Web Services Policy 1.5 - Guidelines for Policy Assertion Authors

W3C Working Draft 10 August 2007

This version:
http://www.w3.org/TR/2007/WD-ws-policy-guidelines-20070810
Latest version:
http://www.w3.org/TR/ws-policy-guidelines
Previous version:
http://www.w3.org/TR/2007/WD-ws-policy-guidelines-20070330
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.30]*] and Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment [p.30]*] 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 an updated Working Draft of the Web Services Policy 1.5 - Guidelines 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 expects to advance this Working Draft to Working Group Note. There are no open issues against this document, see Bugzilla.

A list of changes in this version of the document [p.32] is available. Major editorial changes in this version of the document encompass a general reformation of the document relying on Best Practices statements, changes related to the alignment with various other specifications, and the removal of two sections.

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.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. The group does not expect this document to become a W3C Recommendation. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

Appendices

---

1. Introduction

The WS-Policy specification defines a policy to be a collection of policy alternatives. 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 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 the Primer. It is intended to provide non-normative guidelines for 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 WS-Policy Assertion Authors can also be helpful for:

- WS-Policy expression authors who need to understand the syntax of the language and understand how to build consistent policy expressions
- 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 Assertion Authors

This document assumes a basic understanding of XML, Namespaces in XML, WSDL, SOAP and the Web Services Policy language.

This is a non-normative document and does not provide a definitive specification of the Web Services Policy framework. **B. XML Namespaces** [p.29] 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. List of Best Practice Statements

The following Best Practices appear in this document with discussion and examples, and are summarized here for quick reference:

3. 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 [*XML Schema Structures [p.32]*].

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 will not impact the interoperability of wire messages, but may still be relevant to enabling an interoperable interaction. For example, a provider may not wish to interact unless a client can accept an assertion describing provider behavior. 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. For cases where the provider does not intend the assertion to impact interoperability it may mark it as ignorable.

  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 the assertion may still be relevant to enabling an interoperable interaction. An example is the use of logging or auditing by the provider. If an assertion only describes one participant's behavior then the assertion may be marked as ignorable (indicating it does not impact interoperability). An ignorable policy assertion is ignored for lax policy intersection. If an assertion is not an ignorable assertion then it is deemed important for agreement between both parties.

- 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 the above practices, such as *Web Services Reliable Messaging Policy Assertion [p.31]* and *WS-SecurityPolicy [p.31]* . 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.

4. 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 assertions 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.

4.1 Roles and Responsibilities

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

4.1.1 Assertion Authors

Assertion Authors are part of a community that chooses to exploit the WS-Policy Framework by creating their own specifications 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 Assertion 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 Assertion Authors 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.

Assertion Authors should also specify a policy subject. For instance, if a policy assertion were to be used with WSDL, an assertion description should specify a WSDL policy subject.

An example of a domain specification that follows these practices is the WS-SecurityPolicy specification [*WS-SecurityPolicy [p.31]*]. The WS-SecurityPolicy authors have defined the scope of their target domain (security) 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 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.

4.1.2 Consumers

A consumer of WS-Policy Assertions can be any entity that is capable of parsing a WS-Policy XML expression 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.30]* , *WSDL 2.0 Core Language [p.30]*], and UDDI directory entries [*UDDI API 2.0 [p.31]* , *UDDI Data Structure 2.0 [p.31]* , *UDDI 3.0 [p.31]*].

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 requesters that are capable of adapting to the constraints and capabilities expressed in a WS-Policy document and modifying their own configurations dynamically.

4.1.3 Providers

A provider who expresses capabilities and requirements of a Web service as policies can be any web service implementation that can specify its on-the-wire message behavior as a policy expression that conforms to the Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.30]*] and Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment [p.30]*] specifications. The Web Services Policy 1.5 - Attachment specification has defined a set of subjects and an extensible mechanism for attaching policies to web services subjects.

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 **6. Versioning Policy Assertions** [p.26] and *Web Services Policy Primer [p.30]* that describes service and policy assertion evolution.

5. General Guidelines for Assertion Authors

As Assertion 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 Assertion 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?

5.1 Assertions and Their Target Use

Assertion Authors should understand the functionality that the WS-Policy framework provides and apply the knowledge of the policy framework processing when defining the set of assertions.

Assertions can be simple or they can be complex. Assertion Authors may choose to specify multiple peer assertions, each carrying the semantic of a particular behavior, or they may choose to specify assertions that contain assertion parameters and/or nested policy expressions (nested assertions), where each nested assertion of which relates to an aspect of the behavior, yet encapsulated within a single assertion. There are advantages to simplifying the set of assertions. The ultimate goal of policy is to enable interoperability. By keeping assertion design as simple as possible, Assertion Authors will more likely be able to meet that objective.

Assertion Authors need to have a specific goal in mind for the assertions that they author. Assertion specifications should include a detailed specification of the assertion's semantics and, a set of valid policy subjects to which the assertion maybe attached. The specification should also include the scope of the assertion in the context of a particular policy subject. For example, an assertion with Endpoint Policy Subject can be scoped to a given message exchange with that endpoint, or it can be scoped to all messages exchanged with that endpoint. The former case permits a client to select a different alternative with each successive message exchange. Finally, the ability to combine individual assertions may also need to be considered. 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.

Assertion Authors should include the following items in an assertion specification:

- The definition of the assertion's semantic (See best practice **8. Specify Semantics Clearly** [p.14]).
- The specification of the set of valid policy subjects to which an assertion may be attached (See best practice **24. Specify Policy Subject(s)** [p.24]).
- The scope of the assertion in the context of a particular policy subject (See best practices in Section **5.7 Considerations for Policy Attachment** [p.23]).
- Any composition considerations if the assertion is used with other assertions in a context (See best practice **29. Specify Composition with Related Assertions** [p.26]).

