not be made self describing by utilizing additional properties present in the message as required by the assertion, it would be necessary to determine the behaviors engaged at runtime by additional means. A general protocol that aids in determining such behaviors may be utilized, however a standard protocol for this purpose is currently not available to ensure interoperability. Thus, a private protocol should be used with care.

Another approach is to use of the assertion to selectively apply to subjects. For example, a dedicated endpoint may be allocated to ensure the engagement of a behavior that is expressed by a policy assertion. This approach can be considered when messages can not be self describing.

## 5.5 Policy Assertions Designating Optional Behaviors

Optional behaviors represent behaviors which may be engaged by a consumer. When using the compact authoring form for assertions, behaviors are marked by using `wsp:optional` attribute that has a value, "true". During the process of normalization, the runtime behavior is indicated by two policy alternatives, one with and one without containing the assertion. In a consumer/provider scenario, the choice of engaging the runtime behavior is upon the consumer although the provider is capable of engaging the runtime behavior. In order to simplify reference to such assertions, we just use the term optional assertions in this section.

The *Web Services Policy Primer* document contains an example that proposes the use of *MTOM* as an optional behavior that can be engaged by a consumer. The primer proposes that an assertion that identifies the use of MIME Multipart/Related serialization (see *MTOM*, *XOP* for messages to enable a Policy-aware clients to recognize the policy assertion and if they select an alternative with this assertion, they engage Optimized MIME Serialization for messages.

The semantics of this assertion declare that the behavior is reflected in messages: they use an optimized wire format (MIME Multipart/Related serialization). Note that in order for an optional behaviors to be engaged, the wire message that would utilize the specific assertion must be self describing. For example, an inbound message to a web service that asserts MTOM, must evaluate, the protocol format of the message to determine whether the incoming message adheres to the Optimized MIME Serialization. By examining the message, the provider can determine whether the policy alternate that contains the MTOM assertion is being selected.

Assertion authors should be aware that optional behaviors, like utilizing optional support for Optimized MIME Serialization require some care considering the scoping of the assertion that is applicable.

- Since optional behaviors indicate optionality for both the provider and the consumer, behaviors that must always be engaged by a consumer must not be marked as "optional" with a value "true" since presence of two alternatives due to normalization enables a consumer to choose the alternative that does not contain the assertion, and thus making the behavior not being engaged in an interaction.
- As demonstrated in the MIME optimization behavior, behaviors must be engaged with respect to messages that are targeted to the provider so that the provider can determine that the optional behavior is engaged. In other words, the requirement of self describing nature of messages [**5.4 Self Describing Messages** ] in order to engage behaviors must not be forgotten with regard to the client's ability to detect and select the alternative if it is to participate in the exchange.
- The target scope of an optional assertion is an important factor for assertion authors to consider as it determines the *granularity* where the behavior is optionally engaged. For example, if the assertion is targeted for an endpoint policy subject, it is expected to govern all the messages that are indicated by the specific endpoint when optional behavior is *engaged* . Since the behavior would be applicable to policy subject that is designated, it is important for the policy assertion authors to choose the appropriate level of granularity for optional behaviors, to consider whether a specific message or all messages, etc. are targeted.
    - Attaching optional assertions to outbound-messages using message policy subject require some care. An explicit, out of band mechanism may be necessary to enable a client to indicate that the optional behavior is engaged. Currently such a mechanism is outside the scope of WS-Policy Framework.
    - When optional behaviors are indicated by attaching assertions with only one side of an interaction, such as an inbound message of a request-response, the engagement of the rest of the interaction will be *undefined*. For example, if a request-response interaction only specified MTOM optimization for an inbound message, it would not be clear whether the outbound message from the provider could also utilize the behavior. Therefore, the assertion authors are encouraged to consider how the attachment on a message policy subject on a response message should be treated when optional behaviors are specified for message exchanges within a request response for response messages, using message policy subject. Leaving the semantics undescribed may result in providers making assumptions (i.e. if the incoming message utilized the optimization, the response will be returned utilizing the MTOM serialization). Similarly, if engagement of a behavior is only specified for an outbound message, the policy assertion authors should consider to describe the semantics if the incoming messages also utilized the behavior. This is especially important if the assertion is applicable to more than one specific policy subject. One approach that is currently taken by WS-RM Policy [*Web Services Reliable Messaging Policy*] is to introduce

both message and endpoint policy subjects for one of its assertions and require the use of endpoint policy subject when message policy subject is used via attachment.

