

Basic Profile Version 2.0

Committee Specification Draft 02

27 April 2014

Specification URIs

This version:

<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csd02/BasicProfile-v2.0-csd02.pdf>
(Authoritative)
<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csd02/BasicProfile-v2.0-csd02.html>
<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csd02/BasicProfile-v2.0-csd02.doc>

Previous version:

<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csprd01/BasicProfile-v2.0-csprd01.doc>
(Authoritative)
<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csprd01/BasicProfile-v2.0-csprd01.html>
<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csprd01/BasicProfile-v2.0-csprd01.pdf>

Latest version:

<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/BasicProfile-v2.0.pdf> (Authoritative)
<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/BasicProfile-v2.0.html>
<http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/BasicProfile-v2.0.doc>

Technical Committee:

OASIS Web Services Basic Reliable and Secure Profiles (WS-BRSP) TC

Chair:

Jacques Durand (jdurand@us.fujitsu.com), Fujitsu Limited

Editors:

Tom Rutt (trutt@us.fujitsu.com), Fujitsu Limited
Micah Hainline (micah.hainline@asolutions.com), Asynchrony Solutions, Inc.
Ram Jeyaraman (Ram.Jeyaraman@microsoft.com), Microsoft
Jacques Durand (jdurand@us.fujitsu.com), Fujitsu Limited

Related work:

This specification is related to:

- WS-I Basic Profile 2.0 Final Material 2010-11-09. <http://ws-i.org/Profiles/BasicProfile-2.0-2010-11-09.html>.

Abstract:

This document defines the WS-I Basic Profile 2.0 consisting of a set of clarifications, refinements, interpretations and amplifications to a combination of non-proprietary Web services specifications in order to promote interoperability. It is an evolution of WS-I Basic Profile 1.1 and is based on SOAP 1.2. In particular it is adding support for WS-Addressing.

Status:

This document was last revised or approved by the OASIS Web Services Basic Reliable and Secure Profiles (WS-BRSP) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the

“Send A Comment” button on the Technical Committee’s web page at <https://www.oasis-open.org/committees/ws-brsp/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<https://www.oasis-open.org/committees/ws-brsp/ipr.php>).

Citation format:

When referencing this specification the following citation format should be used:

[BasicProfile-v2.0]

Basic Profile Version 2.0. Edited by Tom Rutt, Micah Hainline, Ram Jeyaraman, and Jacques Durand. 27 April 2014. OASIS Committee Specification Draft 02. <http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/csd02/BasicProfile-v2.0-csd02.html>. Latest version: <http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/BasicProfile-v2.0.html>.

Notices

Copyright © OASIS Open 2014. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Table of Contents

1	Introduction	8
1.1	Relationships to Other Profiles	8
1.1.1	Compatibility with Basic Profile 1.1	8
1.1.2	Relationship to Basic Profile 1.2.....	8
1.2	Guiding Principles	8
1.3	Test Assertions	9
1.4	Notational Conventions.....	10
1.5	Terminology	10
1.6	Profile Identification and Versioning	12
1.7	Normative References	12
1.8	Non-Normative References	15
2	Conformance	16
2.1	Requirements Semantics.....	16
2.2	Conformance Targets	17
2.3	Conformance Scope	17
2.4	Conformance Clauses	18
2.4.1	Core Conformance	18
2.4.2	HTTP Transport Conformance	18
2.5	Claiming Conformance	18
2.5.1	Claiming Conformance using the Conformance Claim Attachment Mechanisms	19
2.5.2	Claiming Conformance using WS-Policy and WS-PolicyAttachment	19
3	Messaging	21
3.1	Message Serialization.....	22
3.1.1	XML Envelope Serialization	22
3.1.2	Unicode BOMs	22
3.1.3	XML Declarations	23
3.1.4	Character Encodings	23
3.1.5	XOP Encoded Messages	23
3.2	SOAP Envelopes	24
3.2.1	SOAP Envelope Structure.....	24
3.2.2	SOAP Body Namespace Qualification	25
3.2.3	Disallowed Constructs	25
3.2.4	xsi:type Attributes	25
3.2.5	SOAP 1.2 attributes on SOAP 1.2 elements.....	25
3.3	SOAP Processing Model	25
3.3.1	SOAP Fault Processing.....	26
3.4	SOAP Faults	26
3.4.1	Identifying SOAP Faults	26
3.5	Use of SOAP in HTTP	26
3.5.1	HTTP Protocol Binding.....	26
3.5.2	Parameters on the Content-Type MIME Header.....	27
3.5.3	HTTP Success Status Codes.....	27
3.5.4	HTTP Redirect Status Codes	27

3.5.5 HTTP Cookies	27
3.5.6 Non-Addressable Consumers and Instances.....	28
3.6 Use of URIs in SOAP.....	28
3.6.1 Use of SOAP-defined URIs	29
3.7 WS-Addressing Support	29
3.7.1 Requiring WS-Addressing SOAP Headers	29
3.7.2 NotUnderstood block in MustUnderstand Fault on WS-Addressing SOAP Headers	29
3.7.3 Use of wsa:Action and WS-Addressing 1.0 - Metadata	29
3.7.4 Valid Values for action Parameter on the Content-Type MIME header When WS-Addressing is Used	30
3.7.5 SOAP Defined Faults Action URI	30
3.7.6 Understanding WS-Addressing SOAP Header Blocks	30
3.7.7 Ignored or Absent WS-Addressing Headers	30
3.7.8 Present and Understood WS-Addressing Headers.....	31
3.7.9 SOAP MustUnderstand or VersionMismatch fault Transmission.....	32
3.7.10 Faulting Behavior with Present and Understood WS-Addressing Headers	32
3.7.11 [message id] and One-Way Operations	33
3.7.12 Refusal to Honor WS-Addressing Headers.....	33
3.7.13 Use of Non-Anonymous Response EPRs.....	33
3.7.14 Optionality of the wsa:To header.....	33
3.7.15 Extending WSDL Endpoints with an EPR	34
3.7.16 Combining Synchronous and Asynchronous Operations	35
3.7.17 Conflicting Addressing Policies	39
4 Service Description.....	40
4.1 Required Description	40
4.2 Document Structure.....	41
4.2.1 WSDL Import location Attribute Structure	41
4.2.2 WSDL Import location Attribute Semantics	41
4.2.3 XML Version Requirements	41
4.2.4 XML Namespace Declarations.....	41
4.2.5 WSDL and the Unicode BOM.....	41
4.2.6 Acceptable WSDL Character Encodings	42
4.2.7 Namespace Coercion	42
4.2.8 WSDL Extensions	42
4.3 Types	43
4.3.1 QName References.....	43
4.3.2 Schema targetNamespace Structure	43
4.3.3 soapenc:Array	43
4.3.4 WSDL and Schema Definition Target Namespaces	45
4.3.5 Multiple GED Definitions with the same QName.....	45
4.3.6 Multiple Type Definitions with the same QName	45
4.4 Messages.....	45
4.4.1 TBindings and Parts	46
4.4.2 Bindings and Faults.....	47
4.4.3 Unbound portType Element Contents.....	47
4.5 Port Types.....	48

4.5.1	Ordering of part Elements	48
4.5.2	Allowed Operations	48
4.5.3	Distinctive Operations	48
4.5.4	parameterOrder Attribute Construction	48
4.5.5	Exclusivity of type and element Attributes.....	49
4.6	Bindings	49
4.6.1	Use of SOAP Binding	49
4.7	SOAP Binding.....	49
4.7.1	HTTP Transport.....	49
4.7.2	Consistency of style Attribute	49
4.7.3	Encodings and the use Attribute	50
4.7.4	Multiple Bindings for portType Elements.....	50
4.7.5	Operation Signatures	50
4.7.6	Multiple Ports on an Endpoint	50
4.7.7	Child Element for Document-Literal Bindings	51
4.7.8	One-Way Operations.....	51
4.7.9	Namespaces for wsoap12 Elements.....	51
4.7.10	Consistency of portType and binding Elements.....	52
4.7.11	Enumeration of Faults	52
4.7.12	Consistency of Envelopes with Descriptions.....	52
4.7.13	Response Wrappers.....	53
4.7.14	Part Accessors	53
4.7.15	Namespaces for Children of Part Accessors	53
4.7.16	Required Headers	55
4.7.17	Allowing Undescribed Headers	55
4.7.18	Ordering Headers.....	55
4.7.19	Describing action Parameter on the Content-Type MIME Header.....	56
4.7.20	SOAPAction HTTP Header	56
4.7.21	SOAP Binding Extensions.....	56
4.8	Use of @soapActionRequired in WSDL 1.1 SOAP 1.2 Binding.....	57
4.9	Use of XML Schema.....	58
4.10	4.10 WS-Addressing 1.0 - Metadata	58
5	WSDL Corrections.....	59
5.1	Document Structure.....	59
5.1.1	WSDL Schema Definitions	59
5.1.2	WSDL and Schema Import.....	59
5.1.3	Placement of WSDL import Elements	61
5.1.4	WSDL documentation Element	63
5.2	Message	63
5.2.1	Declaration of part Elements	63
5.3	SOAP Binding.....	64
5.3.1	Specifying the transport Attribute	64
5.3.2	SOAP 1.2 Binding Extension.....	64
5.3.3	Type and Name of SOAP Binding Elements.....	64
5.3.4	name Attribute on Faults	65

5.3.5 Omission of the use Attribute	65
5.3.6 Default for use Attribute.....	65
6 Service Publication and Discovery	66
6.1 bindingTemplates	66
6.2 tModels	67
7 Security.....	69
7.1 Use of HTTPS.....	70
Appendix A. Extensibility Points	71
Appendix B. Schemas	73
Appendix C. Testing	74
C.1 Testability of Requirements.....	74
C.2 Structure of Test Assertions.....	74
C.3 Test Log Conventions	77
Appendix D. Test Assertions	78
Appendix E. Acknowledgments.....	131
Appendix F. Revision History	132

1 Introduction

This document defines the WS-I Basic Profile 2.0 (hereafter, "Profile"), consisting of a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability.

Section 2 introduces the Profile, and explains its relationships to other profiles.

Section 3, "Profile Conformance," explains what it means to be conformant to the Profile.

Each subsequent section addresses a component of the Profile, and consists of two parts; an overview detailing the component specifications and their extensibility points, followed by subsections that address individual parts of the component specifications. Note that there is no relationship between the section numbers in this document and those in the referenced specifications. Relationships to Other Profiles

1.1 Relationships to Other Profiles

This Profile is derived from the Basic Profile 1.1 [BP1.1] by incorporating any errata to date and including those requirements related to the serialization of envelopes and their representation in messages from the Simple SOAP Binding Profile 1.0 [SSBP1.0].

The Attachments Profile 1.0 [AP1.0] adds support for SOAP with Attachments, and is intended to be used in combination with this Profile.

1.1.1 Compatibility with Basic Profile 1.1

This Profile (BP 2.0) is the first version of the WS-I Basic Profile that changes the version of SOAP in the profile scope from SOAP 1.1 to the W3C SOAP 1.2 Recommendation [SOAP12-1], [SOAP12-2]. As such, BP 1.1 conformant messages are inherently incompatible with those conformant with BP 2.0, while receivers and instances processing these messages may or may not support these two versions of the Basic Profile.

1.1.2 Relationship to Basic Profile 1.2

Similarly to this Profile, Basic Profile 1.2 [BP1.2] (BP 1.2) is derived from Basic Profile 1.1 [BP1.1]. Unlike this Profile, the version of SOAP in scope for BP 1.2 is, like BP 1.1, SOAP 1.1 [SOAP1.1]. To the extent possible, this Profile and BP 1.2 attempt to maintain a common set of requirement numbers, and common requirement and expository text. There are cases where the differences between SOAP 1.1 and SOAP 1.2 necessitate unique requirements and/or differing requirement and expository text. Therefore, some requirements and test assertions may present issues of forward or backward compatibility.

1.2 Guiding Principles

The Profile was developed according to a set of principles that, together, form the philosophy of the Profile, as it relates to bringing about interoperability. This section documents these guidelines.

No guarantee of interoperability

It is impossible to completely guarantee the interoperability of a particular service. However, the Profile does address the most common problems that implementation experience has revealed to date.

Application semantics

Although communication of application semantics can be facilitated by the technologies that comprise the Profile, assuring the common understanding of those semantics is not addressed by it.

Testability

When possible, the Profile makes statements that are testable. However, requirements do not need to be testable to be included in the Profile. Preferably, testing is achieved in a non-intrusive manner (e.g., by examining artifacts "on the wire").

Strength of requirements

The Profile makes strong requirements (e.g., MUST, MUST NOT) wherever feasible; if there are legitimate cases where such a requirement cannot be met, conditional requirements (e.g., SHOULD, SHOULD NOT) are used. Optional and conditional requirements introduce ambiguity and mismatches between implementations.

Restriction vs. relaxation

When amplifying the requirements of referenced specifications, the Profile may restrict them, but does not relax them (e.g., change a MUST to a MAY).

Multiple mechanisms

If a referenced specification allows multiple mechanisms to be used interchangeably, the Profile selects those that are well-understood, widely implemented and useful. Extraneous or underspecified mechanisms and extensions introduce complexity and therefore reduce interoperability.

Future compatibility

When possible, the Profile aligns its requirements with in-progress revisions to the specifications it references. This aids implementers by enabling a graceful transition, and assures that WS-I does not 'fork' from these efforts. When the Profile cannot address an issue in a specification it references, this information is communicated to the appropriate body to assure its consideration.

Compatibility with deployed services

Backwards compatibility with deployed Web services is not a goal for the Profile, but due consideration is given to it; the Profile does not introduce a change to the requirements of a referenced specification unless doing so addresses specific interoperability issues.

Focus on interoperability

Although there are potentially a number of inconsistencies and design flaws in the referenced specifications, the Profile only addresses those that affect interoperability.

Conformance targets

Where possible, the Profile places requirements on artifacts (e.g., WSDL descriptions, SOAP messages) rather than the producing or consuming software's behaviors or roles. Artifacts are concrete, making them easier to verify and therefore making conformance easier to understand and less error-prone.

Lower-layer interoperability

The Profile speaks to interoperability at the application layer; it assumes that interoperability of lower-layer protocols (e.g., TCP, IP, Ethernet) is adequate and well-understood. Similarly, statements about application-layer substrate protocols (e.g., SSL/TLS, HTTP) are only made when there is an issue affecting Web services specifically; WS-I does not attempt to assure the interoperability of these protocols as a whole. This assures that WS-I's expertise in and focus on Web services standards is used effectively.

1.3 Test Assertions

This profile document is complemented by Appendix D Test Assertions (TA) that contains scripted (XPath 2.0) test assertions for assessing conformance of an endpoint to the BP2.0 profile.

Test assertions are not guaranteed to exhaustively cover every case where a profile requirement applies. In several instances, more than one test assertion is needed to address the various situations where a profile requirement applies

Each profile requirement is tagged with:

- The level of conformance this requirement belongs to (either CORE, or HTTP-TRANSPORT). See the Conformance section.
- A testability assessment (TESTABLE, TESTABLE_SCENARIO_DEPENDENT, NOT_TESTED, NOT_TESTABLE)
- Optionally, one or more test assertion identifiers (e.g. BP1905)

The structure of test assertions and the meaning of the testability assessment are described in Appendix C. Testing

1.4 Notational Conventions

The key words MUST , MUST NOT , REQUIRED , SHALL , SHALL NOT , SHOULD , SHOULD NOT , RECOMMENDED , MAY , and OPTIONAL in this document are to be interpreted as described in [RFC2119].