The WS-Policy Attachment specification defines a number of different policy subjects to which an assertion can be attached. For attaching to WSDL subjects see **5.7 Considerations for Policy Attachment** [p.23] for more detail. Additionally, the framework provides for the means to extend the set of policy subjects beyond the set of subjects defined in the WS-Policy Attachment specification.

Although a policy assertion may be constrained to a specific set of policy subjects by Assertion Authors, its semantics should not be dependent upon the mechanism by which the policy expression is attached to a given policy subject. For instance, an assertion "Foo" has the same semantic when attached to an operation policy subject regardless of whether it was attached using XML element policy attachment or the external URI attachment mechanism. Independence from a specific attachment mechanism allows policy tools to choose the most appropriate mechanism to attach a policy without having to analyze the contents of the policy.

Best Practice 1: Semantics Independent of Attachment Mechanisms

The semantics of a policy assertion should not depend on the attachment mechanism used.

Best Practice 2: Define assertions relevant to compatibility tests

Assertion authors should define assertions for behaviors that are relevant to compatibility assessment, such as web service protocols that manifest on the wire.

Assertion authors may define assertions that are not related to compatibility assessment. These assertions may be used to accurately describe behaviour, even if they do not affect compatibility. WS-Policy has the wsp:Ignorable attribute that may be used for indicating assertions that are not related to compatibility assessment, described in **5.5 Designating Ignorable Behavior** [p.20]

Best Practice 3: Mark Ignorable Assertions not related to compatibility

Assertion authors should recommend that assertions that are not relevant to compatibility assessment be marked with the wsp:Ignorable attribute.

5.2 Authoring Styles

WS-Policy supports two different authoring styles, compact form and normal form. 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. A policy expression in the compact form can be translated into its normal form using the policy normalization algorithm described in the Web Service Policy Framework (see section 4.3 Compact Policy Expression).

The two forms of a policy expression are semantically equivalent. When multiple alternatives are present in a policy, the normal form may express the choices more explicitly. On the other hand, the compact form may be more readable for humans when an assertion is marked as optional using the `wsp:optional` attribute. A policy processor may normalize a policy expression originally authored in compact form at any time without changing the semantics of the policy. In general, it is not possible to guarantee in what form a policy expression would be when it is processed. As a result, the description for a policy assertion should not depend on the style used to author a policy expression that contains the assertion.

Best Practice 4: Semantics Independent of the Form

The semantics of an assertion should be independent of the form (compact or normal form) of policy expressions that contain the assertion.

In the example below, the policy expression is shown in its two forms, compact and normal. In compact form, the `wsrmp:RMAssertion` assertion is augmented by the `wsp:Optional="true"` attribute. While the compact form of the expression might be more human readable, the semantics of the particular assertion are independent of the form and of the presence (or absence) of the `wsp:optional` attribute.

Example 5-1. Policy Expression in Compact Form

```
<wsp:Policy xmlns:wsp='http://www.w3.org/ns/ws-policy'
        xmlns:sp='http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702'
        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>
                    <wsp:Policy>
                     <sp:RequireClientCertificate/>
                    </wsp:Policy>
                  </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Example 5-2. Policy Expression in Normal Form

```
<wsp:Policy xmlns:wsp='http://www.w3.org/ns/ws-policy'
        xmlns:sp='http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702'
        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>
                    <wsp:Policy>
                      <sp:RequireClientCertificate/>
                    </wsp:Policy>
                  </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:All>

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

5.3 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.

5.3.1 Minimal approach

New Assertion 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.

If grouping is utilized, choices between such groupings can be indicated by an "exactly one" operator. This basic set of operators allows Assertion 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. To facilitate proper progression of an assertion, Assertion Authors should start with a simple working assertion that allows extensibility. As the design work progresses, one may add more parameters or nested policy assertions to meet one’s interoperability needs.

Best Practice 5: Start with a Simple Assertion

Assertion Authors should start with a simple working assertion that allows assertion parameter extensibility.

New Assertion Authors are encouraged to look at *Web Services Reliable Messaging Policy Assertion [p.31]* to see an example of a relatively simple domain that has defined three assertions. Assertion Authors are encouraged to look at *WS-SecurityPolicy [p.31]* to see an example of a complex domain that has been decomposed into a set of policy expressions.

5.3.2 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. Use a unique QName to identify a distinct behavior.

Best Practice 6: Use Unique QNames

Assertion Authors should use a unique QName to identify a distinct behavior.

Best Practice 7: Provide an XML definition

Assertion authors should provide an XML schema definition to specify the syntax of an assertion. A reader-friendly description such as an XML outline (see below) is also useful.

An example of a specification that provides an XML Outline is the Web Services Reliable Messaging Policy document [*Web Services Reliable Messaging Policy Assertion [p.31]*]. The definition of the outline syntax used in that specification is found in its Terminology section (1.1). As an example of the outline syntax in use, the following outline has been copied from the aforementioned specification.

```
<wsrmp:RMAssertion [wsp:Optional="true"]? ...>
  <wsp:Policy >
    [ <wsrmp:SequenceSTR/> |
      <wsrmp:SequenceTransportSecurity/> ] ?
    <wsrmp:DeliveryAssurance/>
    <wsp:Policy >
       [ <wsrmp:ExactlyOnce/> |
         <wsrmp:AtLeastOnce/> |
         <wsrmp:AtMostOnce/> ]
       <wsrmp:InOrder/> ?
    </wsp:Policy>
    </wsrmp:DeliveryAssurance> ] ?
   </wsp:Policy>
</wsrmp:RMAssertion/>
```

The 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. An example is the following:

```
<sp:IssuedToken sp:IncludeToken="xs:anyURI"? ... >
 <sp:Issuer> wsa:EndpointReferenceType</sp:Issuer>?
 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
   ...
 </sp:RequestSecurityTokenTemplate >
 <wsp:Policy >
   <sp:RequireDerivedKeys /> ?
   <sp:RequireExternalReference /> ?
   <sp:RequireInternalReference /> ?
   ...
 </wsp:Policy> ?
 ...
</sp:IssuedToken>
```

Best Practice 8: Specify Semantics Clearly

Assertion authors should clearly and completely specify the semantics of a policy assertion.