- Optional assertion authors should explicitly state how the behavior that is enabled by the assertion would be engaged when they are designing their assertion, whether by specific headers or some other means. See also **5.4 Self Describing Messages** .

## 5.6 Considerations for Intersection and Merging

## 5.7 Typing Assertions

**Comment [asv42]:** ISSUE 3990 - What is the goal for this section? This section appears to cover a range of topics: versioning policy assertions, external attachment mechanism and identifying policy subjects. BTW, QName as the central mechanism is covered in Section 5.3.3.

Since a *QName* is the central mechanism for identifying a policy assertion, assertion authors should be aware of the possible evolution of their assertions and how this would impact the semantics overtime. A namespace associated with the assertion may be used to indicate a specific version of an assertion.

WS-PolicyAttachment provides a means of associating an assertion with arbitrary subjects, regardless of their nature. This flexibility can lead to ambiguity in the interpretation of policy; for example, if one attaches an assertion with the semantic "must be encrypted" to a SOAP endpoint, it's unclear what must be encrypted.

To avoid this confusion, assertion definitions should be precise about their semantics and include information that restricts their set of permissible policy subjects appropriately and indicates which Qnames are associated with which subjects. One way to do this is to generally determine if an assertion is specific to a policy attachment mechanism. An example could be identifying whether the assertion expressed is associated with behaviors (endpoints) or artifacts ( messages) and then constraining the use of an assertion to one of these subjects.

Thus our example encryption assertion would have a subject of "message", and could only be attached to messages, where the assertion is valid. However, authors need to be aware that *policy attachment subjects are not limited to the subjects defined in WSDL*. The external attachment model in WS-PolicyAttachment allows for the definition of other domain expressions to be policy subjects. More of this topic is covered in the *Web Services Policy Primer*

| EXAMPLE |
|---|

## 5.8 Levels of Abstraction in WSDL

A behavior identified by a policy assertion applies to the associated policy subject. If a policy assertion is to be used within WSDL, policy assertion authors must specify a WSDL policy subject. The policy subject is determined with respect to a behavior as follows:

- If the behavior applies to any message exchange using any of the endpoints offered by a service then the subject is the service policy subject.

- If the behavior applies to any message exchange made using an endpoint then the subject is the endpoint policy subject.
- If the behavior applies to any message exchange defined by an operation then the subject is the operation policy subject.
- If the behavior applies to an input message then the subject is the message policy subject - similarly for output and fault message policy subjects.

WS-Policy authors that wish to utilize WSDL policy subjects need to understand how the assertions will be processed in intersection and merging and the implications of the processing for considering a specific attachment point and policy subject. This topic is considered in detail in *Web Services Policy Primer*

The current set of subjects as mapped to the WSDL 1.1 elements, can also constrain the assertion constructs. For Example, In WS-RM, the domain authors chose to support certain capabilities at the endpoint level. This resulted in the finer granularity of the assertion to apply at the message policy subject, but the assertion semantics also indicates that the if the senders choose to engage RM semantics (although not specified via attachment in WSDL at incoming messages), the providers will honor the engagement of RM. This is illustrative of how the assertion author can specify additional constraints and assumptions for attachment and engagement of behavior.

If the capability is not really suitable and may imply different semantics with respect to attachment points, the assertion authors should consider the following

- Decompose the semantics with several assertions
- Rewrite a single assertion targeting a specific subject.

For a given WSDL policy subject, there may be several attachment points. For example, there are three attachment points for the endpoint policy subject: the port, binding and portType element. Policy assertion authors should identify the relevant attachment point when defining a new assertion. To determine the relevant attachment points, authors should consider the scope of the attachment point. For example, an assertion should only be allowed in the portType element if the assertion reasonably applies to any endpoint that ever references that portType. Most of the known policy assertions are designed for the endpoint, operation or message policy subject.

In using WSDL attachment, it should be noted that the service policy subject is a collection of endpoint policy subjects. The endpoint policy subject is a collection of operation policy subjects and etc. As a result, the WSDL policy subjects compose naturally. It is quite tempting to associate the identified behavior to a broader policy subject than to a fine granular policy subject. For instance, it is convenient to attach a supporting token assertion (defined by the Web Services Security Policy specification) to an endpoint policy subject instead of a message policy subject. Similarly, for authoring convenience, an assertion author may allow the association of an assertion to multiple policy subjects. If an assertion is allowed to be associated with multiple policy subjects then *the assertion author has the burden to describe the semantics of multiple instances of the same assertion attached to multiple policy subjects at the same time in order to avoid conflicting behavior.*

One approach is to specify a policy subject, choose the most granular policy subject that the behavior applies to and specify a preferred attachment point in WSDL.