Normative statements in the Profile (i.e., those impacting conformance, as outlined in Section 2) are called "Requirements" and presented in the following manner:

Rnnnn *Statement text here.*

where "nnnn" is replaced by a number that is unique among the Requirements in the Profile, thereby forming a unique Requirement identifier.

Extensibility points in underlying specifications (see Section 2.3) are presented in a similar manner:

Ennnn *Extensibility Point Name - Description*

where "nnnn" is replaced by a number that is unique among the extensibility points in the Profile. As with requirement statements, extensibility statements can be considered namespace-qualified.

This specification uses a number of namespace prefixes throughout; their associated URIs are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

- **soap12** - " http://www.w3.org/2003/05/soap-envelope "
- **xsi** - " http://www.w3.org/2001/XMLSchema-instance "
- **xsd** - " http://www.w3.org/2001/XMLSchema "
- **soapenc** - " http://www.w3.org/2003/05/soap-encoding "
- **wsdl** - " http://schemas.xmlsoap.org/wsdl/ "
- **wsoap12** - " http://schemas.xmlsoap.org/wsdl/soap12/ "
- **uddi** - " urn:uddi-org:api_v2 "
- **wsa** - " http://www.w3.org/2005/08/addressing "
- **xop** - " http://www.w3.org/2004/08/xop/include "

1.5 Terminology

The following list of terms have specific definitions that are authoritative for this profile:

non-addressable

A CONSUMER or INSTANCE is deemed "non-addressable" when, for whatever reason, it is either unwilling or unable to provide a network endpoint that is capable of accepting connections. This means that the CONSUMER or INSTANCE cannot service incoming HTTP connections and can only transmit HTTP Request messages and receive HTTP Response messages.

rpc-literal binding

An "rpc-literal binding" is a `wsdl:binding` element whose child `wsdl:operation` elements are all rpc-literal operations.

An "rpc-literal operation" is a `wsdl:operation` child element of `wsdl:binding` whose `wsoap12:body` descendant elements specify the `use` attribute with the value "literal", and either:

1. The `style` attribute with the value "rpc" is specified on the child `wsoap12:operation` element; or
2. The `style` attribute is not present on the child `wsoap12:operation` element, and the `wsoap12:binding` element in the enclosing `wsdl:binding` specifies the `style` attribute with the value "rpc".

document-literal binding

A "document-literal binding" is a `wsdl:binding` element whose child `wsdl:operation` elements are all document-literal operations.

A "document-literal operation" is a `wsdl:operation` child element of `wsdl:binding` whose `wsoap12:body` descendent elements specifies the `use` attribute with the value "literal" and, either:

1. The `style` attribute with the value "document" is specified on the child `wsoap12:operation` element; or
2. The `style` attribute is not present on the child `wsoap12:operation` element, and the `wsoap12:binding` element in the enclosing `wsdl:binding` specifies the `style` attribute with the value "document"; or
3. The `style` attribute is not present on both the child `wsoap12:operation` element and the `wsoap12:binding` element in the enclosing `wsdl:binding`.

operation signature

The Profile defines the "operation signature" to be the fully qualified name of the child element of SOAP body of the SOAP input message described by an operation in a WSDL binding and the URI value of the `wsa:Action` SOAP header block, if present.

In the case of rpc-literal binding, the operation name is used as a wrapper for the part accessors. In the document-literal case, since a wrapper with the operation name is not present, the message signatures must be correctly designed so that they meet this requirement.

1.6 Profile Identification and Versioning

This document is identified by a name (in this case, Basic Profile) and a version number (here, 2.0). Together, they identify a *profile* (here, Basic Profile 2.0).

Version numbers are composed of a major and minor portion, in the form "major.minor". They can be used to determine the precedence of a profile instance; a higher version number (considering both the major and minor components) indicates that an instance is more recent, and therefore supersedes earlier instances.

Instances of profiles with the same name (e.g., "Example Profile 1.1" and "Example Profile 5.0") address interoperability problems in the same general scope (although some developments may require the exact scope of a profile to change between instances).

One can also use this information to determine whether two instances of a profile are backwards-compatible; that is, whether one can assume that conformance to an earlier profile instance implies conformance to a later one. Profile instances with the same name and major version number (e.g., "Example Profile 1.0" and "Example Profile 1.1") may be considered compatible. Note that this does not imply anything about compatibility in the other direction; that is, one cannot assume that conformance with a later profile instance implies conformance to an earlier one.

1.7 Normative References

The following specifications' requirements are incorporated into the Profile by reference, except where superseded by the Profile:

- [AP1.0]** "Attachments Profile Version 1.0 (AP1.0)", WS-I Final Material, 20 April 2006, (ISO/IEC 29362:2008 Information technology -- Web Services Interoperability -- WS-I Attachments Profile Version 1.0), <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>
- [BP1.1]** "Basic Profile Version 1.1 (BP 1.1)", WS-I Final Material, 10 April 2006, (ISO/IEC 29361:2008 Information technology -- Web Services Interoperability -- WS-I Basic Profile Version 1.10), <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>
- [BP1.2]** *Basic Profile 1.2*, OASIS Committee Specification Draft, May 2013. (TBD)
- [claimAttachment]** M. Nottingham et al., WS-I Conformance Claim Attachment Mechanisms Version 1.0, November 2004. <http://www.ws-i.org/Profiles/ConformanceClaims-1.0-2004-11-15.html>
- [RFC2119]** Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, BCP 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- [RFC2246]** Dierks, T. and C. Allen, The TLS Protocol Version 1.0, RFC 2246, January 1999, <http://www.ietf.org/rfc/rfc2246.txt>

- [RFC2459]** Housley,R., Ford, W., Polk, W., and D. Solo, Internet X.509 Public Key Infrastructure Certificate and CRL Profile , RFC 2459 , January 1999, <http://www.ietf.org/rfc/rfc2459.txt>
- [RFC2616]** Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, Hypertext Transfer Protocol -- HTTP/1.1 , RFC 2616, June 1999. <http://www.ietf.org/rfc/rfc2616>
- [RFC2818]** E. Rescorla , "HTTP over TLS", RFC 2818, May 2000, <http://www.ietf.org/rfc/rfc2818.txt>
- [RFC2965]** Kristol, D. and L. Montulli, HTTP State Management Mechanism , RFC 2965, October 2000. <http://www.ietf.org/rfc/rfc2965>
- [RFC3023]** Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types , RFC 3023, January 2001,., <http://www.ietf.org/rfc/rfc3023>
- [RFC3986]** Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax , RFC 3896, January 2005, <http://www.apps.ietf.org/rfc/rfc3986.html>
- [SOAP12-mtom]** "SOAP Message Transmission Optimization Mechanism , M. Gudgin, N. Mendelsohn, M. Nottingham, H. Ruellan, Editors, W3C Recommendation, 25 January 2005,., <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>
- [SOAP12-1]** "SOAP Version 1.2 Part 1: Messaging Framework , M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen,, A. Karmarkar, Y. Lafon, Editors, W3C Recommendation, 2007,<http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>
- [SOAP12-2]** "SOAP Version 1.2 Part 2: Adjuncts , M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen,, A. Karmarkar, Y. Lafon, Editors, W3C Recommendation, 2007, <http://www.w3.org/TR/2007/REC-soap12-part2-20070427/>
- [SSBP1.0]** "Simple SOAP Binding Profile Version 1.0 Final Material 24 August 2004 (ISO/IEC 29363:2008 Information technology -- Web Services Interoperability -- WS-I Simple SOAP Binding Profile Version 1.0), <http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html>
- [SSLV3]** Freirer, A,, P. Karlton, and P. Kocher, The SSL Protocol Version 3.0 , Internet Draft , November 18, 1996, <http://tools.ietf.org/search/draft-ietf-tls-ssl-version3-00>
- [UDDI2schema]** UDDI Version 2 XML Schema", 2002, http://uddi.org/schema/uddi_v2.xsd
- [UDDI2.04API]** UDDI Version 2.04 API Specification", UDDI Committee Specification,19 July 2002, <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>
- [UDDI2.03Data]** UDDI Version 2.03 Data Structure Reference', UDDI Committee Specification, 19 July 2002, <http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm>

- [Unicode]** The Unicode Consortium. *The Unicode Standard*, Version 6.3.0, (Mountain View, CA: The Unicode Consortium, 2013. ISBN 978-1-936213-08-5), <http://www.unicode.org/versions/Unicode6.3.0/>
- [WSAddrCore]** "WS-Addressing 1.0 - Core", M. Gudgin, M. Hadley, T. Rogers, Editors, W3C Recommendation, 9 May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- [WSAddrSoap]** "WS-Addressing 1.0 SOAP Binding", M. Gudgin, M. Hadley, T. Rogers, Editors, W3C Recommendation, 9 May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/> (except for sections 4, 5.1.1, 5.2.1 and 6.2)
- [WSAddrMeta]** "WS-Addressing 1.0 Metadata", M. Gudgin, M. Hadley, T. Rogers, Umit Yalcinap, Editors, W3C Recommendation, 4 September 2007, <http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/> (except for sections 4.1.1, 4.4.2, 4.4.3 and 5.2)
- [WSDL1.1]** "Web Services Description Language (WSDL) 1.1", W3C Note, 15 March 2001. <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- [WSDL11SOAP12]** "WSDL 1.1 Binding Extensions for SOAP 1.2", W3C Member Submission, 05 April 2006, <http://www.w3.org/Submission/wsdl11soap12/>
- [WSPolicy1.5]** Web Services Policy 1.5 Framework, A. Vedomuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, Umit Yalinalp, Editors. W3C Recommendation 04 September 2007, <http://www.w3.org/TR/2007/REC-ws-policy-20070904>
- [WSPolicyAtt1.5]** Web Services Policy 1.5 Attachment, A. Vedomuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, Umit Yalinalp, Editors. W3C Recommendation 04 September 2007, <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>
- [XML1.0]** "Extensible Markup Language (XML) 1.0 (Fourth Edition)", T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, W3C Recommendation, 29 September 2006, <http://www.w3.org/TR/2006/REC-xml-20060816/>
- [xmlNames]** "Namespaces in XML 1.0 (Second Edition)", T. Bray, D. Hollander, A. Layman, R. Tobin, Editors, W3C Recommendation, 16 August 2006, <http://www.w3.org/TR/2006/REC-xml-names-20060816/>
- [xmlSchema-1]** XML Schema Part 1: Structures (Second Edition)", H.S Thompson, D. Beech, M. Maloney, N. Mendelsohn, Editors, W3C Recommendation, 28 October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
- [xmlSchema-2]** "XML Schema Part 2: Datatypes (Second Edition)", P. V. Biron, A. Malhotra, Editors, W3C Recommendation, 28 October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- [xop]** "XML-binary Optimized Packaging", M. Gudgin, N. Mendelsohn, M. Nottingham, H. Ruellan, Editors, W3C Recommendation, 25 January 2005, <http://www.w3.org/TR/2005/REC-xop10-20050125/>

1.8 Non-Normative References

- [RFC2145]** Mogul J. C, Fielding R., Gettys J., Frystyk H., Use and Interpretation of HTTP Version Numbers , , RFC 2145, May 1997, <http://www.ietf.org/rfc/rfc2145.txt>
- [SOAP1.1]** "Simple Object Access Protocol (SOAP) 1.1", W3C Note, 08 May 2000, <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- [WSMakeConn]** "Web Services Make Connection (WS-MakeConnection) Version 1.1 , OASIS Standard, 2 February 2009, <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-os.html>

2 Conformance

Conformance to the Profile is defined by adherence to the set of identified *Requirements* defined for a specific *target*, within the *scope* of the Profile. This section explains these terms and describes how conformance is defined and used.

2.1 Requirements Semantics

The Profile is defined using a set of Requirements (see section 1.4 for the general format of a Requirement, including its identification). Each Requirement is an atomic normative statement targeting a particular artifact subject to conformance assessment. In other words, Requirements state the criteria for conformance to the Profile. They typically refer to an existing specification and embody refinements, amplifications, interpretations and clarifications to it in order to improve interoperability. All Requirements in the Profile are normative, and those in the specifications it references that are in-scope (see "Conformance Scope") should likewise be considered normative. When Requirements in the Profile and its referenced specifications contradict each other, the Profile's Requirements take precedence for purposes of Profile conformance.

Requirement levels, using [RFC2119] language (e.g., MUST, MAY, SHOULD) indicate the nature of the requirement and its impact on conformance. Each Requirement is individually identified (e.g., R9999) for convenience.

For example;

R9999 Any **WIDGET** *SHOULD* be round in shape.

This Requirement is identified by "R9999", applies to the target WIDGET (see below), and places a conditional requirement upon widgets.

Each Requirement statement contains exactly one Requirement level keyword (e.g., "MUST") and one conformance target keyword (e.g., "MESSAGE"). The conformance target keyword appears in bold text (e.g. "**MESSAGE**"). Other conformance targets appearing in non-bold text are being used strictly for their definition and NOT as a conformance target. Additional text may be included to illuminate a Requirement or group of Requirements (e.g., rationale and examples); however, prose surrounding Requirement statements must not be considered in determining conformance.

A Requirement is said "mandatory" if its keyword is of mandatory nature (e.g. MUST, MUST NOT, REQUIRED).

Definitions of terms in the Profile are considered authoritative for the purposes of determining conformance.

No other content is normative in this document outside the numbered Requirements and the conformance claim mechanisms (section 2.5).. In particular:

- Examples material is not normative and only intended as illustrative.
- Appendix material is not normative.
- Test Assertions associated with this profile specification are not normative.
- Explanatory text introducing Requirements is not normative.
- Notes are not normative.
- Schemas are not normative.

2.2 Conformance Targets

Conformance targets identify what artifacts (e.g., SOAP message, WSDL description, UDDI registry data) or parties (e.g., SOAP processor, end user) requirements apply to.

This allows for the definition of conformance in different contexts, to assure unambiguous interpretation of the applicability of requirements, and to allow conformance testing of artifacts (e.g., SOAP messages and WSDL descriptions) and the behavior of various parties to a Web service (e.g., clients and service instances).

Requirements' conformance targets are physical artifacts wherever possible, to simplify testing and avoid ambiguity.

