Web Services Policy 1.5 - Guidelines for Policy Assertion Authors

W3C Working Draft 21 December 2006

This version:
http://www.w3.org/TR/2006/WD-ws-policy-guidelines-20061221
Latest version:
http://www.w3.org/TR/ws-policy-guidelines

Editors:
Asir S Vedamuthu, Microsoft Corporation
David Orchard, BEA Systems, Inc.
Frederick Hirsch, Nokia
Maryann Hondo, IBM Corporation
Prasad Yendluri, webMethods, Inc.
Toufic Boubez, Layer 7 Technologies
Ümit Yalçinalp, SAP AG.

This document is also available in these non-normative formats: PDF, PostScript, XML, and plain text.

Abstract

Web Services Policy 1.5 - Guidelines for Policy Assertion Authors is intended to provide guidance for assertion authors that will work with the Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.27]*] and Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment [p.27]*] 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 section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is the First Public Working Draft of the Web Services Policy 1.5 - Guidelines for Policy Assertion Authors specification. This Working Draft was produced by the members of the Web Services Policy Working Group, which is part of the W3C Web Services Activity. The Working Group has not yet decided if it will advance this Working Draft to Recommendation Status.

Several issues have already been filed on this document and are recorded in Bugzilla. The Working Group has not yet considered many of these issues, including issues which might lead to substantive changes like:

- Issue 3988 regarding how to design an assertion
- Issue 3989 regarding the overall format of the Guidelines document
- Issue 4041 regarding coverage of wsp:Ignorable.

Note that this Working Draft does not necessarily represent a consensus of the Working Group. Discussion of this document takes place on the public public-ws-policy@w3.org mailing list (public archive) and within Bugzilla. Comments on this specification should be made following the Description for Issues of the Working Group.

Table of Contents

Appendices

1. Introduction

The WS-Policy specification defines a policy to be a collection of policy alternatives with each policy alternative a collection of policy assertions. The Web Services Policy 1.5 - Framework provides a flexible framework to represent consistent combinations of behaviors from a variety of domains. A policy assertion is a machine readable metadata expression that identifies behaviors required for Web services interactions. *Web Services Policy 1.5 - Guidelines for Policy Assertion Authors* is a resource primarily for assertion authors and 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. To enable interoperability of web services different sets of WS-Policy Assertions need to be defined by different communities based upon domain-specific requirements of the web service.

The focus of these guidelines is to capture best practices and usage patterns for practitioners. 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.

Some of the guidance for authors can also be helpful for:

- 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** [p.25] 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.)

As a companion document to the primer, this document also follows the Socratic style of beginning with a question, and then answering the question.

2. What is an Assertion?

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

Policy assertions representing shared and visible behaviors are useful pieces of metadata to enable interoperability and tooling for automation. The key to understanding when to design policy assertions is to have clarity on the characteristics of a behavior represented by a policy assertion. Some useful ways to discover relevant behaviors are to ask questions like the following:

- Is this behavior a requirement?
- Is the behavior visible?

  A visible behavior refers to a requirement that manifests itself on the wire. Web services provide interoperable machine-to-machine interaction among disparate systems. Web service interoperability is the capability of disparate systems to exchange data using common data formats and protocols supporting characteristics such as messaging, security, reliability and transaction. Such data formats and protocols manifest on the wire. Providers and requesters rely on wire messages conforming to such formats and protocols to achieve interoperability.

If an assertion describes a behavior that does not manifest on the wire then the assertion is not relevant to an interoperable interaction. An example is an assertion that describes the privacy notice information of a provider and the associated regulatory safeguard in place on the provider's side. Such assertions may represent business or regulatory level metadata but do not add any value to interoperability.

If an assertion has no wire or message-level visible behavior then the interacting participants may require some sort of additional mechanism to indicate compliance with the assertion and to enable dispute resolution. Introducing an additional non-repudiation mechanism adds unnecessary complexity to processing a policy assertion.

- Does the behavior apply to two or more Web service participants?