Best Practice 9: Document Ignorable Behavior

An assertion description should include guidance as to the use of (or constraint against the use of) the wsp:Ignorable attribute to indicate whether or not the behavior indicated by the QName may be ignored by policy intersection.

Best Practice 10: Document Use of the Ignorable Attribute in XML

An Assertion Author should document, in the XML outline and/or schema for the assertion, whether or not the assertion allows for the use of the wsp:Ignorable attribute.

To give an example, the WS-ReliableMessaging Policy document specifies attribute extensibility as part of the XML definition, allowing the wsp:Ignorable attribute:

Example 5-5. WS-ReliableMessaging Policy use of attribute extensibility

```
/wsrmp:RMAssertion/@{any}
This is an extensibility mechanism to allow different {extensible} types of information, based on a schema, to be passed.
```

The Policy Framework provides two modes of authoring policy expressions: compact and normal form. One of the mechanisms that the Policy Framework provides to policy authors for purposes of writing compact policy expressions is the wsp:Optional attribute. Assertion Authors should allow for the use of the wsp:Optional attribute in the XML outline and/or schema definition of an assertion as this will allow policy expression authors to compose compact policy expressions.

Best Practice 11: Allow use of wsp:Optional

An assertion's XML outline and/or schema definition should allow the use of the wsp:Optional attribute so as to enable policy authors to compose compact policy expressions.

For example, consider the following two equivalent policy expressions:

Example 5-6. Normal form expression:

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
    <wsam:Addressing>
        <wsp:Policy/>
    </wsam:Addressing>
    </wsp:All>
    <wsp:All>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Example 5-7. Compact form expression:

```
<wsp:Policy>
    <wsam:Addressing wsp:Optional="true">
        <wsp:Policy/>
    </wsam:Addressing>
</wsp:Policy>
```

If the assertion author had not provided for the wsp:Optional attribute to be included on the assertion, then policy expression authors would be forced to express the optionality of a behavior as two explicit policy alternatives, one with and one without that assertion when including assertions of that type in their policies.

5.3.3 Self Describing Messages

WS-Policy is intended to communicate the requirements, capabilities 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. Note that there are other specifications that target specification of semantics of a message, such as *SAWSDL [p.31]* .

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.

Policy assertions should not be used to express the semantics of a message. Firstly, an assertion type indicates a *runtime* behavior. Secondly, Assertion 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.

Best Practice 12: Define message format only

Assertion Authors should not define policy assertions to represent information that is necessary to understand a message.

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.

5.3.4 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 a community supporting implementations of these capabilities 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 Assertion 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.

It is the responsibility of the Assertion Authors to avoid duplication of assertions. 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 13: Avoid Duplication of Assertions

Assertion Authors should reuse an existing assertion (rather than create a new one) whenever possible.

5.4 Comparison of Nested and Parameterized Assertions

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

The main consideration for choosing between use of parameters or nested policy expressions is that the framework intersection algorithm processes nested policy expressions, but does not consider parameters in the algorithm.

5.4.1 Assertions with Parameters

Policy assertion parameters are the opaque payload of an assertion. Parameters carry additional useful information for engaging the behavior described by an assertion and are preserved through policy processing such as normalization, merge and policy intersection. Requesters may use policy intersection to select a compatible policy alternative for an interaction. Assertion parameters do not affect the outcome of policy intersection unless the assertion specifies domain specific processing for policy intersection.

In the XML representation of a policy assertion, the child elements and attributes of the assertion excluding child elements and attributes from the policy language namespace name are the assertion parameters.

Best Practice 14: Use Parameters for Useful Information

Assertion Authors should represent useful additive information necessary for engaging the behavior represented by a policy assertion as assertion parameters.

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). These two parameters identify the parts of a wire message that should be protected. These parameters carry additional useful information for engaging the behavior.

Example 5-8. Policy Assertion with Assertion Parameters

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

5.4.2 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. In particular, when assertion authors define an assertion type that allows nested policy expression, it is important to also define the semantics of that assertion when it contains an empty nested policy expression (for example: <wsam:Addressing><wsp:Policy/></wsam:Addressing>).

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. If there is a nested policy expression, an assertion description should declare it and enumerate the nested policy assertions that are allowed. 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 15: Use Nested Assertions for Dependent Behaviors

Assertion Authors should represent dependent behaviors that are relevant to a compatibility test and apply to the same policy subject using nested policy assertions.

Best Practice 16: Enumerate Nested Assertions

If there is a nested policy expression, then the Assertion Authors should enumerate the nested policy assertions that are allowed.

Assertion 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.

If the assertion authors want to delegate the processing to the framework, utilizing nesting should be considered. Otherwise, domain specific comparison algorithms may need to be devised and be delegated to the specific domain handlers that are not visible to the WS-Policy framework. However, domain specific intersection processing reduces interop and increases the burden to implement an assertion.

Best Practice 17: Discourage Domain Specific Intersection

Assertion authors should only specify domain specific intersection semantics when policy intersection is insufficient.