The following conformance targets are used in the Profile:

- **MESSAGE** - protocol elements that transport the ENVELOPE (e.g., SOAP/HTTP messages)
- **ENVELOPE** - the serialization of the soap12:Envelope element and its content
- **DESCRIPTION** - descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (e.g., WSDL descriptions) (from [BP1.0])
- **INSTANCE** - software that implements a wsdl:port or a uddi:bindingTemplate (from [BP1.0])
- **CONSUMER** - software that invokes an INSTANCE (from [BP1.0])
- **SENDER** - software that generates a message according to the protocol(s) associated with it (from [BP1.0])
- **RECEIVER** - software that consumes a message according to the protocol(s) associated with it (e.g., SOAP processors) (from [BP1.0])
- **REGDATA** - registry elements that are involved in the registration and discovery of Web services (e.g. UDDI tModels) (from [BP1.0])
- **SIMPLE_SOAP_MESSAGE** - A MESSAGE that has as an entity-body that has a 'Content-Type' HTTP header field with a field-value of 'application/soap+xml'HTTP-TRANSPORT
- **XOP_ENCODED_MESSAGE** - A MESSAGE that has an entity-body that has a 'Content-Type' HTTP header field with a field-value of 'multipart/related' with a type parameter of 'application/xop+xml'HTTP-TRANSPORT

2.3 Conformance Scope

The Profile's functional scope includes the set of specifications referenced by it. However the conformance requirements that are proper to each one of these underlying specifications (e.g. defining conformance to SOAP 1.1) are not part of the Profile. Only the requirements and restrictions put on the usage of these specifications are part of the Profile. In other words, claiming conformance to this Profile does not imply conformance to SOAP1.1, but it implies a particular way to use SOAP1.1.

The Profile's scope is further limited by extensibility points. Referenced specifications often provide extension mechanisms and unspecified or open-ended configuration parameters; when identified in the Profile as an extensibility point, such a mechanism or parameter is outside the scope of the Profile, and its use or non-use is not relevant to conformance. These extensibility points are however listed here to point at possible risks of interoperability loss that are not addressed by the Profile. Because the use of extensibility points may impair interoperability, their use should be negotiated or documented in some fashion by the parties to a Web service; for example, this could take the form of an out-of-band agreement.

However the Profile may still express constraints on the use of an extensibility point. Also, specific uses of extensibility points may be further restricted by other profiles, to improve interoperability when used in conjunction with the Profile.

The Profile's scope is defined by the referenced specifications in clause 1.7, as limited by the extensibility points in Appendix A.

2.4 Conformance Clauses

This Profile concerns several conformance targets. Conformance targets are identified in requirements as described in Section . Conformance claims may apply to any above conformance target.

For a given conformance target, there are in addition two major ways to conform to this profile, identified by tags associated with each profile Requirement. These tags are CORE and HTTP-TRANSPORT corresponding respectively to two conformance levels: "Core" and "HTTP-Transport":

- "CORE" (transport- independent) conformance level. When the endpoint advertising conformance to this Profile is using a transport other than HTTP, then only the Requirements tagged with "CORE" apply to the conformance targets under responsibility of this endpoint.
- "HTTP-TRANSPORT" (HTTP transport-specific) conformance level. When the endpoint advertising conformance to this Profile is using HTTP, then all of the Requirements of the Profile tagged either with CORE or with HTTP-TRANSPORT apply to the conformance targets under responsibility of this endpoint.

These define two levels of conformance, as Core conformance is included in HTTP-transport conformance. In other words, conformance at HTTP-transport level implies conformance at Core level.

2.4.1 Core Conformance

A A conformance target (as defined in 2.2) is said to be conforming to this profile at the core conformance level if this target fulfills all the mandatory Requirements (see precise meaning in 1.4) that are tagged CORE and that identify this target type. Conformance at this level is independent from a message transport layer, e.g. could use a different protocol such as SMTP instead of HTTP.

2.4.2 HTTP Transport Conformance

A conformance target (as defined in 2.2) is said to be conforming to this profile at the HTTP transport conformance level if this target fulfills all the mandatory Requirements (see precise meaning in 1.4) that are tagged either as CORE or as HTTP-TRANSPORT and that identify this target type.

In other words, conformance at this level implies conformance at Core level.

2.5 Claiming Conformance

This specification defines two mechanisms to claim conformance to the Profile the use of which needs be agreed upon by users: 1) the Conformance Claim Attachment Mechanisms [claimAttachment] (see Section), or 2) the Web Services Policy - Framework [WSPolicy1.5] and Web Services Policy - Attachment [WSPolicyAtt1.5] (see Section).

In a similar way as for extensibility points, the choice of a conformance claim mechanism is not part of the Profile definition: should the interacting parties decide to use one of them to advertise support for the

Profile, a prior agreement must be established that is beyond the scope of this Profile. Whether these conformance claim mechanisms are supported or not does not affect conformance to the Profile.

In consequence, although the use of these conformance claim mechanisms is optional, they are described in a normative way to help partners define such agreements unambiguously.

2.5.1 Claiming Conformance using the Conformance Claim Attachment Mechanisms

Claims of conformance to this Profile MAY be made using the following Conformance Claim Attachment Mechanisms [claimAttachment] , Such claims concern specific groups of conformance targets as follows:

- **WSDL 1.1 Claim Attachment Mechanism for Web Services Instances** - MESSAGE, DESCRIPTION, INSTANCE, RECEIVER
- **WSDL 1.1 Claim Attachment Mechanism for Web Description Constructs** - DESCRIPTION
- **UDDI Claim Attachment Mechanism for Web Services Instances** - MESSAGE, DESCRIPTION, INSTANCE, RECEIVER
- **UDDI Claim Attachment Mechanism for Web Services Registrations** - REGDATA

The Basic Profile 2.0 conformance claim URI is:

<http://ws-i.org/profiles/basic-profile/2.0/Conformant>

When a web service instance is using HTTP, then all of the requirements of the Profile apply as specified in Section 2.4.2.

2.5.2 Claiming Conformance using WS-Policy and WS-PolicyAttachment

Mechanisms described in Web Services Policy - Framework [WSPolicy1.5] and Web Services Policy - Attachment [WSPolicyAtt1.5] specifications MAY be used to advertise conformance to this Profile. The Profile defines the following policy assertion for this purpose:

```
<bp20:Conformant xmlns:bp20="http://ws-i.org/profiles/basic-profile/2.0/" />
```

A copy of the XML Schema is provided in Appendix B , for convenience.

The presence of this assertion indicates that the policy subject supports the requirements of this Profile in a manner that conforms to Basic Profile 2.0 (See Section). This assertion also requires that CONSUMERS MUST use the effected protocols in a way that conforms to Basic Profile 2.0. The absence of this assertion says nothing about Basic Profile 2.0 conformance; it simply indicates the lack of an affirmative declaration of and requirement for Basic Profile 2.0 conformance.

The `bp20:Conformant` policy assertion applies to the endpoint policy subject.

For WSDL 1.1, this assertion can be attached to a `wsdl11:port` or `wsdl11:binding` . A policy expression containing the `bp20:Conformant` policy assertion MUST NOT be attached to a `wsdl:portType` .

For example,

CORRECT:

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

The example above shows a policy expression that requires Basic Profile 2.0.

For example,

CORRECT:

```
<wsp:Policy xmlns:bp20="http://ws-i.org/profiles/basic-profile/2.0/"
            xmlns:wsp="http://www.w3.org/ns/ws-policy"
            xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata">
  <wsam:Addressing>
    <wsp:Policy/>
  </wsam:Addressing>
  <bp20:Conformant/>
</wsp:Policy>
```

The example above shows a policy expression that requires WS-Addressing and Basic Profile 2.0.

3 Messaging

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [SOAP12-1]
Extensibility points:
 - E0001 - Header blocks - Header blocks are an extensibility mechanism in SOAP. CORETESTABLE BP1901
 - E0002 - Processing order - The order of processing of a SOAP envelope's components (e.g., headers) is unspecified, and therefore may need to be negotiated out-of-band. CORENOT_TESTABLE
 - E0003 - Use of intermediaries - SOAP Intermediaries is an underspecified mechanism in SOAP 1.2, and their use may require out-of-band negotiation. Their use may also necessitate careful consideration of where Profile conformance is measured. CORENOT_TESTABLE
 - E0004 - SOAP role values - Values of the SOAP role attribute, other than the special uri 'http://www.w3.org/2003/05/soap-envelope/role/next', represent a private agreement between parties of the web service. CORETESTABLE BP1904
 - E0005 - Fault details - Faults may have Detail elements. The contents of these elements are not described in SOAP 1.2. CORETESTABLE BP1905
 - E0024 - Namespace Attributes - Namespace attributes on soap12:Envelope and soap12:Header elements CORETESTABLE
 - E0025 - Attributes on soap12:Body elements - Neither namespaced nor local attributes are constrained by SOAP 1.2. CORETESTABLE
- SOAP Version 1.2 Part 2: Adjuncts (Second Edition) [SOAP12-2]
Extensibility points:
 - E0026 - SOAP envelope in HTTP Response message to WSDL one-way operation - The SOAP 1.2 Request Response Binding specification does not specify the purpose or processing of such envelopes. HTTP-TRANSPORTTESTABLE
- RFC2616: Hypertext Transfer Protocol -- HTTP/1.1 [RFC2616]
Extensibility points:
 - E0007 - HTTP Authentication - HTTP authentication allows for extension schemes, arbitrary digest hash algorithms and parameters. HTTP-TRANSPORTTESTABLE
 - E0008 - Unspecified Header Fields - HTTP allows arbitrary headers to occur in messages. HTTP-TRANSPORTTESTABLE
 - E0010 - Content-Encoding - The set of content-codings allowed by HTTP is open-ended and any besides 'gzip', 'compress', or 'deflate' are an extensibility point. HTTP-TRANSPORTTESTABLE
 - E0011 - Transfer-Encoding - The set of transfer-encodings allowed by HTTP is open-ended. HTTP-TRANSPORTTESTABLE
 - E0029 - Use of messages other than SOAP 1.2 or XOP messages - Use of Messages other than a SIMPLE_SOAP_MESSAGE or a XOP_ENCODED_MESSAGE is an extensibility point CORETESTABLE
- RFC2965: HTTP State Management Mechanism [RFC2965]
- WS-Addressing 1.0 - Core [WSAddrCore]
- WS-Addressing 1.0 - SOAP Binding [WSAddrSoap] (except for sections 4, 5.1.1, 5.2.1 and 6.2)
Extensibility points:
 - E0027 - Use of SOAP role attribute and WS-Addressing - WS-Addressing allows multiple instances of headers such as wsa:To, wsa:ReplyTo, and wsa:FaultTo, so long as they are targeted to different SOAP roles. CORETESTABLE

- **E0028** - Endpoint references are extensible - When extension attributes or elements appear as part of an endpoint reference, the processing model for such extensions is defined by the specification for those extensions. **CORENOT_TESTABLE**
- WS-Addressing 1.0 - Metadata [WSAddrMeta] (except for sections 4.1.1, 4.4.2, 4.4.3 and 5.2)
- SOAP Message Transmission Optimization Mechanism [SOAP12-mtom]
- XML-Binary Optimized Packaging [xop]
- XML Media Types [RFC3023]

These extensibility points are listed, along with any extensibility points from other sections of this Profile, in Appendix A.

3.1 Message Serialization

This Profile is intended to compose with mechanisms to describe whether messages are encoded as SIMPLE_SOAP_MESSAGES or XOP_ENCODED_MESSAGES. As such it does not mandate that both of these encodings be supported for any given operation. Indeed, neither of these encodings need be supported if an alternate encoding such as that described in the Attachments Profile 1.0 [AP1.0] is used.

SOAP 1.2 defines an XML structure for serializing messages, the envelope. This Profile places constraints on the use and serialization of the soap12:Envelope element and its content:

This Profile allows for the use of protocol bindings other than HTTP. Section identifies the use of Simple SOAP and XOP encoded messages using HTTP. This subclause identifies how encoding is handled for HTTP only. [RFC2616] and [RFC3023] provide guidance for HTTP, supplemented by requirements throughout this profile. If another transport protocol is used, the responsibility for defining how to handle transport-specific features (e.g. content encoding) falls to the specification of the binding of SOAP to that transport protocol.

This section of the Profile incorporates the following specifications by reference:

- Extensible Markup Language (XML) 1.0 (Fourth Edition) [XML1.0]
- Attachments Profile Version 1.0 [AP1.0]

3.1.1 XML Envelope Serialization

R9701 An **ENVELOPE** *MUST* be serialized as XML 1.0. **CORETESTABLE**
BP1019

3.1.2 Unicode BOMs

XML 1.0 allows UTF-8 encoding to include a BOM; therefore, receivers of envelopes must be prepared to accept them. The BOM is mandatory for XML encoded as UTF-16.

R4006 A **RECEIVER** *MUST* support the presence of a UTF-8 Unicode Byte Order Mark (BOM) in the SOAP envelope when the envelope is correctly encoded using UTF-8 and the "charset" parameter of the HTTP *Content-Type* header has a value of "utf-8" (see [RFC3023](#)). **CORETESTABLE_SCENARIO_DEPENDENT** BP1306

R4007 A **RECEIVER** *MUST NOT* fault due to the presence of a UTF-16 Unicode Byte Order Mark (BOM) in the SOAP envelope when the envelope is correctly encoded using UTF-16 and the "charset" parameter of the HTTP *Content-Type* header has a

value of "utf-16" (see [RFC3023](#)).
CORETESTABLE **SCENARIO_DEPENDENT** **BP1307**

3.1.3 XML Declarations

Presence or absence of an XML declaration does not affect interoperability. Certain implementations might always precede their XML serialization with the XML declaration.

R1010 A **RECEIVER** *MUST NOT* fault due to the presence of an XML Declaration in the SOAP envelope (as specified by Section 2.8 of [XML1.0], "[Prolog and Document Type Declaration](#)").
CORETESTABLE **BP1015**

3.1.4 Character Encodings

As a consequence of Section 4.3.3 of [XM1.0], "[Character Encoding in Entities](#)", which requires XML processors to support both the UTF-8 and UTF-16 character encodings, this Profile mandates that RECEIVERS support both UTF-8 and UTF-16 character encodings.

To improve interoperability, the "charset" parameter of Content-Type HTTP header field must be used to determine the correct character encoding of the message.

As this Profile allows the use of protocol bindings other than HTTP, the transport is responsible for defining how encoding is handled as specified in Section for Simple SOAP and XOP encoded messages using HTTP. This applies to this section and Section .

R1012 An **ENVELOPE** *MUST* be serialized using either UTF-8 or UTF-16 character encoding. **CORETESTABLE** **BP1018**

R1018 A **SIMPLE_SOAP_MESSAGE** *MUST* indicate the correct character encoding, using the "charset" parameter.
CORETESTABLE **BP1018**

R1019 A **RECEIVER** *MUST* ignore, if present, the encoding declaration of the envelope's XML declaration.
CORETESTABLE **SCENARIO_DEPENDENT** **BP1306**

3.1.5 XOP Encoded Messages

There exists some confusion among implementations about the proper encoding of the `action` parameter for XOP encoded messages. The `multipart/related` media type specification does not include an `action` parameter, though it does permit extensibility. Thus, the `action` parameter on the `multipart/relatedContent-Type` header has no defined semantic. The correct encoding is to include the `action` parameter inside the `start-info` parameter of the enclosing MIME `multipart/related` entity body as well as inside the `type` parameter of the root part. Nevertheless, existing SENDERS could emit an XOP message with the `action` parameter encoded as a separate parameter on the `Content-Type` header of the enclosing `multipart/related` MIME entity body. This Profile does not preclude a RECEIVER from accepting such a message.

See Section for conformance criteria when using HTTP.