  A shared behavior refers to a requirement that is relevant to an interoperable Web service interaction and involves two or more participants. If an assertion only describes one participant's behavior (non-shared behavior) then the assertion is not relevant to an interoperable interaction. Non-shared behaviors do not add any value for tooling or interoperability. An example of a non-shared behavior is the use of logging or auditing by the provider. Requesters may use the policy intersection to select a compatible policy alternative for a Web service interaction. If an assertion only describes one participant's behavior then this assertion will not be present in the other participants' policy and the policy intersection will unnecessarily produce false negatives.

- Does the behavior have an implied scoping to a policy subject such as service, endpoint, operation and message?

- Is there a requirement that a choice must be made for successful interaction?

  Sometimes providers and requesters are required to engage in certain behaviors. The use of optimization and reliable messaging are two examples.

There are already many examples in the industry that adhere to this practice, such as *Web Services Reliable Messaging Policy [p.28]* and *WS-SecurityPolicy [p.28]* . 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
- 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.

3. Who is involved in authoring 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.

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.

3.1 Roles and Responsibilities

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

3.1.1 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.

When using the WS-Policy Framework, any domain author defining new WS-Policy assertions must adhere to the MUST's and SHOULD's in the specification 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 follows these practices is the WS-SecurityPolicy specification [*WS-SecurityPolicy [p.28]*]. 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 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 partici-

pant 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.

3.1.2 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. This selected alternative is then used 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 [p.27]* , *WSDL 2.0 Core Language [p.27]*], UDDI directory entries [*UDDI API 2.0 [p.28]* , *UDDI Data Structure 2.0 [p.28]* , *UDDI 3.0 [p.28]*], and WS-Addressing Endpoint References (EPR) [*WS-Addressing Core [p.27]*].

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.

3.1.3 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 useful 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. Lifecycle of Assertions** [p.19] and *Web Services Policy Primer [p.27]* that describes service and policy assertion evolution.

4. General Guidelines for WS-Policy Assertion Authors

As authors begin the task of inventing XML dialects to represent policy assertions they can take advantage of WS-Policy building on XML principles and XML Schema validation in their design. WS-Policy relies on the QName of a policy assertion being an XML element but allows authors to optionally provide additional semantics through nesting assertions, or specifying assertion parameters. This section covers several aspects of assertion design and provides some answers to the following questions:

- What is the intended use of the policy assertion?

- Which authoring style will be used?
- Is this a new policy domain? Does it need to compose with other domains?
- How complex are the assertions?
- Is there a need to consider nesting?
- Do optional behaviors need to be represented?

4.1 Assertions and Their Target Use

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 is 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.

Best practice: WS-Policy authors should include the following items in the dialect specification:

- *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.