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

Securing messages is a complex usage scenario. The WS-SecurityPolicy Assertion 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 5-9. Transport Security Policy Assertion

```
<sp:TransportBinding>
  <Policy>
    <sp:TransportToken>
      <Policy>
        <sp:HttpsToken>
          <wsp:Policy/>
        </sp:HttpsToken>
      </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.

Assertion Authors should note the effect of nested policy expressions on policy intersection in their nested policy design. The result of intersecting an assertion that contains an empty nested policy expression with an assertion of the same type without a nested policy expression is that the assertions are not compatible. Therefore, when providers require dependent behaviors these behaviors should be explicitly specified as assertions in a nested policy expression. When the definition of an assertion allows for nested dependent behaviors, but the use of the assertion only contains an empty nested policy expression, this specific use indicates the specification of no nested dependent behaviors. This use must not be interpreted as being compatible with "any" of the nested dependent behaviors that are allowed by the assertion, unless otherwise specified by the assertion definition.

As an example, WS-Security Policy defines `sp:HttpToken` assertion to contain three possible nested elements, `sp:HttpBasicAuthentication`, `sp:HttpDigestAuthentication` and `sp:RequireClientCertificate`. When the `HttpToken` is used with an empty nested policy in a policy expression by a provider, it will indicate that none of the dependent behaviors namely authentication or client certificate is required.

Example 5-10. Empty Nested Policy Expression

```
<sp:TransportToken>
  <wsp:Policy>
        <sp:HttpsToken>
          <wsp:Policy/>
        </sp:HttpsToken>
  </wsp:Policy>
</sp:TransportToken>
```

A non-anonymous client who requires authentication or client certificate will not be able to use this provider solely on the basis of intersection algorithm alone.

5.5 Designating Ignorable Behavior

5.5.1 Ignorable behavior in authoring

The Policy Framework provides an intersection algorithm that has two defined modes for processing (lax and strict). The Framework also defines an attribute (wsp:Ignorable) that can be used to influence whether assertions are part of the compatability assessment between two alternatives. [see *Web Services Policy Framework [p.30]* and *Web Services Policy Primer [p.30]*]. Assertion authors should consider whether the behavior represented by the Assertion they are defining can be safely ignored for the purposes of intersection, and should follow **9. Document Ignorable Behavior** [p.15] and **10. Document Use of the Ignorable Attribute in XML** [p.15] to include this guidance in the assertion's definition.

5.5.2 Ignorable behavior at runtime

Regardless of whether the assertion allows the ignorable attribute, assertion authors should indicate the semantic of the runtime behavior in the description of the assertion.

As said in section 3.4.1 Strict and Lax Policy Intersection in *Web Services Policy Primer [p.30]* , "Regardless of the chosen intersection mode, ignorable assertions do not express any wire-level requirements on the behavior of consumers - in other words, a consumer could choose to ignore any such assertions that end up in the resulting policy after intersection, with no adverse effects on runtime interactions." Therefore, any assertion that is marked with ignorable should not impose any wire-level requirements on the part of consumers. Assertion Authors are reminded that regardless of whether an assertion is marked as ignorable, policy consumers using strict intersection will not 'ignore' the assertion.

5.6 Designating Optional Behaviors

5.6.1 Optional behavior at runtime

Editorial note	
This section does not have Working Group consensus and there is an outstanding action item to revise it	

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 this would allow the consumer to select the policy alternative that does not contain the assertion, and thus not engaging the behavior.

Best Practice 18: Limit use of Optional Assertions

Assertion Authors should not use optional assertions for behaviors that must be present in compatible policy expressions.

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 Assertion Authors to choose the appropriate level of granularity for optional behaviors, to consider whether a specific message or all messages, etc. are targeted.

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. 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. Similarly, if engagement of a behavior is only specified for an outbound message, the 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 Assertion [p.31]*] 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 19: Consider entire message exchange pattern when specifying Assertions that may be optional

Assertion Authors should associate optional assertions with the appropriate endpoint and use the smallest possible granularity to limit the degree to which optionality applies.

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 need for self describing messages [**5.3.3 Self Describing Messages** [p.16]] should not be forgotten. An explicit, out of band mechanism might be necessary to enable a client to indicate that the optional behavior is engaged. (Such an out of band mechanism is outside the scope of WS-Policy Framework).

Best Practice 20: Indicate use of an Optional Assertion

When a given behavior may be optional, it must be possible for both message participants to determine that the assertion is selected by both parties, either out of band or as reflected by the message content.

The *Web Services Policy Primer [p.30]* document contains an example that outlines the use of *MTOM [p.29]* as an optional behavior that can be engaged by a consumer. Related to this behavior is an assertion that identifies the use of MIME Multipart/Related serialization [*MTOMPolicy [p.29]*]. Policy-aware clients that recognize and engage this policy assertion will use Optimized MIME Serialization for messages.

The semantics of the MTOM assertion declare that the behavior must be reflected in messages by requiring that they use an obvious wire format (MIME Multipart/Related serialization). Thus, this optional behavior is self describing. For example, an inbound message to a web service that requires MTOM must adhere to Optimized MIME Serialization. By examining the message, the provider can determine whether the policy alternate that contains the MTOM assertion is being obeyed (Best Practice: Indicate use of an Optional Assertion [p.22]).

Note that if a MTOM assertion were only bound to an inbound message endpoint, then it would not be clear whether the outbound message from the provider would also utilize the behavior. Thus this assertion should be associated at the granularity of an entire message exchange. The semantics of the assertion should specify this to avoid inappropriate assumptions by implementations. This is important for an optional assertion where it may not be clear whether it is to apply in a message exchange when optionally used in part of that exchange (Best Practice: Consider entire message exchange pattern when specifying Assertions that may be optional [p.22]).

5.7 Considerations for Policy Attachment

5.7.1 General Guidelines

The Policy attachment mechanism used to communicate the policy assertions should not affect or imply additional semantics in 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.

Best Practice 21: Leverage Defined Attachment Mechanisms

Assertion Authors should leverage defined attachment models when possible to extend the deployment of their policy assertions and ensure that there are no additional semantics implied by their assertions.