**Comment [asv43]:** What does this paragraph recommend? If none, suggest dropping this paragraph.

**Comment [asv44]:** What does 'not really suitable' mean in this context?

However, this approach only works if the policy subject is a true WSDL construct other than some other protocol concept that is layered over WSDL message exchanges. For example, the WS-RM Policy is a capability that governs a target endpoints capability to accept sequences that is beyond single message exchanges. Therefore, its semantics encompasses the cases when message level policy subjects may be used as attachment but considers when sequences are present. In addition, when the policy assertions do not target wire-level behaviors but rather abstract requirements, this technique does not apply.

## 5.9 Lifecycle of Assertions

**Comment [asv45]:** ISSUE 3987 - Does the phrase 'lifecycle of assertions' mean 'versioning policy assertions'? If so, why shouldn't we use the phrase 'versioning policy assertions'?

WS-Policy authors need to consider not just the expression of the current set of requirements but how they anticipate new assertions being added to the set. There are three aspects that govern an assertions lifecycle:

- Assertion Extensibility
- Policy Language Extensibility
- Subject attachment Extensibility

**Comment [asv46]:** ISSUE 3987 – Overlaps with Section 4.4.8 in the Primer – 'Versioning Policy Language'.

### 5.9.1 Referencing Policy Expressions

**Comment [asv47]:** ISSUE 3979 - What is the guideline for assertion authors in this section? It is hard to decode any guidelines from this section. If there aren't any guidelines, suggest dropping this paragraph.

The *Web Services Policy Primer* illustrates how providers can utilize the identification mechanism defined in the Policy specification to expose a complex policy expression as a reusable building block for other policy expressions by reference. Domain assertion authors, especially those defining complex assertions that include nesting or complex types should consider specifying or recommending naming conventions in order to promote reuse. Reuse through referencing allows a policy expression to be utilized not only within another expression but also allows specification of additional policy subjects and their association to common policy expressions that are identified. It also promotes manageability of the expressions as they are uniquely identified.

**Comment [asv48]:** ISSUE 3979 - How do assertion naming conventions promote reuse? Should help the assertion authors understand the benefits of any eluded guidelines here.

### 5.9.2 Factors in Extending Assertions within a Domain

Extensibility affects the policy subjects and attachment semantics.

### 5.9.3 Evolution of Assertions (Versioning and Compatibility)

4.4.7. Over time, there may be multiple equivalent behaviors emerging in the Web Service interaction space. Examples of such multiple equivalent behaviors are WSS: SOAP Message Security 1.0 vs. 1.1 and WS-Addressing August 2004 version vs. WS-Addressing W3C Recommendation [*WS-Addressing Core*]. These equivalent behaviors are mutually exclusive for an interaction. Such equivalent behaviors can be modeled as independent assertions. The policy expression in the example below requires the use of WSS: SOAP Message Security 1.0.

| *Example 5-4. Example 4-2. Message-level Security and WSS: SOAP Message Security 1.0* |
|---|

ADD THE EXAMPLE HERE

The policy expression in the example below requires the use of WSS: SOAP Message Security 1.1. These are multiple equivalent behaviors and are represented using distinct policy assertions.

*Example 5-5. Example 4-3. Message-level Security and WSS: SOAP Message Security 1.1*

ADD THE EXAMPLE HERE

Best practice: use independent assertions for modeling multiple equivalent behaviors.

# 6. Inter-domain Policy and Composition Issues

Comment [asv49]: What are the guidelines for assertion authors in this section?

Domain authors must be aware of the interactions between their domain and other domains. For example, security assertions interact with other protocol assertions in a composition. Although modeling protocol assertions may appear to be an independent behavior, protocol assertions and security assertions affect transport bindings and their interactions must be considered. For example, utilization of WS-Security Policy with other protocols affect transport binding and would result in nested policy assertions when additional protocols are composed with *WS-Security 2004*. Thus, domain authors should be aware of the compositional semantics with other related domains. The protocol assertions that require composition with WS-Security should be particularly aware of the nesting requirements on top of transport level security.