4.2 Authoring Styles

WS-Policy supports two 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/2006/07/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.

```
<wsp:Policy xmlns:wsp='http://www.w3.org/2006/07/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 `wsp:optional` attribute as our example illustrates above.

Best practice: use the authoring style most appropriate for the target audience

4.3 Considerations when Modeling New Assertions

When creating a new policy domain, it is important to understand how policy expressions are used by a framework implementation that has followed the specifications.

The examples given in this document reference WS-Policy like WS-SecurityPolicy and WS-RM Policy. These policy expressions represent web services message exchange requirements, but policy authoring can be done by individual groups that wish to represent web services application requirements and deployments that wish to reuse the WS-Policy framework in order to enable dynamic negotiation of business requirements and capabilities at runtime.

4.3.1 Minimal approach

New policy authors are encouraged to try to not 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.

If grouping is utilized, 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 [p.28]* to see an example of a relatively simple domain that has defined three assertions. Domain authors are encouraged to look at *WS-SecurityPolicy [p.28]* to see an example of a complex domain that has been decomposed into a set of policy expressions.

How big should an assertion be? How many assertion parameters should the assertion enumerate? How many dependent behaviors should the assertion enumerate? It is always good to start with a simple working policy assertion that allows extensibility. As your design work progresses, you may add more parameters or nested policy assertions to meet your interoperability needs.

Best practice: Start with a simple working assertion that allows extensibility.

4.3.2 QName and XML Information Set representation

Web Services Policy language allows assertion authors to invent their own XML dialects to represent policy assertions. The policy language relies only on the policy assertion XML element QName. This QName is unique and identifies the behavior represented by a policy assertion. Assertion authors have the option to represent an assertion parameter as a child element (by leveraging natural XML nesting) or an attribute of an assertion. The general guidelines on when to use XML elements versus attributes apply.

The syntax of an assertion can be represented using an XML outline (plus an XML schema document). If the assertion has a nested policy expression then the assertion XML outline can enumerate the nested assertions that are allowed.

Best practice: Use a unique QName to identify the behavior and provide an XML outline (plus an XML schema document) to specify the syntax of an assertion.

4.3.3 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.

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, authors need to indicate how the runtime behavior represented in 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 cannot be self describing.

Best practice: Policy assertions should not be used to express the semantics of a message.

4.3.4 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.

The model advocated for new assertion development is a cooperative marketplace [some might say it is 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.

Best practice: Avoid duplication of assertions.

4.4 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.

4.4.1 Assertions with Parameters

The framework allows WS-Policy domain authors to define parameters, 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 4-3. Policy Assertion with Assertion Parameters

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

4.4.2 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.

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 4-4. 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. This means the complexity of security usage is absorbed by a policy-aware client and hidden from Web service application developers.

4.4.3 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. The tradeoff is the generality vs. the flexibility and complexity of the comparison expected for a domain.

The following design questions below can help to determine when to use nested policy expressions:

- Are these assertions designed for the same policy subject?

- Do these assertions represent dependent behaviors?

If the answers are yes to both of these questions then leveraging nested policy expressions is something to consider. Keep in mind that a nested policy expression participates in the policy intersection algorithm. If a requester uses policy intersection to select a compatible policy alternative then the assertions in a nested policy expression play a first class role in the outcome. There is one caveat to watch out for: policy assertions with deeply nested policy can greatly increase the complexity of a policy and should be avoided when they are not needed.

Best practice: If the domain authors want to delegate the processing to the framework, utilizing nesting should be considered. Otherwise, domain specific comparison algorithms will need to be devised and be delegated to the specific domain handlers that are not visible to the WS-Policy framework.

4.5 Designating Optional Behaviors

4.5.1 Optional behavior in Compact authoring

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 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.

4.5.2 Optional behavior at runtime

The *Web Services Policy Primer [p.27]* document contains an example that proposes the use of *MTOM [p.26]* 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 [p.26]* , *XOP [p.27]* 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 behavior 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 [**4.3.3 Self Describing Messages** [p.11]] 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 not specified or incompletely specified 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 describing 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 [p.28]*] 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.

Best Practice: 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 **4.3.3 Self Describing Messages** [p.11] .

4.6 Typing Assertions

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 of the assertion overtime. A namespace associated with the assertion may be used to indicate a specific version of an assertion but this has its limitations. See Lifecycle material **5. Lifecycle of Assertions** [p.19] for more detail.

The typing must be done in combination with the scoping of the semantics to a policy subject. 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.

One way to disambiguate the semantic 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 [p.27]*

Best Practice- To avoid 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.

1. Description must clearly and completely specify the syntax (plus an XML Schema document) and semantics of a policy assertion.
2. If there is a nested policy expression, description must declare it and enumerate the nested policy assertions that are allowed.
3. A policy alternative may contain multiple instances of the same policy assertion. Description must specify the semantics of parameters and nested policy (if any) when there are multiple instances of a policy assertion in the same policy alternative.
4. If a policy assertion is to be used with WSDL, description must specify a WSDL policy subject – such as service, endpoint, operation and message.

4.7 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 [p.27]*

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 a sender chose 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. 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. Lifecycle of 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

  Over time, the Policy WG or third parties can version or extend the Policy Language with new or modified constructs. These constructs may be compatible or incompatible with previous versions.

  Policy authors should review the WS-Policy Primer *Web Services Policy Primer [p.27]* and the specifications *Web Services Policy Framework [p.27] Web Services Policy Attachment [p.27]* for details on extensibility.

  The current WS-Policy language *Web Services Policy Framework [p.27]* provides extensibility points on 6 elements with a combination of attribute and/or element extensibility:

  1. Policy: element from ##other namespace and any attribute
  2. PolicyReference: any attribute and a proposal to add any element ExactlyOne, All: element from ##other namespace, no attribute extensibility
  3. PolicyAttachment: element from ##other namespace and any attribute
  4. AppliesTo: any element and any attribute

- Subject attachment Extensibility

  PolicyAttachment and AppliesTo also have extensibility points.

5.1 Referencing Policy Expressions

The *Web Services Policy Primer [p.27]* 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.

5.2 Evolution of Assertions (Versioning and Compatibility)

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 [p.27]*]. 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 5-1. Message-level Security and WSS: SOAP Message Security 1.0