R1020 A **XOP_ENCODED_MESSAGE** *MUST* include the `start-info` parameter in the `Content-Type` header of the enclosing `multipart/related` MIME entity body. **CORETESTABLE** **BP1020**

R1021 A **XOP_ENCODED_MESSAGE** *MUST* include the full value of the `type` parameter from the root entity body part inside the `start-info` parameter of the enclosing `multipart/related`

MIME entity body part's Content-Type header. CORETESTABLE
BP1021

R1022 A **RECEIVER MUST NOT** fault due to the *action* parameter of an XOP encoded message being included with the value of the *start-info* parameter inside the *Content-Type* header of the enclosing *multipart/related* MIME entity body.
CORENOT_TESTABLE

For example,

INCORRECT:

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
  type="application/xop+xml";
  start="<mymessage.xml@example.org>";
  start-info="application/soap+xml";
  action="ProcessData"
```

```
--MIME_boundary
Content-Type: application/xop+xml;
  charset=UTF-8;
  type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>
...
```

CORRECT:

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
  type="application/xop+xml";
  start="<mymessage.xml@example.org>";
  start-info="application/soap+xml; action=\"ProcessData\""
Content-Description: A SOAP message with my pic and sig in it

--MIME_boundary
Content-Type: application/xop+xml;
  charset=UTF-8;
  type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>
...
```

3.2 SOAP Envelopes

Section 5 of [SOAP1.2-1] [Soap Envelope](#) , defines a structure for composing messages, the "SOAP Envelope". The Profile mandates the use of that structure, and places the following constraints on its use.

3.2.1 SOAP Envelope Structure

There are obvious interoperability problems if different implementations do not agree on the number of allowable children for the `soap12:Body` element.

R9980 An **ENVELOPE MUST** conform to the structure specified in *SOAP Version 1.2 Part 1, Section 5, "SOAP Envelope"* (subject to amendment by the Profile). CORETESTABLE BP1600

R9981 An **ENVELOPE** *MUST* have exactly zero or one child elements of the `soap12:Body` element. CORETESTABLE BP1881

See the requirements in [Section 4.4.1](#) for the corresponding, requisite constraints on a DESCRIPTION.

3.2.2 SOAP Body Namespace Qualification

The use of unqualified element names may cause naming conflicts, therefore qualified names must be used for the children of `soap12:Body`.

R1014 The children of the `soap12:Body` element in an **ENVELOPE** *MUST* be namespace qualified. CORETESTABLE BP1202

3.2.3 Disallowed Constructs

XML DTDs and PIs may introduce security vulnerabilities, processing overhead and semantic ambiguity when used in envelopes. As a result, certain XML constructs are disallowed by section 5 of [SOAP1.2-1].

Although published errata NE05 (see <http://www.w3.org/XML/xml-names-19990114-errata>) allows the namespace declaration `xmlns:xml="http://www.w3.org/XML/1998/namespace"` to appear, some older processors considered such a declaration to be an error. These requirements ensure that conformant artifacts have the broadest interoperability possible.

R1008 An **ENVELOPE** *MUST NOT* contain a Document Type Declaration. CORETESTABLE BP1007

R1009 An **ENVELOPE** *MUST NOT* contain Processing Instructions. CORETESTABLE BP1208

R1033 An **ENVELOPE** *MUST NOT* contain the namespace declaration `xmlns:xml="http://www.w3.org/XML/1998/namespace"`. CORETESTABLE BP1033

3.2.4 xsi:type Attributes

In many cases, senders and receivers will share some form of type information related to the envelopes being exchanged.

R1017 A **RECEIVER** *MUST NOT* fault on the absence of the `xsi:type` attribute in envelopes, except in cases where this attribute is required to indicate a derived type (see [XML Schema Part 1: Structures, Section 2.6.1](#)). CORENOT_TESTABLE

3.2.5 SOAP 1.2 attributes on SOAP 1.2 elements

R1032 The `soap12:Envelope`, `soap12:Header`, and `soap12:Body` elements in an **ENVELOPE** *MUST NOT* have attributes in the namespace "`http://www.w3.org/2003/05/soap-envelope`". CORETESTABLE BP1032

3.3 SOAP Processing Model

Section 2 of [SOAP1.2-1] [SOAP Processing Model](#) defines a model for the processing of envelopes. In particular, it defines rules for the processing of header blocks and the envelope body. It also defines rules related to generation of faults. The Profile places the following constraints on the processing model:

3.3.1 SOAP Fault Processing

When a fault is generated, no further processing should be performed. In request-response exchanges, a fault message will be transmitted to the sender of the request, and some application level error will be flagged to the user.

Both SOAP and this Profile use the term 'generate' to denote the creation of a SOAP Fault. It is important to realize that generation of a Fault is distinct from its transmission, which in some cases is not required.

R1029 *Where the normal outcome of processing a SOAP envelope would have resulted in the transmission of a SOAP response, but rather a fault is generated instead, the **RECEIVER MUST NOT** transmit the non-faulting response.* CORENOT_TESTABLE

Note that there may be valid reasons (such as security considerations) why a fault might not be transmitted.

3.4 SOAP Faults

3.4.1 Identifying SOAP Faults

Some consumer implementations erroneously use only the HTTP status code to determine the presence of a Fault. Because there are situations where the Web infrastructure changes the HTTP status code, and for general reliability, the Profile requires that they examine the envelope. A Fault is an envelope that has a single child element of the `soap12:Body` element, that element being the `soap12:Fault` element.

R1107 *A **RECEIVER MUST** interpret a SOAP message as a Fault when the `soap12:Body` of the message has a single `soap12:Fault` child.* CORENOT_TESTABLE

3.5 Use of SOAP in HTTP

While SOAP itself is not transport specific, this Profile focuses on its use with HTTP and makes no requirements on the use of any other transport. Other profiles might be developed to focus on the particulars of other transports, but that is out of scope for this Profile. With respect to compliance to this Profile, any requirement that mentions the HTTP transport applies only when HTTP is being used. Any requirement that is not specific to HTTP (i.e. does not mention HTTP specifically) applies toward conformance regardless of the transport mechanism being used. For convenience, the HTTP transport-specific requirements have been identified and tagged as specified in Section .

Section 7 of [SOAP1.2-2] [SOAP HTTP Binding](#) defines a single protocol binding, for HTTP 1.1 RFC2616]. The Profile makes use of that binding, and places the following constraints on its use:

For this section, the conformance criteria for the use of HTTP as a transport protocol are specified.

3.5.1 HTTP Protocol Binding

Several versions of HTTP are defined. HTTP/1.1 [RFC2616] has performance advantages, and is more clearly specified than HTTP/1.0.

R1141 *When HTTP is used as the transport, a **MESSAGE MUST** be sent using either HTTP/1.1 or HTTP/1.0.* HTTP-TRANSPORT TESTABLE BP1002

R1140 *When HTTP is used as the transport, a **MESSAGE SHOULD** be sent using HTTP/1.1.* HTTP-TRANSPORT TESTABLE BP1001

Note that support for HTTP/1.0 is implied in HTTP/1.1, and that intermediaries may change the version of a message; for more information about HTTP versioning, see [RFC2145], "Use and Interpretation of HTTP Version Numbers."

3.5.2 Parameters on the Content-Type MIME Header

R1109 *If present, the values of the following parameters - type, start-info, action, and boundary - on the Content-Type MIME header field-value in a request MESSAGE MUST be a quoted string.* HTTP-TRANSPORT TESTABLE BP1006

3.5.3 HTTP Success Status Codes

HTTP uses the 2xx series of status codes to communicate success. In particular, 200 is the default for successful messages, but 202 can be used to indicate that a message has been submitted for processing. Additionally, other 2xx status codes may be appropriate, depending on the nature of the HTTP interaction.

R1124 *An INSTANCE MUST use a 2xx HTTP status code on a response message that indicates the successful outcome of a HTTP Request.* HTTP-TRANSPORT NOT_TESTABLE

R1111 *An INSTANCE SHOULD use a "200 OK" HTTP status code on a response message that contains an envelope that is not a fault.* HTTP-TRANSPORT TESTABLE BP1100

R1112 *An INSTANCE SHOULD use either a "200 OK" or "202 Accepted" HTTP status code for a response message that does not contain a SOAP envelope but indicates the successful outcome of a HTTP Request.* HTTP-TRANSPORT TESTABLE BP1101

Despite the fact that the HTTP 1.1 assigns different meanings to response status codes "200" and "202", in the context of the Profile they should be considered equivalent by the initiator of the request. The Profile accepts both status codes because some SOAP implementations have little control over the HTTP protocol implementation and cannot control which of these response status codes is sent.

3.5.4 HTTP Redirect Status Codes

There are interoperability problems with using many of the HTTP redirect status codes, generally surrounding whether to use the original method, or GET. The Profile mandates "307 Temporary Redirect", which has the semantic of redirection with the same HTTP method, as the correct status code for redirection. For more information, see the 3xx status code descriptions in [RFC2616].

R1130 *An INSTANCE MUST use the "307 Temporary Redirect" HTTP status code when redirecting a request to a different endpoint.* HTTP-TRANSPORT NOT_TESTABLE

[RFC2616] notes that user-agents should not automatically redirect requests; however, this requirement was aimed at browsers, not automated processes (which many Web services will be). Therefore, the Profile allows, but does not require, consumers to automatically follow redirections.

3.5.5 HTTP Cookies

The HTTP State Management Mechanism [RFC2965] ("Cookies") allows the creation of stateful sessions between Web browsers and servers. Because they are designed for hypertext browsing, Cookies do not have well-defined semantics for Web services, and, because they are external to the envelope, are not

accommodated by either SOAP 1.2 or WSDL 1.1. This Profile limits the ways in which Cookies can be used, without completely disallowing them.

R1122 An **INSTANCE** using Cookies **SHOULD** conform to RFC2965.
HTTP-TRANSPORTNOT_TESTED

R1121 An **INSTANCE** **SHOULD NOT** require consumer support for Cookies in order to function correctly. **HTTP-TRANSPORTNOT_TESTED**

The Profile recommends that cookies not be required by instances for proper operation; they should be a hint, to be used for optimization, without materially affecting the execution of the Web service.

3.5.6 Non-Addressable Consumers and Instances

Definition: non-addressable

A CONSUMER or INSTANCE is deemed "non-addressable" when, for whatever reason, it is either unwilling or unable to provide a network endpoint that is capable of accepting connections. This means that the CONSUMER or INSTANCE cannot service incoming HTTP connections and can only transmit HTTP Request messages and receive HTTP Response messages.

Non-addressable CONSUMERS and INSTANCES, by their nature, cannot service incoming HTTP connections. Therefore any ENVELOPEs that they receive, either as requests (in the case of INSTANCES) or responses (in the case of CONSUMERS), **MUST**, when HTTP is used, be carried in the entity-body of an HTTP Request message.

R1202 When a CONSUMER is non-addressable, a SOAP **ENVELOPE**, that is described by the output message of a WSDL operation supported by an INSTANCE, **MUST** be bound to a HTTP Response message. **HTTP-TRANSPORTTESTABLE BP1126a BP1126b**

R1203 When an INSTANCE is non-addressable, a SOAP **ENVELOPE**, that is described by the input message of a WSDL operation supported by the INSTANCE, **MUST** be bound to a HTTP Response message. **HTTP-TRANSPORTTESTABLE**

R1204 When an INSTANCE is non-addressable, a SOAP **ENVELOPE**, that is described by the output message of a WSDL operation supported by the INSTANCE, **MUST** be bound to a HTTP Request message. **HTTP-TRANSPORTTESTABLE**

Note that INSTANCES can poll for requests from CONSUMERS using mechanisms such as those described in WS Make Connection [WSMakeConn].

3.6 Use of URIs in SOAP

This section of the Profile incorporates the following specifications by reference:

- [RFC3986] : Uniform Resource Identifier (URI): Generic Syntax

Section 6 of [SOAP1.2-1] [Use of URIs in SOAP](#) describes the use URIs as identifiers. For example, the `role` attribute value is a URI that identifies the SOAP node(s) to which a particular header block is targeted. To ensure interoperability it is important that SENDERS and RECEIVERS share a common understanding of how such URI values will be compared. The Profile places the following constraints on the use of such URI values:

3.6.1 Use of SOAP-defined URIs

A SOAP 1.2 defined URI, such as the `role` value "http://www.w3.org/2003/05/soap-envelope/role/next", is treated as follows:

R1160 A **RECEIVER**, for the purposes of comparison of URI values of information items defined by the SOAP 1.2 specification, **MUST** treat the computed absolute URI values as simple strings as defined in RFC3986 (see [RFC3986, Section 6.2.1](#)).
CORENOT_TESTABLE

3.7 WS-Addressing Support

WS-Addressing is a part of core Web services infrastructure. To facilitate interoperability and to provide a common baseline, profiling of WS-Addressing is focusing on WS-Addressing Core, WS-Addressing SOAP Binding and WS-Addressing Metadata.

Support for WS-Addressing by a specific "service" is optional. However, a service may require the use of WS-Addressing, in which case, for successful interaction with that service, a client will need to support it.

Note that two BP compliant web services instances may both support the use of WS-Addressing yet fail to agree on a common set of features necessary to interact with one another. For example, a **RECEIVER** may require the use of non-anonymous response EPRs (and advertise this via the `wsam:NonAnonymousResponses` nested policy assertion) yet a **SENDER**, for various reasons (e.g. the presence of NATs or firewalls), may only support the use of anonymous response EPRs.

The following Requirements are profiling the use of WS-Addressing.

3.7.1 Requiring WS-Addressing SOAP Headers

R1040 If an endpoint requires use of WS-Addressing by use of a `wsam:Addressing` policy assertion, an **ENVELOPE** sent by a **SENDER** **MUST** carry all required WS-Addressing SOAP headers. **CORETESTABLE** BP1040a BP1040b BP1040c BP1142a BP1142b BP1142c BP1143a BP1143b BP1143c

3.7.2 NotUnderstood block in MustUnderstand Fault on WS-Addressing SOAP Headers

R1041 An **ENVELOPE** that is a `MustUnderstand` SOAP fault, sent from an endpoint that has a policy alternative containing the `wsam:Addressing` assertion attached to its WSDL endpoint subject, **MUST NOT** contain a `NotUnderstood` SOAP header block with the `qname` attribute value that identifies a WS-Addressing defined SOAP header block. **CORETESTABLE** BP1041

3.7.3 Use of `wsa:Action` and WS-Addressing 1.0 - Metadata

WS-Addressing 1.0 - Metadata, Section 5.1 **[WSAddrMeta]** defines additional constraints on the cardinality of WS-Addressing Message Addressing Properties defined in WS-Addressing 1.0 **Core** **[WSAddrCore]**. These constraints are defined for every message involved in WSDL 1.1 transmission primitives. The Profile requires conformance to this section when WS-Addressing is used in conjunction with a WSDL 1.1 description.

R1142 An **ENVELOPE** that includes a `wsa:Action` SOAP header block and which is described using a WSDL 1.1 description **MUST** conform to WS-Addressing 1.0 - Metadata, Section 5.1.

CORETESTABLE BP1142a BP1142b BP1142c BP1143a BP1143b BP1143c

3.7.4 Valid Values for action Parameter on the Content-Type MIME header When WS-Addressing is Used

There could be some confusion with regards to the range of valid values for the `action` parameter on the Content-Type MIME header when WS-Addressing is used.

When composed with WS-Addressing, the value of the `action` parameter, if present, is limited to the absolute URI that matches the value specified for `wsa:Action`. The `action` parameter is optional and is therefore not required to be present on the header. This is useful, for example, when the value of `wsa:Action` is sensitive and is therefore encrypted.