# 7. Applying Best Practices for Policy Attachment

Comment [asv50]: ISSUE 3978 - This section enumerates best practices for policy attachment authors (NOT Assertion Authors). Is the WG planning to provide guidelines for policy attachment authors? If positive, does this section outline sufficient guidelines for policy attachment authors?

Given the scope of this document 'Guidelines for Policy Assertion Authors', section 7 is not in the minimum to declare victory on the guidelines document.

## 7.1 Appropriate Attachment: Preserving Context-Free Policies

Policy attachment should not affect the interpretation of Policy alternatives. If it did, each policy assertion would need to be written with different (and possibly unknown) attachment mechanisms in mind. In particular, the timing of a policy attachment or the role that a party who attaches policy should have no bearing on the evaluation of the policy assertion

Comment [asv51]: What does 'timing of a policy attachment' mean?

## 7.2 Appropriate Attachment: Identifying Assertion Subjects

Each policy attachment mechanism should unambiguously identify the subject of the attached assertions. Generally, this should be a specific SOAP node or a specific message between two SOAP nodes. Some attachment mechanisms may encompass multiple notes or messages, for example, "the message along its entire path".

### 7.2.1 Interaction between Subjects

If the best practices are followed, and the assertions are scoped according to their subject, then multiple policy domains may be combined without conflict. Each domain

should define any limitations at the policy subject level that might impact interoperability (i.e. WS-SecurityPolicy - binding abstraction to group capabilities per message exchange).

### 7.3 Appropriate Attachment: Identifying Assertion Sources

As with identifying Policy subjects, policy attachment mechanisms should make it possible to clearly identify the source of a poly assertion both for debugging and for verification. This could take several forms: it could be assumed ( in WSDL, the source of the assertion is the same as the WSDL provider) or it could be proven ( using *WS-Trust*).

## 8. Scenario and a worked example

**Comment [asv52]:** ISSUE 3988 - This section describes how to use transport binding security policy assertion (and other assertions such as addressing and asymmetric binding), how to name and reference policy expressions, how to attach policy expressions to WSDL 11 constructs, etc.

Given the scope of this document – 'Guidelines for Policy Assertion Authors' – it will be very useful to illustrate how to design an assertion by answering a set of questions identified by this document and applying best practices or guidelines outlined in this document.

To illustrate the topics explored in this document, we include an example of a web service and how a fictitious company might utilize the WS-Policy Framework to enable Web Service interoperability. CompanyA has determined to utilize WS-Security, WS-Addressing and WS-Reliable Messaging in all its new web service offerings and has instructed its developers to use the policy assertions defined by the following documents:

- Web Services Security Policy
- Web Services Reliable Messaging Policy
- Web Services Addressing WSDL Binding

The application developers at CompanyA are instructed to review the current web services at CompanyA and propose a plan for adding policy assertions.

The application developers collect information about web services within CompanyA and determine that all of the web services already have a WSDL 1.1 description. The developers have determined that Company A's web services fall into two types of web services. There are those that fall into the "default" category, and will use a predefined set of policy assertions, and there are those that use the default but also extend the policy alternatives.

They have also determined that for the both types, the appropriate policy subject is the endpoint. They determined this because the capabilities apply to all operations and messages for the web service not to any one individual operation or message exchange.

Service A is a WSDL 1.1 conformant web service and requires the use of transport-level security for protecting messages as well as including addressing headers. Employees of CompanyA have already incorporated `wss:Security` headers into their messages.

*Example 8-1. Message with Security Headers*

```
<soap:Envelope>

  <soap:Header>

      <wss:Security soap:mustUnderstand ="1">
```

```
       <wsu:Timestamp u:Id=_0">
            <wsu:Created> 20006-01-19T02:49:53.914Z </u:Created>
            <wsu:Expires> 20006-01-19T02:54:53.914Z </u:Expires>
       </wsu:Timestamp>
    </wss:Security>
      <wsa:To> http://CompanyA/quote <wsa:To>
      <wsa:Action> http://CompanyA/GetRealQuote</wsa:Action>
</soap:Header>
<soap:Body>
</soap:Envelope>
```

The SOAP message in the example above includes security timestamps that express creation and expiration times of this message. CompanyA requires the use of these security timestamps and transport-level security, such as HTTPS for protecting messages.

The example below illustrates a policy expression that CompanyA has created for its employees to use on their web services to indicate the use of addressing and transport-level security for securing messages.