Assertion authors are also encouraged to use the policy subjects defined by the policy attachments specification when possible.

Best Practice 22: Use Defined Policy Subjects

Assertion Authors should leverage defined policy subjects when possible to facilitate the deployment of their policy assertions. Common Policy subjects have been defined and used by other policy assertion authors and new policy assertions that leverage these existing subjects will be easier to define and group.

Policy assertion authors should unambiguously identify the appropriate policy subjects for their assertions. 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.

Best Practice 23: Identify Policy Subjects

Assertion Authors should review the policy subjects defined in WS-PolicyAttachments and identify existing policy subjects when possible to facilitate the deployment of their policy assertions and include this information in the definition of the assertions.

An example of this is the Reliable Messaging Policy Assertion document [*Web Services Reliable Messaging Policy Assertion [p.31]*]. In the Sequence STR Assertion (section 2.5.1) the Reliable Messaging Policy Assertion authors state that "The STR assertion defines the requirement that an RM Sequence MUST be bound to an explicit token that is referenced from a `wsse:SecurityTokenReference` in the CreateSequence message. This assertion MUST apply to [Endpoint Policy Subject]. This assertion MUST NOT be used for an endpoint that does not also use the RM assertion". This is illustrative of how the domain assertion author can specify additional constraints and assumptions for attachment and engagement of behavior in addition to the capabilities specified in WS-PolicyAttachment [*Web Services Policy Attachment [p.30]*]. Such additional constraints must be clearly specified by the assertion authors.

5.7.2 Considerations for Policy Attachment 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, Assertion Authors should specify a WSDL policy subject.

The specific WSDL 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.

Best Practice 24: Specify Policy Subject(s)

Assertion Authors should specify the set of relevant policy subjects with which the assertion may be associated. For instance, if a policy assertion is to be used with WSDL, the assertion description should specify a WSDL policy subject - such as service, endpoint, operation and message.

Assertion Authors that wish to utilize WSDL policy subjects need to understand how the assertions will be processed in an 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.30]*

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. Assertion Authors should identify the relevant attachment point when defining a new assertion. To determine the relevant attachment points, Assertion 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 so on. 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. However such policy attachments to policy subjects of broader scope and granularity should be done only after careful evaluation.

Best Practice 25: Choose the Most Granular Policy Subject

Assertion Authors should choose the most granular policy subject to which the behavior represented by a policy assertion applies.

For authoring convenience, Assertion Authors may allow the association of an assertion to multiple policy subjects within the same context of use (e.g in the same WSDL description). If an assertion is allowed to be associated with multiple policy subjects as is possible with WSDL, then the Assertion Authors have the burden to describe the rules when multiple instances of the same assertion are attached to different policy subjects in order to avoid non-interoperable behavior.

Best Practice 26: Define Rules for Attachment of an Assertion type to Multiple Policy Subjects

If an assertion is allowed to be associated with multiple policy subjects, the assertion author should describe the rules for multiple instances of the same assertion attached to multiple policy subjects in the same context.

To give one example, section 2.3 of the Web Services Reliable Messaging Policy Assertion specification [*Web Services Reliable Messaging Policy Assertion [p.31]*] gives rules on which Policy Subjects may be associated with the RM Policy assertion, and which WSDL 1.1 elements may have RM Policy assertions attached.

If the capability 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.

Since many attachment points are available in WSDL, it would be necessary for Assertion Authors to recommend a preferred attachment point. One approach would be to identify different attachment points in a policy subject, choose the most granular policy subject that the behavior applies to and specify that as a preferred attachment point. 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, as described previously the WS-RM Policy is a capability that governs a target endpoint's capability to accept message sequences that are beyond single message exchange. Therefore, its semantics encompass the cases when message level policy subjects may be used as attachment but also considers the case 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.

Best Practice 27: Specify Preferred Attachment Point

If an assertion can be attached at multiple attachment points within a policy subject, Assertion Authors should specify a preferred attachment point for the chosen policy subject.

Assertion Authors that utilize WSDL policy subjects need to understand how the assertions will be processed in merging and the specify implications of ending up with multiple assertions of the same kind in an alternative, in the merged policy. For example, consider the SignedParts assertion defined in WS-SecurityPolicy 1.2. The definition of SignedParts assertion explicitly permits multiple SignedParts assertions to be present within a policy alternative, and declares it to be equivalent to a single SignedParts assertion containing the union of all specified message parts. So, if a SignedParts assertion is specified in a WSDL binding at the input message level and subsequently an additional SignedParts assertion is specified at the WSDL endpoint policy subject level, then the effective policy at the endpoint could have more than one SignedParts assertion in the same alternative. However, the clear semantics defined by the SignedParts assertion enable processing of the multiple occurrences properly.

Best Practice 28: Describe Semantics of Multiple Assertions of Same Type

A policy alternative can contain multiple instances of the same policy assertion type. Assertion authors should specify the semantics of multiple instances of same policy assertion type in the same policy alternative and the semantics of parameters and nested policy (if any) when there are multiple instances of a policy assertion type in the same policy alternative.

5.8 Interrelated domains

Assertion Authors need to be clear about how assertions defined in their domain may fit with assertions for interrelated domains. Assertion Authors should not duplicate existing assertions and should also make sure that when adding assertions those new assertions are consistent with pre-existing assertions of any interrelated domain.

Best Practice 29: Specify Composition with Related Assertions

Assertion authors should clearly specify how an assertion may compose with other related assertions, if any.

A classic example of such an interrelated domain is security, because security tends to cut across all aspects of a solution. Web Services Reliable Messaging Policy Assertions [*Web Services Reliable Messaging Policy Assertion [p.31]*] defines additional assertions related to [*WS-SecurityPolicy [p.31]*], an interrelated security domain. One such additional assertion specifies the use of transport security to protect a message sequence, for example.

Example 5-11. Reliable Message Sequence Security