```
<Policy>
  <sp:Wss10>…</sp:Wss10>
</Policy>
```

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 5-2. Message-level Security and WSS: SOAP Message Security 1.1

```
<Policy>
  <sp:Wss11>…</sp:Wss11>
</Policy>
```

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

6. Inter-domain Policy and Composition Issues

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 affects transport bindings and would result in nested policy assertions when additional protocols are composed with *WS-Security 2004 [p.28]* . 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

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

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 [p.28]*).

8. Scenario and a worked example

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. Company A 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 Company A are instructed to review the current web services at Company A and propose a plan for adding policy assertions.

The application developers collect information about web services within Company A 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 Company A 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. Company A 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. Company A’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 three 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 Company A 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 Company A 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 Company A, 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, Company A 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 (Company A gets bought by Company B 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 Company A’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 Company A has decided to use well known policy expressions that are 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 1.1 document 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.

A. Security Considerations

Security considerations are discussed in the *Web Services Policy Framework [p.27]* 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 [p.27]*]
`sp`	`http://schemas.xmlsoap.org/ws/2005/07/securitypolicy`	[*WS-SecurityPolicy [p.28]*]
`wsa`	`http://www.w3.org/2005/08/addressing`	[*WS-Addressing Core [p.27]*]
`wsdl`	`http://schemas.xmlsoap.org/wsdl/`	[*WSDL 1.1 [p.27]*]
`wsp`	`http://www.w3.org/2006/07/ws-policy`	[*Web Services Policy Framework [p.27] , Web Services Policy Attachment [p.27]*]
`wsrmp`	`http://docs.oasis-open.org/ws-rx/wsrmp/200608`	[*Web Services Reliable Messaging Policy [p.28]*]
`wss`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd`	[*WS-Security 2004 [p.28]*]
`wsu`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`	[*WS-Security 2004 [p.28]*]

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, P. Yendluri and T. Boubez, Editors. World Wide Web Consortium, 17 November 2006. This version of the Web Services Policy 1.5 - Framework specification is at http://www.w3.org/TR/2006/WD-ws-policy-20061117. 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, P. Yendluri and T. Boubez, Editors. World Wide Web Consortium, 17 November 2006. This version of the Web Services Policy 1.5 - Attachment specification is at http://www.w3.org/TR/2006/WD-ws-policy-attach-20061117. 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, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and Ü. Yalçinalp, Editors. World Wide Web Consortium, 21 December 2006. This version of the Web Services Policy 1.5 - Primer specification is at http://www.w3.org/TR/2006/WD-ws-policy-primer-20061221. The latest version of Web Services Policy 1.5 - Primer is available at http://www.w3.org/TR/ws-policy-primer/.

[Web Services Reliable Messaging]
Web Services Reliable Messaging (WS-ReliableMessaging), Doug Davis (IBM), Anish Karmarkar (Oracle), Gilbert Pilz (BEA), Steve Winkler (SAP), Ü. Yalçinalp (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), Ü. Yalçinalp (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.), Paul Knight (Nortel Networks), 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), Bob Natale (MITRE 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 discussions on public-ws-policy@w3.org 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
20061115	MH	First attempt at restructuring to include primer content
20061120	MH	Restructure to address action items 64,77, which refer to bugzilla 3705 and F2F RESOLUTION 3792
20061127	ASV	Updated the list of editors. Added Frederick and Umit to the list of editors. Editors' action 86.
20061128	MH	Replaced section in Lifecycle with pointer to the text in the primer: related to action 77
20061129	FH	Editorial revision (editorial actions 84 and 90) - includes suggestions from Asir: Part 1 and Part 2.
20061129	ASV	Formatted examples in **5.2 Evolution of Assertions (Versioning and Compatibility)** [p.20] .
20061218	FS	Formatted examples in **4.2 Authoring Styles** [p.9] and **8. Scenario and a worked example** [p.21] .