*Example 8-2. CompanyA-ProfileA*

```
<Policy URI=http://www.CompanyA.com/WebServicesProfileA.xml>
     <wsa:UsingAddressing />
     <sp:TransportBinding></sp:TransportBinding>
</Policy>
```

The `sp:TransportBinding` element is a policy assertion. The assertion identifies the use of transport-level-security - such as HTTPS for protecting messages at the transport level. CompanyA's policy aware clients can now recognize this policy assertion and if they support it, engage in transport level security for protecting messages and providing security timestamps in SOAP envelopes for any WSDL with this policy attached.

When creating the policy for the default web services, the developers took into consideration several factors. First, all their web services were WSDL 1.1 web services. Second, they wanted to reuse policy assertions where ever possible. Third, they wanted to ensure that where possible they would support alternatives rather than forcing a single client configuration.

The developers read the WS-Policy specification and noted that there were 3 ways to express combinations of behaviors. The three policy operators, ( Policy, All and ExactlyOne) were considered and the result was the creation of two policy elements.

The first policy is shown in Figure *CompanyA-ProfileA* and it is the policy that is used by many web services at Company A that rely on https to provide transport level protection of messages.

The second policy is shown in Figure *CompanyA-ProfileB* and it offers requestors of a service the ability to provide additional integrity protection by including WS-Security Headers to protect the message content after it is processed by the transport. The additional security processing is not required by all CompanyA web services.

*Example 8-3. CompanyA-ProfileB ( not expanded)*

```
<Policy wsu:Id="CompanyA-ProfileB">

    <wsa:UsingAddressing /> <ExactlyOne>

    <sp:TransportBinding></sp:TransportBinding>

    <sp:AsymmetricBinding></sp:AssymetricBinding>

    </ExactlyOne> </Policy>
```

We have shown above that CompanyA offered a second profile that included two security options. The details of the Bindings, requires a more detailed exploration of some of the other features of the WS-Policy Framework.

When WS-Policy authors create sets of Policy assertions, like WS-Security Policy they need to consider expressing the semantics of their domain in a way that policy consumers, like Company A, can utilize them. In this case, the WS-SecurityPolicy authors factored out common elements of security mechanisms and utilized a feature of WS-Policy called "nested" assertions. In the case of an `sp:TransportBinding` assertion, just indicating the use of transport-level security for protecting messages is not sufficient. For a consumer of a web service provided by a company, like CompanyA, to successfully interact, the consumer must also know what transport token, what algorithm suite, etc. is required. The `sp:TransportBinding` assertion, can (and has) represent (ed) these dependent behaviors as "nested" policy assertions.

In the example below the child Policy element is a nested policy behavior and further qualifies the behavior of the `sp:TransportBinding` policy assertion.

*Example 8-4. CompanyA-ProfileB (fully expanded)*

```
<Policy wsu:Id="CompanyA-ProfileB">

     <wsa:UsingAddressing />

     <ExactlyOne>

        <sp:TransportBinding>

             <Policy>
```

```
              <sp:TransportToken>
                 <Policy>
                   <sp:HttpsToken RequireClienCertificate="false" />
                    </Policy>
                 </sp:TransportToken>
                 <sp:AlgorithmSuite>
                    <Policy>
                    <sp:Basic256Rsa15 />
                    </Policy>
                 </sp:AlgorithmSuite>
              </Policy>
       </sp:TransportBinding>
        <sp:AsymmetricBinding>
  </sp:AssymetricBinding>
      </ExactlyOne>
</Policy>
```

`The sp:AlgorithmSuite` is a nested policy assertion of the `sp:TransportBinding` assertion and indicates that this suite is required. The `sp:TransportToken` is a nested policy assertion that indicates the use of a specific type of token, in this case an HttpsToken.

It should be noted that each policy has an Identifier. In the case of the default policy expression, CompanyA has decided that this policy expression should be broadly available via a URI. There are advantages and disadvantages to using each type of identifier. For URI's there is the issue of maintaining the policy expression when it may no longer be used ( CompanyA gets bought by CompanyB and starts using the policies of Company B, but some "old" consumers may still try to reference the URI).

For the second type of web services, which may be used only by certain of CompanyA's business partners, the id is an XML ID. The relative URI for referencing this within the same WSDL document is #CompanyA-ProfileB. This can be useful for company's when the policy expressions are agreed to between partners but may be changed as the business agreements change. But the disadvantage is that the policy expression must be included in each WSDL document.