```
<wsrmp:SequenceTransportSecurity [wsp:Optional="true"]? ... />
```

The Reliable Message Policy specification states "This assertion is effectively meaningless unless it occurs in conjunction with the `RMAssertion` and a `sp:TransportBinding` assertion that requires the use of some transport-level security mechanism (e.g. `sp:HttpsToken`)".

6. Versioning Policy Assertions

Assertion 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 to versioning policy assertions:

- 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.

  Assertion Authors should review the WS-Policy Primer *Web Services Policy Primer [p.30]* and the specifications *Web Services Policy Framework [p.30] Web Services Policy Attachment [p.30]* for details on extensibility.

  The current WS-Policy language *Web Services Policy Framework [p.30]* 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. ExactlyOne, All: element from ##other namespace; no attribute extensibility
  3. PolicyReference: any element and any attribute
  4. PolicyAttachment: element from ##other namespace and any attribute
  5. AppliesTo: any element and any attribute
  6. URI: any attribute

- Supporting New Policy Subjects

6.1 Referencing Policy Expressions

The *Web Services Policy Primer [p.30]* 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. Reuse may also be useful for domain Assertion Authors, especially those defining complex assertions utilizing references to policy expressions by nesting. Statically available parameterized content may also be reused by different assertions. However, such referencing mechanism is outside the scope of WS-Policy naming and referencing framework and other mechanisms could be used. As an example, in *Web Services Policy Primer [p.30]* Section 4.2, the `sp:issuedToken` assertion utilizes the `sp:RequestSecurityTokenTemplate` parameter that contains necessary information to request a security token. The contents of the parameter are static and allows reuse in different security scenerios.

6.2 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.30]*]. These equivalent behaviors are mutually exclusive for an interaction. Such equivalent behaviors can be modeled as independent assertions.

Best Practice 30: Independent Assertions for Different Versions of a Behavior

Assertion Authors should use independent assertions for modeling different versions of a behavior.

The policy expression in the example below requires the use of WSS: SOAP Message Security 1.0.

Example 6-1. 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 6-2. Message-level Security and WSS: SOAP Message Security 1.1

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

6.3 Supporting New Policy Subjects

The best practice **24. Specify Policy Subject(s)** [p.24] specifies that policy authors should define the set of policy subjects to which policy assertions can be attached. Over time, new policy subjects may need to be defined. When this occurs, policy Assertion Authors may update the list of policy subjects supported by an assertion.

When the assertion's semantics do not change to invalidate any of the original policy subjects but new policy subjects need to be added, it may be possible to use the same assertion to designate the additional policy subjects without a namespace change. For example, a policy assertion for a protocol that is originally designed for endpoint policy subject may add message policy subject to indicate finer granularity in the attachment provided that endpoint policy subject is also retained in its design. When new policy subjects are added it is incumbent on the authors to retain the semantic of the policy assertion.

Best Practice 31: Document changes to policy subject

If the policy subjects change over time, the assertion description should also be versioned to reflect this change.

A. Security Considerations

Security considerations are discussed in the *Web Services Policy Framework [p.30]* 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.30]*]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	[*WS-SecurityPolicy [p.31]*]
wsa	http://www.w3.org/2005/08/addressing	[*WS-Addressing Core [p.30]*]
wsam	http://www.w3.org/2007/05/addressing/metadata	[*WS-Addressing Metadata [p.30]*]
wsdl	http://schemas.xmlsoap.org/wsdl/	[*WSDL 1.1 [p.30]*]
wsp	http://www.w3.org/ns/ws-policy	[*Web Services Policy Framework [p.30]* , *Web Services Policy Attachment [p.30]*]
wsrmp	http://docs.oasis-open.org/ws-rx/wsrmp/200608	[*Web Services Reliable Messaging Policy Assertion [p.31]*]
wss	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[*WS-Security 2004 [p.31]*]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[*WS-Security 2004 [p.31]*]

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/.

[MTOMPolicy]
MTOM Serialization Policy Assertion (WS-MTOMPolicy), C Ferris, K Gavrylyuk, J Marsh , J Schlimmer, Authors. September 2006. Version 1.0 at http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization/optimizedmimeserialization-policy.pdf.

[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.

[WS-Addressing Metadata]
Web Services Addressing 1.0 - Metadata, M. Gudgin, M. Hadley, T. Rogers and Ü. Yalçinalp, Editors. World Wide Web Consortium, 31 July 2007. This version of Web Services Addressing 1.0 - Metadata is http://www.w3.org/TR/2007/PR-ws-addr-metadata-20070731/. The latest version of Web Services Addressing 1.0 - Metadata is available at http://www.w3.org/TR/ws-addr-metadata.

[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, 26 June 2007. This version of the WSDL 2.0 specification is http://www.w3.org/TR/2007/REC-wsdl20-20070626/. 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, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and Ü. Yalçinalp, Editors. World Wide Web Consortium, 6 July 2007. This version of the Web Services Policy 1.5 - Framework specification is at http://www.w3.org/TR/2007/PR-ws-policy-20070706/. 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, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and Ü. Yalçinalp, Editors. World Wide Web Consortium, 6 July 2007. This version of the Web Services Policy 1.5 - Attachment specification is at http://www.w3.org/TR/2007/PR-ws-policy-attach-20070706/. 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, 10 August 2007. This version of Web Services Policy 1.5 - Primer specification is http://www.w3.org/TR/2007/WD-ws-policy-primer-20070810/. 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), D. Davis, A. Karmarkar G. Pilz, S. Winkler, Ü. Yalçinalp, Editors. Organization for the Advancement of Structured Information Standards, 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 Assertion]
Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.1, D. David, A. Kamarkar, G. Pilz, and Ü. Yalçinalp, Editors. Organization for the Advancement of Structured Information Standards, OASIS Standard, 14 June 2007. This version available at
http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01.pdf.

[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 v1.2, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist, Editors. Organization for the Advancement of Structured Information Standards, 1 July 2007. Available at
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf.
Namespace document available at http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702.

[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.

[SAWSDL]
Semantic Annotations for WSDL and XML Schema Joel Farrell, Holger Lausen, Editors. World Wide Web Consortium, 26 January 2007. This version of the specification is at
http://www.w3.org/TR/sawsdl. The latest version of Semantic Annotations for WSDL and XML Schema is available at http://www.w3.org/TR/sawsdl/.

[XML Schema Datatypes]
XML Schema Part 2: Datatypes Second Edition, P. Byron and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001, revised 28 October 2004. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2004/REC-xmlschema-2-20041028. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2.

[XML Schema Structures]
XML Schema Part 1: Structures Second Edition, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 2 May 2001, revised 28 October 2004. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2004/REC-xmlschema-1-20041028. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1.

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), Symon Chang (BEA Systems, Inc.), Paul Cotton (Microsoft Corporation), Glen Daniels (Progress Software), Doug Davis (IBM Corporation), Jacques Durand (Fujitsu Limited), Ruchith Fernando (WSO2), Christopher Ferris (IBM Corporation), William Henry (IONA Technologies, Inc.), Frederick Hirsch (Nokia), Maryann Hondo (IBM Corporation), Ondrej Hrebicek (Microsoft Corporation), Steve Jones (Layer 7 Technologies), Tom Jordahl (Adobe Systems Inc.), Paul Knight (Nortel Networks), Philippe Le Hégaret (W3C/MIT), Mark Little (JBoss Inc.), Mohammad Makarechian (Microsoft Corporation), Ashok Malhotra (Oracle Corporation), Jonathan Marsh (WSO2), Monica Martin (Sun Microsystems, Inc.), Arnaud Meyniel (Axway Software), Jeff Mischkinsky (Oracle Corporation), Dale Moberg (Axway Software), Anthony Nadalin (IBM Corporation), David Orchard (BEA Systems, Inc.), Sanjay Patil (SAP AG), Manjula Peiris (WSO2), Fabian Ritzmann (Sun Microsystems, Inc.), Daniel Roth (Microsoft Corporation), Tom Rutt (Fujitsu Limited), Sanka Samaranayake (WSO2), Felix Sasaki (W3C/Keio), Yakov Sverdlov (CA), Asir Vedamuthu (Microsoft Corporation), Sanjiva Weerawarana (WSO2), Ümit Yalçinalp (SAP AG), Prasad Yendluri (webMethods, Inc.).

Previous members of the Working Group were: Jeffrey Crump, Jong Lee, Bob Natale, Eugene Osovetsky, Bijan Parsia, Skip Snow, Seumas Soltysik, Mark Temple-Raston.

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 (and major editorial) changes since the Working Draft dated 30 March, 2007 is below:

- Reformatted the document to follow the model of the Architecture of the World Wide Web, Volume One (issue 3989).
- Created a consolidated list of Best Practices at the beginning of the document (**2. List of Best Practice Statements** [p.4]) (issue 3989).
- Incorporated the Best Practices from IBM and Microsoft Contribution (issue 3989).
- Made editorial changes to align with the OASIS WS-SecurityPolicy specification.
- Made editorial changes to align with the W3C WS-Addressing 1.0 Metadata specification.
- Reorganized the guidelines on XML Information Set representation.
- Dropped Section 6 Applying Best Practices for Policy Attachment (issue 3978).
- Dropped Section 7 Scenario and a worked example (issue 3988).

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	FJH	Editorial revision (editorial actions 84 and 90) - includes suggestions from Asir: Part 1 and Part 2.

20061129	ASV	Formatted examples in **6.2 Evolution of Assertions (Versioning and Compatibility)** [p.27] .
20061218	FS	Formatted examples in **5.2 Authoring Styles** [p.11] and scenario section.
20061219	TIB	Editorial revision: most parts of editorial action 96. Remaining editorials to be reviewed.
20061220	TIB	Editorial revision: completed missing parts of editorial action 96 after editorial reviews by co-editors.
20061226	MH	Editorial revision: reconciled terms related to "Assertion Authors" 106 and bug http://www.w3.org/Bugs/Public/show_bug.cgi?id=3983
20070104	UY	Resolution of Issue 3982 Based on Minutes for resolution, Minor formatting for consistent use of the term "Assertion Author"
20070104	UY	Resolution of Issue 3980
20070108	ASV	Reset Section **E. Changes in this Version of the Document** [p.32] .
20070122	PY	Completed action item: 127 Resolution for issue 4197
20070130	UY	Completed action item: 144. Resolution for issues 3985 and 3986
20070130	UY	Completed action item: 137. Resolution for issue 4198
20070130	UY	Completed action item: 119. Resolution for issue 4141
20070130	UY	Completed action item: 126. Resolution for issue 4188
20070130	UY	Fixed SAWSDL ref name
20070131	FJH	Fixed numerous spelling and typo errors. Implement resolution for issue 3953 as noted in message 90 and amended as noted in message 217. Changes correspond to editor’s action 152.
20070221	MH	Partial implementation for issue 4072 in response to editor’s action 154 . NOTE ALSO- I needed to put back in the "prefix" entity defintion [line7] to get the build to work.
20070306	ASV	Implemented partial resolution for issue 3987. Related editorial action is 153.
20070308	DBO	Changed "lifecycle" spec references to versioning to fix build.
20070314	FJH	Implemented resolution for issue 4072 as outlined in proposal. Editorial action 204.
20070314	FJH	Implemented resolution for issue 3987 as outlined in proposal. Editorial action 203.
20070315	ASV	Implemented the resolution for issue 3979. Editors’ action 198.

20070315	ASV	Implemented the resolution for issue 3981. Editors' action 205.
20070315	FJH	Implemented resolution for issue 4035 as outlined in proposal. Editorial action 197.
20070319	MH	Implemented resolution for issue 4073 in response to editor's action 199 as outlined in proposal .