R1144 When the `wsa:Action` SOAP header block is present in an envelope, the value of the `action` parameter, if present, on the Content-Type MIME header of the **MESSAGE** **MUST** be an absolute URI that has the same value as the value of the `wsa:Action` header. HTTP-TRANSPORTTESTABLE BP1144

3.7.5 SOAP Defined Faults Action URI

WS-Addressing provides the URI `http://www.w3.org/2005/08/addressing/soap/fault` for "SOAP defined faults". However, it only recommends, rather than mandates its use for the SOAP 1.2 defined `MustUnderstand` and `VersionMismatch` faults. This Profile mandates the use of the WS-Addressing defined `wsa:Action` value for SOAP 1.2 defined `MustUnderstand` and `VersionMismatch` faults, for interoperability.

R1035 An **ENVELOPE** **MUST** use the `http://www.w3.org/2005/08/addressing/soap/fault` URI as the value for the `wsa:Action` SOAP header element, when present, for either of the SOAP 1.2 defined `VersionMismatch` and `MustUnderstand` faults. CORETESTABLE BP1035

3.7.6 Understanding WS-Addressing SOAP Header Blocks

WS-Addressing 1.0 - SOAP Binding [WSAddrSoap] defines multiple SOAP header blocks (`wsa:To`, `wsa:From`, `wsa:ReplyTo`, `wsa:FaultTo`, `wsa:Action`, `wsa:MessageID`, and `wsa:RelatesTo`). These SOAP header blocks are part of the same module. A SOAP node that conforms to the Profile understands and honors all of these SOAP header blocks (when it understands WS-Addressing) or none at all (when it does not understand WS-Addressing).

R1143 When a message contains multiple WS-Addressing SOAP header blocks with at least one of those header blocks containing a `soap12:mustUnderstand='1'` attribute, then a RECEIVER **MUST** honor all the WS-Addressing SOAP header blocks or none of them. CORETESTABLE BP1043a BP1043b

3.7.7 Ignored or Absent WS-Addressing Headers

When WS-Addressing headers are present in a SOAP envelope, but do not contain a `soap12:mustUnderstand='1'` attribute, a RECEIVER may choose to ignore these SOAP headers (per R1143). Consistent with R1036, valid reasons may exist why (not where) faults are not transmitted.

R1145 *If a SOAP envelope does not contain any WS-Addressing header blocks, or contains WS-Addressing header blocks that do not include any `soap12:mustUnderstand="1"` attributes, and the **RECEIVER** chooses to ignore them, then any response (normal or fault) **SHOULD** be transmitted. If it is transmitted then it is transmitted on the HTTP Response message (if available).* HTTP-TRANSPORTNOT_TESTED

3.7.8 Present and Understood WS-Addressing Headers

When any WS-Addressing header blocks are present in a SOAP envelope (where `soap12:mustUnderstand="1"` attributes exist or the header contents are understood), any non-faulting response will be transmitted to the endpoint referred to by the `wsa:ReplyTo` header. Should a fault be generated, it replaces the non-faulting response.

R1146 *A **RECEIVER** **MUST** transmit non-faulting responses to the endpoint referred to by the `wsa:ReplyTo` header or generate a fault instead (per [R1029](#)).* CORETESTABLE BP1146

SOAP 1.2 allows a RECEIVER to ignore headers that it does not understand. This behavior is particularly relevant for WS-Addressing headers that affect message processing and routing. As an example, take the following message sent to a SOAP node that does not understand the "http://www.w3.org/2005/08/addressing" namespace:

For example,

```
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <soap12:Header>
    <wsa:MessageID>uuid:8B82EA41-1485-13A6-5631527DC83F4168</wsa:MessageID>
    <wsa:Action>http://www.wstf.org/docs/scenarios/sc002/Echo</wsa:Action>
    <wsa:ReplyTo>
      <wsa:Address>http://server.foobie.com/NotifyEcho/asynchResp</wsa:Address>
    </wsa:ReplyTo>
    ...
  </soap12:Header>
  <soap12:Body>
    ...
  </soap12:Body>
</soap12:Envelope>
```

The SENDER expects the response to be sent "server.foobie.com". Yet, because it does not recognize the WS-Addressing 1.0 namespace, the RECEIVER will ignore the WS-Addressing headers as if WS-Addressing weren't engaged; consequently the SOAP response will be sent in the entity-body of the HTTP Response and may be missed by the SENDER.

Another example is where a message with an empty SOAP Body carries the semantic intent in its `wsa:Action` header.

In situations where the ability of the receiving node to understand WS-Addressing 1.0 headers is in doubt and the correct processing of the WS-Addressing is semantically significant (such as the two examples given), the SENDER is encouraged to add the `soap12:mustUnderstand` attribute with a value of "1" to the `wsa:Action` header. This prompts the RECEIVER to generate a MustUnderstand fault in cases where the WS-Addressing headers are not understood.

3.7.9 SOAP MustUnderstand or VersionMismatch fault Transmission

SOAP MustUnderstand and VersionMismatch faults are detected irrespective of the use of WS-Addressing headers. There may be valid reasons why (not where) faults are transmitted, e.g. security concerns or the HTTP Response connection is no longer available. In these cases the SENDER will not receive any SOAP envelope response.

R1036 *Regardless of whether the `wsa:ReplyTo` or `wsa:FaultTo` SOAP headers appear in the incoming message, a **RECEIVER** that receives a SOAP envelope that generates either a SOAP MustUnderstand or VersionMismatch fault **SHOULD** transmit either fault. If it is transmitted, such a fault is transmitted on the HTTP Response message (if available).* HTTP-TRANSPORTNOT_TESTED

3.7.10 Faulting Behavior with Present and Understood WS-Addressing Headers

When WS-Addressing headers are present in a SOAP envelope (where `soap12:mustUnderstand="1"` attributes exist or the header contents are understood), should a fault be generated, it will be transmitted to the endpoint referred to by the `wsa:FaultTo` header. WS-Addressing specifies expected behavior should the `wsa:FaultTo` header be absent.

R1147 *If a fault is generated, the **RECEIVER** **SHOULD** transmit the fault (per [R1029](#)).* CORENOT_TESTED

R1161 *Other than those faults specified in [R1036](#), faults in [R1147](#) **SHOULD** be transmitted by the **RECEIVER** as specified in [WS-Addressing 1.0 - Core, Section 3.4](#).* CORETESTABLE

R1162 *When the `wsa:FaultTo` SOAP header exists, the **RECEIVER** **MUST NOT** transmit faults to the endpoint referred to by the `wsa:ReplyTo` header.* CORETESTABLE

R1148 *If an error occurs when transmitting the fault in [R1147](#), a **RECEIVER** **MAY** choose to send a fault related to this transmission error on the HTTP Response (if available).* HTTP-TRANSPORTNOT_TESTED

Note: To avoid a recursive situation, if a fault is generated while trying to transmit to the endpoint referred to by the `wsa:ReplyTo` header ([R1146](#)) and the `wsa:FaultTo` header is absent, [R1147](#) does not apply.

3.7.11 [message id] and One-Way Operations

When sending a one-way message the SENDER could choose to ignore any possible response - for example, a fault. However, if the SENDER is interested in receiving those messages, the SENDER will need to include a [message id] property in the one-way message to ensure that the response can be successfully transmitted (see Section 3.4 of [WSAddrCore] [Formulating a Reply Message](#)).

R1163 *When applying the processing rules defined by WS-Addressing 1.0 – Core [WSAddrCore], Section 3.4, if a related message lacks a [message id] property, the **RECEIVER** MUST generate a `wsa:MessageAddressingHeaderRequired` fault.* CORETESTABLE

While the RECEIVER is under no obligation to transmit faults, including a [message id] property will provide the RECEIVER with sufficient information to generate a response if needed.

3.7.12 Refusal to Honor WS-Addressing Headers

There may be many reasons (e.g. security, unsupported wsa:Address values, ...) why a RECEIVER does not honor any WS-Addressing headers. In these cases and irrespective of where the condition occurs, when any WS-Addressing headers are present in a SOAP envelope (where soap12:mustUnderstand=1 attributes exist or the header contents are understood), the RECEIVER must generate a fault.

R1149 *If a **RECEIVER** detects one of the error conditions specified in Section 6.4 of the Web Services Addressing 1.0 - SOAP Binding {WSAddrSoap} , it MUST generate a fault using the [Code], [Subcode], and [Subsubcode] listed for that particular error condition.* CORETESTABLE BP1149a BP1149b BP1149c BP1149d

3.7.13 Use of Non-Anonymous Response EPRs

The WS-Addressing [destination] URI of an outgoing message influences where this message will be sent. In the case of the outgoing response (normal or fault), if this URI is a non-anonymous URI then this message will be sent over a separate HTTP connection from one used to carry the request message.

R1152 *If an **INSTANCE** attempts to send a message to a non-anonymous [destination] URI then the message MUST be transmitted in the entity-body of an HTTP Request.* CORETESTABLE BP1152a BP1152b BP1152c

3.7.14 Optionality of the wsa:To header

WS-Addressing 1.0 - Core [WSAddrCore] and WS-Addressing 1.0 - Metadata [WSAddrMeta] are unclear about whether and when the `wsa:To` header element is required in a SOAP message. This Profile makes the following, clarifying requirement.

R1153 *Except in cases in which an instance exposes a WSDL description and its endpoint includes a `wsdl:port` that has been extended with a `wsa:EndpointReference`, a **RECEIVER** MUST NOT fault a SOAP request message due to the absence of the `wsa:To` header.* CORETESTABLE BP1153a BP1153b

Although the `wsa:To` header is optional, as a matter of best practice implementations are encouraged to include this header (with a non-anonymous value) as its presence provides a greater degree of flexibility in handling certain situations; for example, when moving a service endpoint from one URI to another.

As per WS-Addressing 1.0 - Core, the [destination] message addressing property of a request message without a `wsa:To` header is "http://www.w3.org/2005/08/addressing/anonymous". Note that none of the WS-Addressing 1.0 specifications describes the semantics of sending a SOAP request message, over

HTTP, either without a `wsa:To` header or with a `wsa:To` header with the value of "http://www.w3.org/2005/08/addressing/anonymous". To clarify, such a request is considered to be addressed to "the entity listening at the URI of the HTTP Request that contains this message". Sent over a connection to http://www.example.org, the following three example messages are consistent:

For example,

CORRECT:

```
POST /NotifyEcho/soap12service HTTP/1.1
Content-Type: text/xml; charset=UTF-8
...
<soap12:Envelope ...>
  <soap12:Header>
    <wsa:Action>http://www.wstf.org/sc002/Echo</wsa:Action>
  </soap12:Header>
  <soap12:Body>
    ...
  </soap12:Body>
</soap12:Envelope>
```

CORRECT:

```
POST /NotifyEcho/soap12service HTTP/1.1
Content-Type: text/xml; charset=UTF-8
...
<soap12:Envelope ...>
  <soap12:Header>
    <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
    <wsa:Action>http://www.wstf.org/sc002/Echo</wsa:Action>
  </soap12:Header>
  <soap12:Body>
    ...
  </soap12:Body>
</soap12:Envelope>
```

CORRECT:

```
POST /NotifyEcho/soap12service HTTP/1.1
Content-Type: text/xml; charset=UTF-8
...
<soap12:Envelope ...>
  <soap12:Header>
    <wsa:To>http://www.example.org/NotifyEcho/soap12service</wsa:To>
    <wsa:Action>http://www.wstf.org/sc002/Echo</wsa:Action>
  </soap12:Header>
  <soap12:Body>
    ...
  </soap12:Body>
</soap12:Envelope>
```

3.7.15 Extending WSDL Endpoints with an EPR

WS-Addressing 1.0 - Metadata [WSAddrMeta] is unclear about the relationship between the elements of a WSDL 1.1 description of an endpoint and the values of the addressing properties of a message sent to that endpoint. In particular, the value of the [destination] message addressing property needs to be clarified in order to insure interoperability between SENDER and RECEIVER. There are two cases to consider. The first case is where the `wsdl:port` has been extended with a `wsa:EndpointReference` as described by Section 4.1 of [WSAddrMeta]. In this case the following requirement applies:

R1154 *When sending a request message to an endpoint which is specified by a WSDL 1.1 description in which the `wsdl:port` element has been extended with a `wsa:EndpointReference`, if*

*the `wsa:Action` SOAP header block is present, the **SENDER** **MUST** populate the `wsa:To` and reference parameter SOAP headers of that request message with the values of the `wsa:Address` and `wsa:ReferenceParameters` elements (respectively) of the extending endpoint reference.*
CORETESTABLE

Note that, since [address] is a required property of an endpoint reference, extending a `wsdl:port` with a `wsa:EndpointReference` has the effect of populating the [destination] property of the outgoing message, thus mandating the inclusion of the `wsa:To` header.

The second case is where the `wsdl:port` has not been extended with a `wsa:EndpointReference`.

R1155 *When sending a request message to an endpoint which is specified by a WSDL 1.1 description in which the `wsdl:port` element has **not** been extended with a `wsa:EndpointReference`, if the `wsa:Action` SOAP header block is present, the **SENDER** **MAY** populate the `wsa:To` SOAP header of that request message with the value of the `location` attribute of the `wssoap12:address` extension element.*
CORETESTABLE

3.7.16 Combining Synchronous and Asynchronous Operations

WS-Addressing 1.0 - Metadata [WSAddrMeta] defines a policy assertion, `wsam:Addressing`, that is used to indicate whether WS-Addressing is supported or required. It is a nested policy container assertion and can contain additional restrictions (specifically the `wsam:AnonymousResponses` and `wsam:NonAnonymousResponses` policy assertions) on the value of the response endpoint EPRs in request messages. A top-level assertion without any nested assertions implies that both anonymous and non-anonymous are allowed. The WS-Addressing 1.0 - Metadata specification sets the scope of this assertion to be endpoint policy subject. However, with regards to the anonymous/non-anonymous restrictions, experience has shown that it is often desirable to have different policies for different operations on the same endpoint. For example, some of the operations of an endpoint may need to be synchronous while others may need to be asynchronous. It is worthwhile to indicate this difference in a WSDL description. In the absence of any guidance on the mechanism(s) for expressing such per-operation distinctions, individual implementations will create their own extensions for enabling this feature. To avoid the interoperability problems inherent in such an approach, the Profile defines the following extension to the behavior defined by WS-Addressing 1.0 Metadata.

WS-Addressing 1.0 Metadata allows policies containing the `wsam:Addressing` policy assertion to be attached to either a `wsdl:port` or a `wsdl:binding`. To these two options the Profile adds a third option which allows policies containing the `wsam:Addressing` policy assertion to be attached to `wsdl:binding/wsdl:operation` elements. When the `wsam:Addressing` policy assertion is attached to the `wsdl:binding/wsdl:operation` element, it applies to the operation policy subject. Nevertheless, it should always be the case that if one operation of an endpoint supports or requires WS-Addressing, then all operations of that endpoint must support or require WS-Addressing (although, potentially, with different restrictions). Furthermore, to simplify the calculation of the effective policy for each operation and decrease the possibility of creating conflicting policies, each operation within such an endpoint should affirmatively declare its policy with respect to WS-Addressing.