Since CompanyA has decided to use well known policy expressions that are themselves part of a specification, they adhere to the guidance given in the WS-SecurityPolicy specification and attach the policies to the web service endpoint policy

subject as recommended by the WS-SecurityPolicy specification. For the default web services, the URI is included in the wsdl binding for each web service.

*Example 8-5.*

```
<wsdl:binding name="CompanyADefaultBinding" type="tns:CompanyADefault">

  <wsp:PolicyReference
URI="http://www.CompanyA.com/WebServicesProfileA.xml">



<wsdl:operation name="GetQuote"> </wsdl:operation>

</wsdl:binding>
```

The partner specified policy is included in the beginning of the wsdl 1.1document and referenced by the binding for the service as in the example below.

*Example 8-6.*

```
<wsdl:definitions name="StokeQuote"

    targetNamespace="http:..">

<wsp:Policy wsu:Id="CompanyA-ProfileB">

     <wsa:UsingAddressing />

     <ExactlyOne>

        <sp:TransportBinding>

             <Policy>

              <sp:TransportToken>

                <wsp:Policy>

                   <sp:HttpsToken RequireClienCertificate="false" />

                  </wsp:Policy>

               </sp:TransportToken>

               <sp:AlgorithmSuite>

                  <wsp:Policy>

                   <sp:Basic256Rsa15 />

                  </wsp:Policy>

               </spAlgorithmSuite>

             </Policy>
```

```
            </sp:TransportBinding>
          <sp:AsymmetricBinding>
            </sp:AssymetricBinding>
       </ExactlyOne>
</wsp:Policy>

<wsdl:binding name="CompanyADefaultBinding" type="tns:CompanyADefault">
  <wsp:PolicyReference id=#CompanyA-ProfileB>
      <wsdl:operation name="GetQuote"> </wsdl:operation>

</wsdl:binding>
```

In some cases, companies may chose to implement their own assertions. When companies chose to become policy authors they need to consider not only the definition of the behavior that the assertion represents but they need to consider how new assertions will be intersected and merged with other assertions in the calculation of an effective policy and this also indicates they need to consider policy subjects.

The policy framework only defines an algorithm for calculating effective policies for WSDL 1.1 based subjects.

**Comment [asv53]:** This statement is incorrect.

## A. Security Considerations

Security considerations are discussed in the *Web Services Policy Framework* document.

## B. XML Namespaces

The table below lists XML Namespaces that are used in this document. The choice of any namespace prefix is arbitrary and not semantically significant.

*Table B-1. Prefixes and XML Namespaces used in this specification.*

| Prefix | XML Namespace | Specifications |
|---|---|---|
| soap | http://www.w3.org/2003/05/soap-envelope | [*SOAP 1.2 Messaging Framework*] |
| sp | http://schemas.xmlsoap.org/ws/2005/07/securitypolicy | [*WS-* |

*Table B-1. Prefixes and XML Namespaces used in this specification.*

| Prefix | XML Namespace | Specifications |
|---|---|---|
| | | *SecurityPolicy*] |
| wsa | http://www.w3.org/2005/08/addressing | [*WS-Addressing Core*] |
| wsdl | http://schemas.xmlsoap.org/wsdl/ | [*WSDL 1.1*] |
| wsp | http://www.w3.org/@@@@/@@/ws-policy | [*Web Services Policy Framework*, *Web Services Policy Attachment*] |
| wsrmp | http://docs.oasis-open.org/ws-rx/wsrmp/200608 | [*Web Services Reliable Messaging Policy*] |
| wss | http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd | [*WS-Security 2004*] |
| wsu | http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd | [*WS-Security 2004*] |

## C. References


**[MTOM]**
*SOAP Message Transmission Optimization Mechanism*, M. Gudgin, N. Mendelsohn, M. Nottingham and H. Ruellan, Editors. World Wide Web Consortium, 25 January 2005. This version of the SOAP Message Transmission Optimization Mechanism Recommendation is http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/. The latest version of SOAP Message Transmission Optimization Mechanism is available at http://www.w3.org/TR/soap12-mtom/.

**[SOAP 1.1]**
*Simple Object Access Protocol (SOAP) 1.1*, D. Box, et al, Editors. World Wide Web Consortium, 8 May 2000. Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

**[SOAP 1.2 Messaging Framework]**
*SOAP Version 1.2 Part 1: Messaging Framework*, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 24 June 2003. This version of the SOAP Version 1.2 Part 1: Messaging Framework Recommendation is http://www.w3.org/TR/2003/REC-

soap12-part1-20030624/. The latest version of SOAP Version 1.2 Part 1: Messaging Framework is available at http://www.w3.org/TR/soap12-part1/.