20070320	ASV	Implemented the resolution for issue 4319. Editors' action 206.
20070320	ASV	Implemented the resolution for issue 3990. Editors' action 210.
20070320	ASV	Implemented the resolution for issue 4212. Editors' action 207.
20070321	ASV	Updated section **E. Changes in this Version of the Document** [p.32] .
20070329	DBO	Changed all <p>Best Practice: to <p role="practice">
20070416	DBO	Updated 6.2 and 6.3 for issue 3989. Note, removed one best practice that was a dup.
20070423	FJH	Updated 5.5 Designating Optional Behaviors for issue 3989. Added informative reference for MTOMPolicy. Added two best practices, one is similar to G16 but focused on optional. Revised practice that was there.
20070425	MH	Updated 5.3 "Considerations when Modeling New Assertions" related to issue 3989. [Editorial Action 229] Restructured text to follow examples
20070425	TIB	Updated 5.2 Authoring Styles for issue 3989 and editors' action item 227
20070426	PY	Editorial changes to align with the OASIS WS-SecurityPolicy specification. For issue 4318. Editors' action 245.
20070427	FJH	Updated 5.5.1 Optional behavior in Compact authoring adding G7 and G8 for issue 3989 and editors' action item 250 as noted in message 69. Also replaced TBD in section 2 with descriptive text."
20070501	ASV	Reset Section **E. Changes in this Version of the Document** [p.32] .
20070507	PY	Updated 5.6 WSDL guidelines section, to follow the new format and added G15, G16, G17 and G18. Accounts for parts of resolution for issue 3989 corresponding to editors' action items 232, 253, and 256.
20070507	TIB	Updated 5.1 Assertions and their Target Use for issue 3989 and editors' action item 227.
20070508	MH	Updated Section 5 for adding guidelines G9, G10 on ignorable, and G5 , G6 (general) to address editors' action items 251. 256.
20070511	PY	Updated 5.6 WSDL guidelines section to add G19 identified in AI 256 (now G24). Accounts for parts of resolution for issue 3989 corresponding to editors' action item 256 - now complete.

20070513	ASV	Updated Section 5.4.1 to use the new format re issue issue 3989. Editors' action 230.
20070514	ASV	Updated Section 5.4.2 to use the new format re issue issue 3989. Editors' action 230. Collapsed Section 5.4.2 and 5.4.3.
20070514	ASV	Added G11 and G13 to Section 5.4.1 and 5.4.2 re issue issue 3989. Editors' action 252 and 255.
20070516	PY	Editorial change to section 5.7 to place best practices after the associated explanatory text and to fix grammar.
20070518	PY	Ensured Best Practices G1, G3 and G20 of original IBM/MS Contribution are reflected.
20070518	PY	Updated Appendix E, Changes in this Version of the Document (**E. Changes in this Version of the Document** [p.32]).
20070520	ASV	Added Best Practice **29. Specify Composition with Related Assertions** [p.26] (from the IBM and MS Contribution to **5.8 Interrelated domains** [p.26] . Added an ed note that Section **5.8 Interrelated domains** [p.26] needs to be re-structured.
20070520	ASV	Added Best Practice **12. Define message format only** [p.16] (from the IBM and MS Contribution to **5.3.3 Self Describing Messages** [p.16] .
20070520	ASV	Added an ed note that Section **5.5 Designating Ignorable Behavior** [p.20] looks incomplete.
20070520	ASV	Fixed typos.
20070520	ASV	Added an ed note in Section **5.1 Assertions and Their Target Use** [p.9] that there is an open issue against Best Practice G2.
20070524	DBO	Editorial changes to align with the W3C WS-Addressing Metadata specification. For issue 4375. Editors' action 284.
20070529	PY	Implemented Resolution for issue 4573. Apply "Best Practices" consistently.
20070529	PY	Implemented Resolution in Editors' action 290. Consistent use of Assertion Authors.
20070529	PY	Implemented Resolution in Editors' action 291. Consistent use of should in place of must in the best practice statements.
20070529	PY	Implemented Resolution in Editors' action 294.
20070530	PY	Implemented Resolution in Editors' action 303.
20070530	PY	Implemented Resolution in Editors' action 304.
20070530	PY	Implemented Resolution in Editors' action 305.
20070530	PY	Implemented Resolution in Editors' action 306.

20070530	PY	Implemented Resolution in Editors' action 307.
20070530	PY	Implemented Resolution in Editors' action 308.
20070601	TIB	Implemented Resolution in Editors' actions 310 and 311.
200706013	MH	Implemented Resolution in Editors' actions 292 and 293.
200706016	ASV	Implemented Editors' action 289.
20070616	ASV	Implemented the resolution for issue 4074. Editors' action 286.
200706018	ASV	Implemented Editors' action 295.
200706018	TIB	Implemented place holder for Editors' action 249 for locking the document.
20070713	FJH	Restructured and updated **5.8 Interrelated domains** [p.26] to use Architecture of WWW format and add example, according to Editors' action 309. Updated the WSDL 20 reference [*WSDL 2.0 Core Language [p.30]*] and WS-SecurityPolicy reference [*WS-SecurityPolicy [p.31]*] for issue 4831. Editors' action 326
20070717	FJH	Implemented the resolution for issue 4853. Editors' action 333.
20070717	FJH	Implemented the resolution for issue 4852. Editors' action 332.
20070717	DBO	Implemented partial resolution, section 5.5 updates, for issue 4662, Editors' action 332 #2.
20070718	ASV	Implemented the resolution for issue 3988. Editors' action: 338, drop Section 7 Scenario and a worked example
20070718	ASV	Implemented the resolution for issue 3978. Editors' action: 339, drop Section 6 Applying Best Practices for Policy Attachment
20070718	DBO	Implemented the resolution for issue 4661, 4662, 4861. Editors' action: 342, 346.
20070718	DBO	Implemented the resolution for issue 4664. Editors' action: 343.
20070718	DBO	Implemented the resolution for issue 4566. Editors' action: 249, 328.
20070718	FJH	Implemented the resolution for issue 4862. Editors' action: 348.
20070718	FJH	Implemented the resolution for issue 4654. Editors' action: 340. Add new section **5.7.1 General Guidelines** [p.23] .
20070718	FJH	Updated Web Services Reliable Messaging Policy reference [*Web Services Reliable Messaging Policy Assertion [p.31]*] and WS-Addressing Metadata reference [*WS-Addressing Metadata [p.30]*]. Editors' action 331.
20070719	FJH	Implemented the resolution for issue 4859. Editors' action: 335.
20070727	ASV	Implemented the resolution for issue 4660. Editors' action: 342.
20070727	ASV	Implemented the resolution for issue 4695. Editors' action: 347.

20070727	ASV	Updated Section **E. Changes in this Version of the Document** [p.32] .
20070806	FS	Updated references for draft publication.