R1156 *In a **DESCRIPTION**, a policy that contains the `wsam:Addressing` assertion **MUST** be attached to either a `wsdl:port`, a `wsdl:binding` or a `wsdl:binding/wsdl:operation`.*
CORENOT_TESTABLE_XPATH

R1157 If a **DESCRIPTION** has a policy alternative containing the *wsam:Addressing* assertion attached to a *wsdl:binding/wsdl:operation*, then all of the *wsdl:operations* within that *wsdl:binding* **MUST** also have a policy alternative containing the *wsam:Addressing* assertion attached to them.
CORENOT_TESTABLE_XPATH

In addition to the above restrictions and as stated in [R1158](#), the effective policy alternatives for a given policy subject must not contain conflicting assertions.

For example,

INCORRECT:

```
<wsdl:binding name="sc009SOAP12Binding" type="tns:sc009PortType">
  <wsp:Policy>
    <wsam:Addressing>
      <wsp:Policy/>
    </wsam:Addressing>
  </wsp:Policy>
  ...
  <wsdl:operation name="CreatePO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy>
          <wsam:NonAnonymousResponses/>
        </wsp:Policy>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="GetPOStatus">
    ...
  </wsdl:operation>

  <wsdl:operation name="UpdatePO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy>
          <wsam:NonAnonymousResponses/>
        </wsp:Policy>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="CancelPO">
    ...
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="sc009Service">
  <wsdl:port name="soap12port" binding="tns:sc009SOAP12Binding">
    ...
  </wsdl:port>
</wsdl:service>
```

The above example is incorrect for two reasons. Firstly, it violates R1157 because the GetPOStatus and CancelPO operations do not have policies containing the `wsam:Addressing` assertion attached to them. Secondly, the effective policies for both the CreatePO and UpdatePO operations contain conflicting assertions (a `wsam:Addressing` assertion that is unconstrained with regards to anonymous/non-anonymous and a `wsam:Addressing` assertion that is constrained to just non-anonymous) within the same alternative.

For example,

INCORRECT:

```
<wsdl:binding name="sc009SOAP12Binding" type="tns:sc009PortType">
  ...
  <wsdl:operation name="CreatePO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy>
          <wsam:NonAnonymousResponses/>
        </wsp:Policy>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="GetPOStatus">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy/>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="UpdatePO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy>
          <wsam:NonAnonymousResponses/>
        </wsp:Policy>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="CancelPO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy/>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="sc009Service">
  <wsdl:port name="soap12port" binding="tns:sc009SOAP12Binding">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy/>
      </wsam:Addressing>
    </wsp:Policy>
  </wsdl:port>
</wsdl:service>
```

```

    </wsam:Addressing>
  </wsp:Policy>
  ...
</wsdl:port>
</wsdl:service>

```

The above example is incorrect because the effective policies for both the CreatePO and UpdatePO operations contain conflicting assertions (a `wsam:Addressing` assertion that is unconstrained with regards to anonymous/non-anonymous and a `wsam:Addressing` assertion that is constrained to just non-anonymous) within the same alternative.

For example,

CORRECT:

```

<wsdl:binding name="sc009SOAP12Binding" type="tns:sc009PortType">
  ...
  <wsdl:operation name="CreatePO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy>
          <wsam:NonAnonymousResponses/>
        </wsp:Policy>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="GetPOStatus">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy/>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="UpdatePO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy>
          <wsam:NonAnonymousResponses/>
        </wsp:Policy>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>

  <wsdl:operation name="CancelPO">
    <wsp:Policy>
      <wsam:Addressing>
        <wsp:Policy/>
      </wsam:Addressing>
    </wsp:Policy>
    ...
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="sc009Service">

```

```
<wsdl:port name="soap12port" binding="tns:sc009SOAP12Binding">
  ...
</wsdl:port>
</wsdl:service>
```

The above example is correct. All of the operations in the soap12port of the s009Service require WS-Addressing. While the response EPRs for GetPOStatus and CancelPO are unconstrained, the response EPRs for the CreatePO and UpdatePO operations must be non-anonymous.

3.7.17 Conflicting Addressing Policies

When used together, the `wsam:AnonymousResponses` and `wsam:NonAnonymousResponses` nested policy assertions could result in an effective policy that contradicts WS-Addressing 1.0 - Metadata (i.e. "request messages sent to this endpoint must use response endpoint EPRs that simultaneously do and do not contain the WS-Addressing anonymous URI"). The Profile restricts the use of the `wsam:AnonymousResponses` and `wsam:NonAnonymousResponses` nested policy assertions to avoid this situation.

R1158 *In a DESCRIPTION the effective policy for a given endpoint MUST NOT contain both the `wsam:AnonymousResponses` and `wsam:NonAnonymousResponses` assertions within a single policy alternative.* `CORENOT_TESTABLE_XPATH`

4 Service Description

The Profile uses Web Services Description Language (WSDL) [WSDL1.1] to enable the description of services as sets of endpoints operating on messages.

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- Namespaces in XML 1.0 (Second Edition) [xmlNames]
- XML Schema Part 1: Structures [xmlSchema-1]
Extensibility points:
 - E0017 - Schema annotations - XML Schema allows for annotations, which may be used to convey additional information about data structures. CORE
- XML Schema Part 2: Datatypes [xmlSchema-2]
- Web Services Description Language (WSDL) 1.1 [WSDL1.1]
Extensibility points:
 - E0013 - WSDL extensions - WSDL allows extension elements and attributes in certain places, including the use and specification of alternate protocol binding extensions; use of such extensions requires out-of-band negotiation. CORE
 - E0014 - Validation mode - whether the parser used to read WSDL and XML Schema documents performs DTD validation or not. CORE
 - E0015 - Fetching of external resources - whether the parser used to read WSDL and XML Schema documents fetches external entities and DTDs. CORE
 - E0016 - Relative URIs - WSDL does not adequately specify the use of relative URIs for the following: wsoap12:body/@namespace, wsoap12:address/@location, wsdl:import/@location, xsd:schema/@targetNamespace and xsd:import/@schemaLocation. Their use may require further coordination; see XML Base for more information. CORE

These extensibility points are listed, along with any extensibility points from other sections of this Profile, in Appendix A.

4.1 Required Description

An instance of a Web service is required to make the contract that it operates under available in some fashion.

R0001 *Either an **INSTANCE**'s WSDL 1.1 description, its UDDI binding template, or both **MUST** be available to an authorized consumer upon request.* CORE TESTABLE BP2703

This means that if an authorized consumer requests a service description of a conformant service instance, then the service instance provider must make the WSDL document, the UDDI binding template, or both available to that consumer. A service instance may provide run-time access to WSDL documents from a server, but is not required to do so in order to be considered conformant. Similarly, a service instance provider may register the instance provider in a UDDI registry, but is not required to do so to be considered conformant. In all of these scenarios, the WSDL contract must exist, but might be made available through a variety of mechanisms, depending on the circumstances.

4.2 Document Structure

Section 2.1 of [WSDL1.1] [WSDL Document Structure](#) defines the overall structure of an XML document for describing Web services. The Profile mandates the use of that structure, and places the following constraints on its use.

Note that Section Document Structure , contains additional, corrective requirements on the structure of a WSDL 1.1 document.

4.2.1 WSDL Import location Attribute Structure

WSDL 1.1 is not clear about whether the `location` attribute of the `wsdl:import` statement is required, or what its content is required to be.

R2007 A DESCRIPTION *MUST specify a non-empty `location` attribute on the `wsdl:import` element.* CORETESTABLE BP2098

Although the `wsdl:import` statement is modeled after the `xsd:import` statement, the `location` attribute is required by `wsdl:import` while the corresponding attribute on `xsd:import`, `schemaLocation` is optional. Consistent with `location` being required, its content is not intended to be empty.

4.2.2 WSDL Import location Attribute Semantics

WSDL 1.1 is unclear about whether WSDL processors must actually retrieve and process the WSDL document from the URI specified in the `location` attribute on the `wsdl:import` statements it encounters.

R2008 A CONSUMER *MAY, but need not, retrieve a WSDL description from the URI specified in the `location` attribute on a `wsdl:import` element.* CORENOT_TESTED

The value of the `location` attribute of a `wsdl:import` element is a hint. A WSDL processor may have other ways of locating a WSDL description for a given namespace.

4.2.3 XML Version Requirements

Neither WSDL 1.1 nor XML Schema 1.0 mandate a particular version of XML. For interoperability, WSDL documents and the schemas they import expressed in XML must use version 1.0.

R4004 A DESCRIPTION *MUST use [XML 1.0].* CORENOT_TESTED

4.2.4 XML Namespace Declarations

Although published errata NE05 (see <http://www.w3.org/XML/xml-names-19990114-errata>) allows this namespace declaration to appear, some older processors considered such a declaration to be an error. This requirement ensures that conformant artifacts have the broadest interoperability possible.

R4005 A DESCRIPTION *SHOULD NOT contain the namespace declaration*
`xmlns:xml="http://www.w3.org/XML/1998/namespace".`
CORETESTABLE BP2034

4.2.5 WSDL and the Unicode BOM

XML 1.0 allows documents that use the UTF-8 character encoding to include a BOM; therefore, description processors must be prepared to accept them.

R4002 A **DESCRIPTION** MAY include the Unicode Byte Order Mark (BOM). **CORENOT_TESTED**

4.2.6 Acceptable WSDL Character Encodings

The Profile consistently requires either UTF-8 or UTF-16 encoding for both SOAP and WSDL.

R4003 A **DESCRIPTION** MUST use either UTF-8 or UTF-16 encoding. **CORETESTABLE** **BP2201**

4.2.7 Namespace Coercion

Namespace coercion on `wsdl:import` is disallowed by the Profile.

R2005 The *targetNamespace* attribute on the *wsdl:definitions* element of a description that is being imported MUST have same the value as the *namespace* attribute on the *wsdl:import* element in the importing **DESCRIPTION**. **CORETESTABLE** **BP2104**

4.2.8 WSDL Extensions

Requiring support for WSDL extensions that are not explicitly specified by this or another WS-I Profile can lead to interoperability problems with development tools that have not been instrumented to understand those extensions.

R2025 A **DESCRIPTION** containing WSDL extensions MUST NOT use them to contradict other requirements of the Profile. **CORENOT_TESTABLE**

R2026 A **DESCRIPTION** SHOULD NOT include extension elements with a *wsdl:required* attribute value of "true" on any WSDL construct (*wsdl:binding*, *wsdl:portType*, *wsdl:message*, *wsdl:types* or *wsdl:import*) that claims conformance to the Profile. **CORETESTABLE** **BP2123**

R2027 If during the processing of a description, a consumer encounters a WSDL extension element that has a *wsdl:required* attribute with a boolean value of "true" that the consumer does not understand or cannot process, the **CONSUMER MUST** terminate processing. **CORENOT_TESTABLE**

Development tools that consume a WSDL description and generate software for a Web service instance might not have built-in understanding of an unknown WSDL extension. Hence, use of required WSDL extensions should be avoided. Use of a required WSDL extension that does not have an available specification for its use and semantics imposes potentially insurmountable interoperability concerns for all but the author of the extension. Use of a required WSDL extension that has an available specification for its use and semantics reduces, but does not eliminate the interoperability concerns that lead to this refinement.

For the purposes of the Profile, all elements in the "http://schemas.xmlsoap.org/wsdl/" namespace are extensible via element as well as attributes. As a convenience, WS-I has published a version of the WSDL 1.1 schema that reflects this capability at: <http://ws-i.org/profiles/basic/1.1/wsdl-2004-08-24.xsd>

4.3 Types

Section 2.2 of [WSDL1.1] [Types](#) defines the `wsdl:types` element to enclose data type definitions that are relevant to the Web service described. The Profile places the following constraints pertinent to those portions of the content of the `wsdl:types` element that are referred to by WSDL elements that make Profile conformance claims:

4.3.1 QName References

XML Schema requires each QName reference to use either the target namespace, or an imported namespace (one marked explicitly with an `xsd:import` element). QName references to namespaces represented only by nested imports are not allowed.

WSDL 1.1 is unclear as to which schema target namespaces are suitable for QName references from a WSDL element. The Profile allows QName references from WSDL elements both to the target namespace defined by the `xsd:schema` element, and to imported namespaces. QName references to namespaces that are only defined through a nested import are not allowed.

R2101 A **DESCRIPTION** *MUST NOT* use QName references to WSDL components in namespaces that have been neither imported, nor defined in the referring WSDL document. [CORETESTABLE](#)
[BP2416](#)

R2102 A QName reference to a Schema component in a **DESCRIPTION** *MUST* use the namespace defined in the `targetNamespace` attribute on the `xsd:schema` element, or to a namespace defined in the `namespace` attribute on an `xsd:import` element within the `xsd:schema` element. [CORETESTABLE](#) [BP2417](#)

4.3.2 Schema targetNamespace Structure

Requiring a `targetNamespace` on all `xsd:schema` elements that are children of `wsdl:types` is a good practice, places a minimal burden on authors of WSDL documents, and avoids the cases that are not as clearly defined as they might be.

R2105 All `xsd:schema` elements contained in a `wsdl:types` element of a **DESCRIPTION** *MUST* have a `targetNamespace` attribute with a valid and non-null value, unless the `xsd:schema` element has `xsd:import` and/or `xsd:annotation` as its only child element(s). [CORETESTABLE](#) [BP2107](#)

4.3.3 soapenc:Array

The recommendations in WSDL 1.1 Section 2.2 for declaration of array types have been interpreted in various ways, leading to interoperability problems. Further, there are other clearer ways to declare arrays.