**[XOP]**

*XML-binary Optimized Packaging*, M. Gudgin, N. Mendelsohn, M. Nottingham and H. Ruellan, Editors. World Wide Web Consortium, 25 January 2005. This version of the XML-binary Optimized Packaging Recommendation is http://www.w3.org/TR/2005/REC-xop10-20050125/. The latest version of XML-binary Optimized Packaging is available at http://www.w3.org/TR/xop10/.

**[WS-Addressing Core]**

*Web Services Addressing 1.0 - Core*, M. Gudgin, M. Hadley, and T. Rogers, Editors. World Wide Web Consortium, 9 May 2006. This version of the Web Services Addressing 1.0 - Core Recommendation is http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. The latest version of Web Services Addressing 1.0 - Core is available at http://www.w3.org/TR/ws-addr-core.

**[WSDL 1.1]**

*Web Services Description Language (WSDL) 1.1*, E. Christensen, et al, Authors. World Wide Web Consortium, March 2001. Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

**[WSDL 2.0 Core Language]**

*Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, R. Chinnici, J. J. Moreau, A. Ryman, S. Weerawarana, Editors. World Wide Web Consortium, 27 March 2006. This version of the WSDL 2.0 specification is http://www.w3.org/TR/2006/CR-wsdl20-20060327. The latest version of WSDL 2.0 is available at http://www.w3.org/TR/wsdl20.

**[Web Services Policy Framework]**

*Web Services Policy 1.5 - Framework*, A. S. Vedamuthu, D. Orchard, M. Hondo, T. Boubez and P. Yendluri, Editors. World Wide Web Consortium, @@, @@@@ @@@@. This version of the Web Services Policy 1.5 - Framework specification is at http://www.w3.org/TR/ws-policy/. The latest version of Web Services Policy 1.5 - Framework is available at http://www.w3.org/TR/ws-policy/.

**[Web Services Policy Attachment]**

*Web Services Policy 1.5 - Attachment*, A. S. Vedamuthu, D. Orchard, M. Hondo, T. Boubez and P. Yendluri, Editors. World Wide Web Consortium, @@, @@@@ @@@@. This version of the Web Services Policy 1.5 - Attachment specification is at http://www.w3.org/TR/ws-policy-attach. The latest version of Web Services Policy 1.5 - Attachment is available at http://www.w3.org/TR/ws-policy-attach/.

**[Web Services Policy Primer]**

*Web Services Policy 1.5 - Primer*, A. S. Vedamuthu, D. Orchard, M. Hondo, T. Boubez and P. Yendluri, Editors. World Wide Web Consortium, Draft.

**[Web Services Reliable Messaging]**

*Web Services Reliable Messaging (WS-ReliableMessaging)*, Doug Davis (IBM), Anish Karmarkar (Oracle), Gilbert Pilz (BEA), Steve Winkler (SAP), Umit Yalcinalp (SAP), August 7th, 2006, available at: http://docs.oasis-open.org/ws-rx/wsrm/200608/wsrm-1.1-rddl-200608.html

**[Web Services Reliable Messaging Policy]**

*Web Services Reliable Messaging Policy Assertion v1.1*, Doug Davis (IBM), Anish Karmarkar (Oracle), Gilbert Pilz (BEA), Steve Winkler (SAP), Umit Yalcinalp (SAP), August 4, 2006, available at: http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-rddl-200608.html

**[WS-Security 2004]**
*Web Services Security: SOAP Message Security 1.0*, A. Nadalin, C. Kaler, P. Hallam-Baker and R. Monzillo, Editors. Organization for the Advancement of Structured Information Standards, March 2004. Available at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

**[WS-SecurityPolicy]**
*WS-SecurityPolicy v1.0*, A. Nadalin, M. Gudgin, A. Barbir, and H. Granqvist, Editors. Organization for the Advancement of Structured Information Standards, 8 December 2005. Available at http://www.oasis-open.org/committees/download.php/15979/oasis-wssx-ws-securitypolicy-1.0.pdf.