R2110 In a **DESCRIPTION**, declarations *MUST NOT* extend or restrict the `soapenc:Array` type. [CORETESTABLE](#) [BP2108b](#)

R2111 In a **DESCRIPTION**, declarations *MUST NOT* use `wsdl:arrayType` attribute in the type declaration. [CORETESTABLE](#)
[BP2108a](#)

R2112 In a **DESCRIPTION**, elements *SHOULD NOT* be named using the convention `ArrayOfXXX`. [CORETESTABLE](#) [BP2110](#)

R2113 An **ENVELOPE** **MUST NOT** include the *soapenc:arrayType* attribute. CORETESTABLE BP1204

For example,
INCORRECT:

Given the WSDL Description:

```
<xsd:element name="MyArray2" type="tns:MyArray2Type"/>
<xsd:complexType name="MyArray2Type"
  xmlns:soapenc="http://www.w3.org/2003/05/soap-encoding"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <xsd:complexContent>
    <xsd:restriction base="soapenc:Array">
      <xsd:sequence>
        <xsd:element name="x" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="tns:MyArray2Type[]" />
    </xsd:restriction> </xsd:complexContent>
  </xsd:complexType>
```

The envelope would serialize as (omitting namespace declarations for clarity):

```
<MyArray2 soapenc:arrayType="tns:MyArray2Type[]" ">
  <x>abcd</x>
  <x>efgh</x>
</MyArray2>
```

CORRECT:

Given the WSDL Description:

```
<xsd:element name="MyArray1" type="tns:MyArray1Type"/>
<xsd:complexType name="MyArray1Type">
  <xsd:sequence>
    <xsd:element name="x" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The envelope would serialize as (omitting namespace declarations for clarity):

```
<MyArray1>
  <x>abcd</x>
  <x>efgh</x>
</MyArray1>
```

4.3.4 WSDL and Schema Definition Target Namespaces

The names defined by schemas and the names assigned to WSDL definitions are in separate symbol spaces.

R2114 *The target namespace for WSDL definitions and the target namespace for schema definitions in a **DESCRIPTION** MAY be the same.* **CORE**NOT_TESTED

4.3.5 Multiple GED Definitions with the same QName

The schema components of all the `xs:schema` children, and their imports and includes, of the `wsdl:types` element comprise a single symbol space containing all the global element declarations. Thus, when global element declarations share a qualified name, a single component will be represented in the symbol space. If two declarations are identical, there is no ambiguity in the structure of the component, but if the declarations differ, it is indeterminate as to which of the declarations will be represented, which may lead to interoperability problems. Because defining an equivalence algorithm is impractical, this requirement warns against any appearance of declarations with the same qualified name. However, duplicate declarations are not strictly prohibited, as user inspection may determine that two declarations are actually identical (e.g. they were imported from the same set of components) and thus are unlikely to cause interoperability problems.

R2115 *A **DESCRIPTION** SHOULD NOT contain multiple global element declarations that share the same qualified name.* **CORE**TESTABLE
BP2124

4.3.6 Multiple Type Definitions with the same QName

The schema components of all the `xs:schema` children, and their imports and includes, of the `wsdl:types` element comprise single symbol spaces containing all the type definitions. Thus, when type definitions share a qualified name, a single component will be represented in the symbol space. If two definitions are identical, there is no ambiguity in the structure of the component, but if the definitions differ, it is indeterminate as to which of the definitions will be represented, which may lead to interoperability problems. Because defining an equivalence algorithm is impractical, this requirement warns against any appearance of definitions with the same qualified name. However, duplicate definitions are not strictly prohibited, as user inspection may determine that two definitions are actually identical (e.g. they were imported from the same set of components) and thus are unlikely to cause interoperability problems.

R2116 *A **DESCRIPTION** SHOULD NOT contain multiple type definitions that share the same qualified name.* **CORE**TESTABLE
BP2125

4.4 Messages

Section 2.3 of [WSDL1.1] [Messages](#) defines the `wsdl:message` elements that are used to represent abstract definitions of the data being transmitted. It uses `wsdl:binding` elements to define how the abstract definitions are bound to a specific message serialization. The Profile places the following constraints on `wsdl:message` elements and on how conformant `wsdl:binding` elements may use `wsdl:message` element(s).

Note that Section [Message](#) contains additional, corrective requirements on the structure of `wsdl:message` elements.

4.4.1 TBindings and Parts

There are various interpretations about how many `wsdl:part` elements are permitted or required for document-literal and rpc-literal bindings and how they must be defined.

R2201 A document-literal binding in a **DESCRIPTION** **MUST**, in each of its `wsoap12:body` element(s), have at most one part listed in the `parts` attribute, if the `parts` attribute is specified. CORETESTABLE BP2111

R2210 If a document-literal binding in a **DESCRIPTION** does not specify the `parts` attribute on a `wsoap12:body` element, the corresponding abstract `wsdl:message` **MUST** define zero or one `wsdl:part`s. CORETESTABLE BP2119

R2202 A `wsdl:binding` in a **DESCRIPTION** **MAY** contain `wsoap12:body` element(s) that specify that zero parts form the `soap12:Body`. CORENOT_TESTED

R2203 An rpc-literal binding in a **DESCRIPTION** **MUST** refer, in its `wsoap12:body` element(s), only to `wsdl:part` element(s) that have been defined using the `type` attribute. CORETESTABLE BP2013

R2211 An **ENVELOPE** described with an rpc-literal binding **MUST NOT** have the `xsi:nil` attribute with a value of "1" or "true" on the part accessors. CORETESTABLE BP1211a BP1211b

R2207 A `wsdl:message` in a **DESCRIPTION** **MAY** contain `wsdl:parts` that use the `elements` attribute provided those `wsdl:parts` are not referred to by a `wsoap12:body` in an rpc-literal binding. CORENOT_TESTED

R2204 A document-literal binding in a **DESCRIPTION** **MUST** refer, in each of its `wsoap12:body` element(s), only to `wsdl:part` element(s) that have been defined using the `element` attribute. CORETESTABLE BP2012

R2208 A binding in a **DESCRIPTION** **MAY** contain `wsoap12:header` element(s) that refer to `wsdl:parts` in the same `wsdl:message` that are referred to by its `wsoap12:body` element(s). CORENOT_TESTED

R2212 An **ENVELOPE** described using an rpc-literal binding **MUST** contain exactly one part accessor element for each of the `wsdl:part` elements bound to the `wsoap12:body` element in the rpc-literal binding corresponding to the envelope. CORETESTABLE BP1212a BP1212b

R2213 In a doc-literal description where the value of the `parts` attribute of `wsoap12:body` is an empty string, the corresponding **ENVELOPE** **MUST** have no element content in the `soap12:Body` element. CORETESTABLE BP1213a BP1213b

R2214 In a rpc-literal description where the value of the `parts` attribute of `wsoap12:body` is an empty string, the corresponding

ENVELOPE *MUST* have no part accessor elements.

CORETESTABLE BP1214a BP1214b

Use of `wsdl:message` elements with zero parts is permitted in Document styles to permit operations that can send or receive envelopes with empty `soap12:Body`s. Use of `wsdl:message` elements with zero parts is permitted in RPC styles to permit operations that have no (zero) parameters and/or a return value.

For document-literal bindings, the Profile requires that at most one part, abstractly defined with the `element` attribute, be serialized into the `soap12:Body` element.

When a `wsdl:part` element is defined using the `type` attribute, the serialization of that part in a message is equivalent to an implicit (XML Schema) qualification of a `minOccurs` attribute with the value "1", a `maxOccurs` attribute with the value "1" and a `nillable` attribute with the value "false".

It is necessary to specify the equivalent implicit qualification because the `wsdl:part` element does not allow one to specify the cardinality and nillability rules. Specifying the cardinality and the nillability rules facilitates interoperability between implementations. The equivalent implicit qualification for nillable attribute has a value of "false" because if it is specified to be "true" one cannot design a part whereby the client is always required to send a value. For applications that want to allow the `wsdl:part` to be nillable, it is expected that applications will generate a complexType wrapper and specify the nillability rules for the contained elements of such a wrapper.

4.4.2 Bindings and Faults

There are several interpretations for how `wsdl:part` elements that describe `wsoap12:fault`, `wsoap12:header`, and `wsoap12:headerfault` may be defined.

R2205 A *wsdl:binding* in a **DESCRIPTION** **MUST** refer, in each of its *wsoap12:header*, *wsoap12:headerfault* and *wsoap12:fault* elements, only to *wsdl:part* element(s) that have been defined using the *element* attribute. CORETESTABLE BP2113

Because faults and headers do not contain parameters, `wsoap12:fault`, `wsoap12:header` and `wsoap12:headerfault` assume, per WSDL 1.1, that the value of the `style` attribute is "document". R2204 requires that all `wsdl:part` elements with a `style` attribute whose value is "document" that are bound to `wsoap12:body` be defined using the `element` attribute. This requirement does the same for `wsoap12:fault`, `wsoap12:header` and `wsoap12:headerfault` elements.

4.4.3 Unbound portType Element Contents

WSDL 1.1 is not explicit about whether it is permissible for a `wsdl:binding` to leave the binding for portions of the content defined by a `wsdl:portType` unspecified.

R2209 A *wsdl:binding* in a **DESCRIPTION** **SHOULD** bind every *wsdl:part* of a *wsdl:message* in the *wsdl:portType* to which it refers to one of *wsoap12:body*, *wsoap12:header*, *wsoap12:fault* or *wsoap12:headerfault*. CORETESTABLE BP2114

A `portType` defines an abstract contract with a named set of operations and associated abstract messages. Although not disallowed, it is expected that every part of the abstract input, output and fault messages specified in a `portType` is bound to `wsoap12:body` or `wsoap12:header` (and so forth) as appropriate when using the SOAP binding as defined in WSDL 1.1 Section 3. Un-bound `wsdl:parts` should be ignored.

4.5 Port Types

Section 2.4 of [WSDL1.1] [Port Types](#) defines the `wsdl:portType` elements that are used to group a set of abstract operations. The Profile places the following constraints on conformant `wsdl:portType` element(s):

4.5.1 Ordering of part Elements

Permitting the use of `parameterOrder` helps code generators in mapping between method signatures and messages on the wire.

R2301 *The order of the elements in the `soap12:Body` of an **ENVELOPE** MUST be the same as that of the `wsdl:parts` in the `wsdl:message` that describes it for each of the `wsdl:part` elements bound to the envelope's corresponding `wsdl:body` element.* [CORETESTABLE](#) [BP1111a](#) [BP1111b](#) [BP1012a](#) [BP1012b](#)

R2302 *A **DESCRIPTION** MAY use the `parameterOrder` attribute of an `wsdl:operation` element to indicate the return value and method signatures as a hint to code generators.* [CORENOT_TESTED](#)

4.5.2 Allowed Operations

Solicit-Response and Notification operations are not well defined by WSDL 1.1; furthermore, WSDL 1.1 does not define bindings for them.

R2303 *A **DESCRIPTION** MUST NOT use Solicit-Response and Notification type operations in a `wsdl:portType` definition.* [CORETESTABLE](#) [BP2208](#)

4.5.3 Distinctive Operations

Operation name overloading in a `wsdl:portType` is disallowed by the Profile.

R2304 *A `wsdl:portType` in a **DESCRIPTION** MUST have operations with distinct values for their `name` attributes.* [CORETESTABLE](#) [BP2010](#)

Note that this requirement applies only to the `wsdl:operations` within a given `wsdl:portType`. A `wsdl:portType` may have `wsdl:operations` with names that are the same as those found in other `wsdl:portTypes`.

4.5.4 parameterOrder Attribute Construction

WSDL 1.1 does not clearly state how the `parameterOrder` attribute of the `wsdl:operation` element (which is the child of the `wsdl:portType` element) should be constructed.

R2305 *A `wsdl:operation` element child of a `wsdl:portType` element in a **DESCRIPTION** MUST be constructed so that the `parameterOrder` attribute, if present, omits at most 1 `wsdl:part` from the output message.* [CORETESTABLE](#) [BP2014](#)

If a `wsdl:part` from the output message is omitted from the list of `wsdl:parts` that is the value of the `parameterOrder` attribute, the single omitted `wsdl:part` is the return value. There are no restrictions on the type of the return value. If no part is omitted, there is no return value.

4.5.5 Exclusivity of type and element Attributes

WSDL 1.1 does not clearly state that both `type` and `element` attributes cannot be specified to define a `wsdl:part` in a `wsdl:message`.

R2306 A *wsdl:message* in a **DESCRIPTION** **MUST NOT** specify both *type* and *element* attributes on the same *wsdl:part*.
CORETESTABLE BP2116

4.6 Bindings

In WSDL 1.1, the `wsdl:binding` element supplies the concrete protocol and data format specifications for the operations and messages defined by a particular `wsdl:portType`. The Profile places the following constraints on conformant binding specifications:

4.6.1 Use of SOAP Binding

The Profile limits the choice of bindings to the well-defined and most commonly used SOAP 1.2 binding.

R2401 A *wsdl:binding* element in a **DESCRIPTION** **MUST** use the *SOAP 1.2 binding as defined in the WSDL 1.1 Binding Extension for SOAP 1.2*. CORETESTABLE BP2402

Note that this places a requirement on the construction of conformant `wsdl:binding` elements. It does not place a requirement on descriptions as a whole; in particular, it does not preclude WSDL documents from containing non-conformant `wsdl:binding` elements. Also, a binding may have WSDL extensibility elements present which change how messages are serialized.

4.7 SOAP Binding

The WSDL 1.1 Binding Extension for SOAP 1.2 [WSDL11SOAP12] defines a binding for SOAP 1.2 endpoints. This Profile mandates the use of the SOAP 1.2 binding as defined in WSDL 1.1 Binding Extension for SOAP 1.2, and places the following constraints on its use:

Note that Section SOAP Binding contains additional, corrective requirements on the use of the SOAP 1.2 binding.

4.7.1 HTTP Transport

The profile limits the underlying transport protocol to HTTP.

R2702 When HTTP is used, a *wsdl:binding* element in a **DESCRIPTION** **MUST** specify the HTTP transport protocol with SOAP binding. Specifically, the *transport* attribute of its *wsoap12:binding* child **MUST** have the value *"http://schemas.xmlsoap.org/soap/http"*. HTTP-TRANSPORTTESTABLE BP2404

Note that this requirement does not prohibit the use of HTTPS; See R5000.

4.7.2 Consistency of style Attribute

The *style*, "document" or "rpc", of an interaction is specified at the `wsdl:operation` level, permitting `wsdl:bindings` whose `wsdl:operations` have different *styles*. This has led to interoperability problems. Additionally, use of document-literal binding, which generally allows for simpler

implementations than the rpc-literal binding, is encouraged. This hint is not always appropriate, especially in the case of some existing implementations, which continue to be supported by this profile.

R2705 A *wsdl:binding* in a **DESCRIPTION** **MUST** either be a rpc-literal binding or a document-literal binding. CORETESTABLE BP2017

4.7.3 Encodings and the use Attribute

The Profile prohibits the use of encodings, including the SOAP encoding.

R2706 A *wsdl:binding* in a **DESCRIPTION** **MUST** use the value of "literal" for the use attribute in all *wssoap12:body*, *wssoap12:fault*, *wssoap12:header* and *wssoap12:headerfault* elements. CORETESTABLE BP2406

4.7.4 Multiple Bindings for portType Elements

The Profile explicitly permits multiple bindings for the same portType.

R2709 A *wsdl:portType* in a **DESCRIPTION** **MAY** have zero or more *wsdl:bindings* that refer to it, defined in the same or other WSDL documents. CORENOT_TESTED

4.7.5 Operation Signatures

Definition: operation signature

The Profile defines the "operation signature" to be the fully qualified name of the child element of SOAP body of the SOAP input message described by an operation in a WSDL binding and the URI value of the *wsa:Action* SOAP header block, if present.

In the case of rpc-literal binding, the operation name is used as a wrapper for the part accessors. In the document-literal case, since a wrapper with the operation name is not present, the message signatures must be correctly designed so that they meet this requirement.

An endpoint that supports multiple operations must unambiguously identify the operation being invoked based on the input message that it receives. This is only possible if all the operations specified in the *wsdl:binding* associated with an endpoint have a unique operation signature.

R2710 The operations in a *wsdl:binding* in a **DESCRIPTION** **MUST** result in operation signatures that are different from one another. CORETESTABLE BP2120a BP2120b

4.7.6 Multiple Ports on an Endpoint

When input messages destined for two different *wsdl:ports* at the same network endpoint are indistinguishable on the wire, it may not be possible to determine the *wsdl:port* being invoked by them. This may cause interoperability problems. However, there may be situations (e.g., SOAP versioning, application versioning, conformance to different profiles) where it is desirable to locate more than one port on an endpoint; therefore, the Profile allows this.

R2711 A **DESCRIPTION** **SHOULD NOT** have more than one *wsdl:port* with the same value for the location attribute of the *wssoap12:address* element. CORETESTABLE BP2711

4.7.7 Child Element for Document-Literal Bindings

WSDL 1.1 is not completely clear what, in document-literal style bindings, the child element of `soap12:Body` is.

R2712 A document-literal binding *MUST* be serialized as an **ENVELOPE** with a `soap12:Body` whose child element is an instance of the global element declaration referenced by the corresponding `wsdl:message` part. CORETESTABLE BP1011a BP1011b

4.7.8 One-Way Operations

There are differing interpretations of how HTTP is to be used when performing one-way operations. The SOAP 1.2 Request-Response SOAP MEP specifies the expectations for the SOAP/HTTP binding.

R2714 For one-way operations, an HTTP Response **MESSAGE** MAY contain an envelope. HTTP-TRANSPORTNOT_TESTED

R2727 For one-way operations, a **CONSUMER** *MUST NOT* interpret a successful HTTP Response status code (i.e., 2xx) to mean the message is valid or that the receiver would process it. HTTP-TRANSPORTNOT_TESTABLE

One-way operations typically do not produce SOAP responses. However, some INSTANCES may choose to communicate infrastructure-related faults (e.g. MustUnderstand, VersionMismatch) in the HTTP Response message. In addition to this, the use of some protocol extensions (e.g. WS-ReliableMessaging) may create the possibility for non-empty responses to one-way messages. For these reasons the Basic Profile 1.1 requirement that the HTTP Response message not contain a SOAP envelope has been relaxed. Note: the fact that an INSTANCE may choose to communicate infrastructure-related faults in the HTTP Response does not mean that the CONSUMER can expect it to do so.

The HTTP Response to a one-way operation indicates the success or failure of the transmission of the message. Based on the semantics of the different response status codes supported by the HTTP protocol, the Profile specifies that "200" and "202" are the preferred status codes that the sender should expect, signifying that the one-way message was received. A successful transmission does not indicate that the SOAP processing layer and the application logic has had a chance to validate the envelope or have committed to processing it.

4.7.9 Namespaces for wsoap12 Elements

There is confusion about what namespace is associated with the child elements of various children of `soap12:Envelope`, which has led to interoperability difficulties. The Profile defines these.

R2716 A document-literal binding in a **DESCRIPTION** *MUST NOT* have the `namespace` attribute specified on contained `wsoap12:body`, `wsoap12:header`, `wsoap12:headerfault` and `wsoap12:fault` elements. CORETESTABLE BP2019

R2717 An rpc-literal binding in a **DESCRIPTION** *MUST* have the `namespace` attribute specified, the value of which *MUST* be an absolute URI, on contained `wsoap12:body` elements. CORETESTABLE BP2020

R2726 An rpc-literal binding in a **DESCRIPTION** *MUST NOT* have the `namespace` attribute specified on contained `wsoap12:header`, `wsoap12:headerfault` and `wsoap12:fault` elements. CORETESTABLE BP2117

In a document-literal SOAP binding, the serialized element child of the `soap12:Body` gets its namespace from the `targetNamespace` of the schema that defines the element. Use of the `namespace` attribute of the `wsoap12:body` element would override the element's namespace. This is not allowed by the Profile.

Conversely, in a rpc-literal SOAP binding, the serialized child element of the `soap12:Body` element consists of a wrapper element, whose namespace is the value of the `namespace` attribute of the `wsoap12:body` element and whose local name is either the name of the operation or the name of the operation suffixed with "Response". The `namespace` attribute is required, as opposed to being optional, to ensure that the children of the `soap12:Body` element are namespace-qualified.

4.7.10 Consistency of portType and binding Elements

The WSDL description must be consistent at both `wsdl:portType` and `wsdl:binding` levels.

R2718 A *wsdl:binding* in a **DESCRIPTION** **MUST** have the same set of *wsdl:operations* as the *wsdl:portType* to which it refers. **CORE** **TESTABLE** **BP2118**

4.7.11 Enumeration of Faults

A Web service description should include all faults known at the time the service is defined. There is also need to permit generation of new faults that had not been identified when the Web service was defined.

R2740 A *wsdl:binding* in a **DESCRIPTION** **SHOULD** contain a *wsoap12:fault* describing each known fault. **CORE** **NOT_TESTABLE**

R2741 A *wsdl:binding* in a **DESCRIPTION** **SHOULD** contain a *wsoap12:headerfault* describing each known header fault. **CORE** **NOT_TESTABLE**

R2742 An **ENVELOPE** **MAY** contain a fault with a *detail* element that is not described by a *wsoap12:fault* element in the corresponding WSDL description. **CORE** **NOT_TESTABLE**

R2743 An **ENVELOPE** **MAY** contain the details of a header processing related fault in a SOAP header block that is not described by a *wsoap12:headerfault* element in the corresponding WSDL description. **CORE** **NOT_TESTABLE**

4.7.12 Consistency of Envelopes with Descriptions

These requirements specify that when an instance receives an envelope that does not conform to the WSDL description, a fault should be generated unless the instance takes it upon itself to process the envelope regardless of this.

As specified by the SOAP processing model, (a) a "VersionMismatch" faultcode must be generated if the namespace of the "Envelope" element is incorrect, (b) a "MustUnderstand" fault must be generated if the instance does not understand a SOAP header block with a value of "1" for the `soap12:mustUnderstand` attribute. In all other cases where an envelope is inconsistent with its WSDL description, a fault with a "Client" faultcode should be generated.

R2724 If an **INSTANCE** receives an envelope that is inconsistent with its WSDL description, it **SHOULD** generate a *soap12:Fault* with a faultcode of "Client", unless a "MustUnderstand" or "VersionMismatch" fault is generated. **CORE** **NOT_TESTED**

R2725 If an **INSTANCE** receives an envelope that is inconsistent with its WSDL description, it **MUST** check for "VersionMismatch",

"MustUnderstand" and "Client" fault conditions in that order.

CORENOT_TESTABLE

4.7.13 Response Wrappers

[WSDL1.1] Section 3.5 could be interpreted to mean the RPC response wrapper element must be named identical to the name of the `wsdl:operation`.

R2729 An **ENVELOPE** described with an *rpc-literal binding* that is a response **MUST** have a wrapper element whose name is the corresponding `wsdl:operation` name suffixed with the string "Response". CORETESTABLE BP1005

4.7.14 Part Accessors

For *rpc-literal envelopes*, WSDL 1.1 is not clear what namespace, if any, the accessor elements for parameters and return value are a part of. Different implementations make different choices, leading to interoperability problems.

R2735 An **ENVELOPE** described with an *rpc-literal binding* **MUST** place the part accessor elements for parameters and return value in no namespace. CORETESTABLE BP1008a BP1008b

R2755 The part accessor elements in a **MESSAGE** described with an *rpc-literal binding* **MUST** have a local name of the same value as the *name* attribute of the corresponding `wsdl:part` element. CORETESTABLE BP1755a BP1755b

Settling on one alternative is crucial to achieving interoperability. The Profile places the part accessor elements in no namespace as doing so is simple, covers all cases, and does not lead to logical inconsistency.

4.7.15 Namespaces for Children of Part Accessors

For *rpc-literal envelopes*, WSDL 1.1 is not clear on what the correct namespace qualification is for the child elements of the part accessor elements when the corresponding abstract parts are defined to be of types from a different namespace than the `targetNamespace` of the WSDL description for the abstract parts.

R2737 An **ENVELOPE** described with an *rpc-literal binding* **MUST** namespace qualify the descendents of part accessor elements for the parameters and the return value, as defined by the schema in which the part accessor types are defined. CORENOT_TESTABLE_XPATH

[WSDL1.1] Section 3.5 states: "The part names, types and value of the namespace attribute are all inputs to the encoding, although the namespace attribute only applies to content not explicitly defined by the abstract types."

However, it does not explicitly state that the element and attribute content of the abstract (`complexType`) types is namespace qualified to the `targetNamespace` in which those elements and attributes were defined. WSDL 1.1 was intended to function in much the same manner as XML Schema. Hence, implementations must follow the same rules as for XML Schema. If a `complexType` defined in `targetNamespace` "A" were imported and referenced in an element declaration in a schema with `targetNamespace` "B", the element and attribute content of the child elements of that `complexType` would be qualified to namespace "A" and the element would be qualified to namespace "B".

For example,

CORRECT:

Given this WSDL, which defines some schema in the "http://example.org/foo/" namespace in the `wsdl:types` section contained within a `wsdl:definitions` that has a `targetNamespace` attribute with the value "http://example.org/bar/" (thus, having a type declared in one namespace and the containing element defined in another);

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsoap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:bar="http://example.org/bar/"
  targetNamespace="http://example.org/bar/"
  xmlns:foo="http://example.org/foo/">
  <types>
    <xsd:schema targetNamespace="http://example.org/foo/"
      xmlns:tns="http://example.org/foo/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      attributeFormDefault="unqualified">
      <xsd:complexType name="fooType">
        <xsd:sequence>
          <xsd:element ref="tns:bar"/>
          <xsd:element ref="tns:baf"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="bar" type="xsd:string"/>
      <xsd:element name="baf" type="xsd:integer"/>
    </xsd:schema>
  </types>

  <message name="BarMsg">
    <part name="BarAccessor" type="foo:fooType"/>
  </message>

  <portType name="BarPortType">
    <operation name="BarOperation">
      <input message="bar:BarMsg"/>
    </operation>
  </portType>

  <binding name="BarSOAP12Binding" type="bar:BarPortType">
    <wsoap12:binding transport="http://schemas.xmlsoap.org/soap/http"
style="rpc"/>
    <operation name="BarOperation">
      <input>
        <wsoap12:body use="literal" namespace="http://example.org/bar/" />
      </input>
    </operation>
  </binding>

  <service name="serviceName">
    <port name="BarSOAPPort" binding="bar:BarSOAP12Binding">
      <wsoap12:address location="http://example.org/myBarSOAPPort"/>
    </port>
  </service>
</definitions>
```