**[WS-Trust]**
*Web Services Trust Language (WS-Trust)*, S. Anderson, et al, Authors. Actional Corporation, BEA Systems, Inc., Computer Associates International, Inc., International Business Machines Corporation, Layer 7 Technologies, Microsoft Corporation, Oblix Inc., OpenNetwork Technologies Inc., Ping Identity Corporation, Reactivity Inc., RSA Security Inc., and VeriSign Inc., February 2005. Available at http://schemas.xmlsoap.org/ws/2005/02/trust.

**[UDDI API 2.0]**
*UDDI Version 2.04 API*, T. Bellwood, Editor. Organization for the Advancement of Structured Information Standards, 19 July 2002. This version of UDDI Version 2.0 API is http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm. The latest version of the UDDI 2.0 API is available at http://uddi.org/pubs/ProgrammersAPI_v2.htm.

**[UDDI Data Structure 2.0]**
*UDDI Version 2.03 Data Structure Reference*, C. von Riegen, Editor. Organization for the Advancement of Structured Information Standards, 19 July 2002. This version of UDDI Version 2.0 Data Structures is http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm. The latest version of the UDDI 2.0 Data Structures is available at http://uddi.org/pubs/DataStructure_v2.htm.

**[UDDI 3.0]**
*UDDI Version 3.0.1*, L. Clément, et al, Editors. Organization for the Advancement of Structured Information Standards, 14 October 2003. This version of the UDDI Version 3.0 is http://uddi.org/pubs/uddi-v3.0.1-20031014.htm. The latest version of the UDDI 3.0 specification is available at http://uddi.org/pubs/uddi_v3.htm.

## D. Acknowledgements (Non-Normative)

This document is the work of the W3C Web Services Policy Working Group.

Members of the Working Group are (at the time of writing, and by alphabetical order): Dimitar Angelov (SAP AG), Abbie Barbir (Nortel Networks), Charlton Barreto (Adobe Systems Inc.), Sergey Beryozkin (IONA Technologies, Inc.), Vladislav Bezrukov (SAP

AG), Toufic Boubez (Layer 7 Technologies), Paul Cotton (Microsoft Corporation), Jeffrey Crump (Sonic Software), Glen Daniels (Sonic Software), Jacques Durand (Fujitsu Limited), Ruchith Fernando (WSO2), Christopher Ferris (IBM Corporation), William Henry (IONA Technologies, Inc.), Frederick Hirsch (Nokia), Maryann Hondo (IBM Corporation), Tom Jordahl (Adobe Systems Inc.), Philippe Le Hégaret (W3C/MIT), Jong Lee (BEA Systems, Inc.), Mark Little (JBoss Inc.), Ashok Malhotra (Oracle Corporation), Monica Martin (Sun Microsystems, Inc.), Jeff Mischkinsky (Oracle Corporation), Dale Moberg (Cyclone Commerce, Inc.), Anthony Nadalin (IBM Corporation), David Orchard (BEA Systems, Inc.), Fabian Ritzmann (Sun Microsystems, Inc.), Daniel Roth (Microsoft Corporation), Tom Rutt (Fujitsu Limited), Sanka Samaranayake (WSO2), Felix Sasaki (W3C/Keio), Skip Snow (Citigroup), Yakov Sverdlov (Computer Associates), Mark Temple-Raston (Citigroup), Asir Vedamuthu (Microsoft Corporation), Sanjiva Weerawarana (WSO2), Ümit Yalçinalp (SAP AG), Prasad Yendluri (webMethods, Inc.).

Previous members of the Working Group were: Bijan Parsia (University of Manchester), Seumas Soltysik (IONA Technologies, Inc.)

The people who have contributed to http://lists.w3.org/Archives/Public/public-ws-policy/ are also gratefully acknowledged.

## E. Changes in this Version of the Document (Non-Normative)

A list of substantive changes since the previous publication is below:

- TBD

## F. Web Services Policy 1.5 - Guidelines for Policy Assertion Authors Change Log (Non-Normative)

| Date | Author | Description |
|---|---|---|
| 20060829 | UY | Created first draft based on agreed outline and content |
| 20061013 | UY | Editorial fixes (suggested by Frederick), fixed references, bibl items, fixed dangling pointers, created eds to do |
| 20061018 | MH | Editorial fixes for readability, added example for Encrypted parts |
| 20061030 | UY | Fixes for Paul Cotton's editorial comments (20061020) |
| 20061031 | UY | Fixes for Frederick's editorial comments (20061025) |
| 20061031 | UY | Optionality discussion feedback integration |