The resulting envelope for BarOperation is:

```
<s:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:foo="http://example.org/foo/">
  <soap12:Header/>
  <soap12:Body>
    <m:BarOperation xmlns:m="http://example.org/bar/">
      <BarAccessor>
        <foo:bar>String</foo:bar>
        <foo:baf>0</foo:baf>
      </BarAccessor>
    </m:BarOperation>
  </soap12:Body>
</soap12:Envelope>
```

4.7.16 Required Headers

WSDL 1.1 does not clearly specify whether all `wsoap12:headers` specified on the `wsdl:input` or `wsdl:output` elements of a `wsdl:operation` element in the SOAP binding section of a WSDL description must be included in the resultant envelopes when they are transmitted. The Profile makes all such headers mandatory, as there is no way in WSDL 1.1 to mark a header optional.

R2738 An **ENVELOPE** *MUST* include all `wsoap12:headers` specified on a `wsdl:input` or `wsdl:output` of a `wsdl:operation` of a `wsdl:binding` that describes it. CORETESTABLE BP1009a BP1009b
BP1009c BP1009d

4.7.17 Allowing Undescribed Headers

Headers are SOAP's extensibility mechanism. Headers that are not defined in the WSDL description may need to be included in the envelopes for various reasons.

R2739 An **ENVELOPE** *MAY* contain SOAP header blocks that are not described in the `wsdl:binding` that describes it. CORENOT_TESTED

R2753 An **ENVELOPE** containing SOAP header blocks that are not described in the appropriate `wsdl:binding` *MAY* have the `mustUnderstand` attribute on such SOAP header blocks set to '1'. CORENOT_TESTED

4.7.18 Ordering Headers

There is no correlation between the order of `wsoap12:headers` in the description and the order of SOAP header blocks in the envelope. Similarly, more than one instance of each specified SOAP header block may occur in the envelope.

R2751 The order of `wsoap12:header` elements in `wsoap12:binding` sections of a **DESCRIPTION** *MUST* be considered independent of the order of SOAP header blocks in the envelope. CORENOT_TESTABLE

R2752 An **ENVELOPE** *MAY* contain more than one instance of each SOAP header block for each `wsoap12:header` element in the

appropriate child of `wssoap12:binding` in the corresponding description. **CORENOT_TESTED**

4.7.19 Describing action Parameter on the Content-Type MIME Header

See Section Parameters on the Content-Type MIME Header

R2744 If the *action* parameter on the HTTP *Content-Type* header is present in a **MESSAGE**, its value **MUST** be equal to the value of the *soapAction* attribute of the corresponding *wssoap12:operation* in the WSDL description, if this attribute is present and not empty. **HTTP-TRANSPORTTESTABLE BP1116a BP1116b**

See also **R1109** and related requirements for more discussion of action parameter on the *Content-Type* MIME header.

For example,

CORRECT:

A WSDL Description that has:

```
<wssoap12:operation soapAction="http://example.org/foo"/>
```

results in a message with the *action* parameter on the *Content-Type* MIME header field-value set to:

```
"http://example.org/foo"
```

4.7.20 SOAPAction HTTP Header

Unlike SOAP 1.1, SOAP 1.2 HTTP binding does not use the SOAPAction HTTP header in the HTTP Request messages. Relying on the presence of SOAPAction HTTP header or the value of SOAPAction HTTP header may result in interoperability problems.

R2760 A **RECEIVER** **MUST NOT** rely on the presence of a *SOAPAction* HTTP header to correctly process a message. **HTTP-TRANSPORTNOT_TESTABLE**

R2761 A **SENDER** **SHOULD NOT** include the *SOAPAction* HTTP header. **HTTP-TRANSPORTTESTABLE BP1761**

4.7.21 SOAP Binding Extensions

The `wsdl:required` attribute has been widely misunderstood and used by WSDL authors sometimes to incorrectly indicate the optionality of `wssoap12:headers`. The `wsdl:required` attribute, as specified in WSDL 1.1, is an extensibility mechanism aimed at WSDL processors. It allows new WSDL extension elements to be introduced in a graceful manner. The intent of `wsdl:required` is to signal to the WSDL processor whether the extension element needs to be recognized and understood by the WSDL processor in order that the WSDL description be correctly processed. It is not meant to signal conditionality or optionality of some construct that is included in the envelopes. For example, a `wsdl:required` attribute with the value "false" on a `wssoap12:header` element must not be interpreted to signal to the WSDL processor that the described SOAP header block is conditional or optional in the envelopes generated from the WSDL description. It is meant to be interpreted as "in order to send a envelope to the endpoint that includes in its description the `wssoap12:header` element, the WSDL processor **MUST** understand the semantic implied by the `wssoap12:header` element."

The default value for the `wsdl:required` attribute for WSDL 1.1 SOAP Binding extension elements is "false". Most WSDL descriptions in practice do not specify the `wsdl:required` attribute on the SOAP Binding extension elements, which could be interpreted by WSDL processors to mean that the extension elements may be ignored. The Profile requires that all WSDL 1.1 extensions be understood and processed by the consumer, irrespective of the presence or the value of the `wsdl:required` attribute on an extension element.

R2747 A **CONSUMER** *MUST* understand and process all WSDL 1.1 SOAP Binding extension elements, irrespective of the presence or absence of the `wsdl:required` attribute on an extension element; and irrespective of the value of the `wsdl:required` attribute, when present. **CORENOT_TESTABLE**

R2748 A **CONSUMER** *MUST NOT* interpret the presence of the `wsdl:required` attribute on a `wsoap12` extension element with a value of "false" to mean the extension element is optional in the envelopes generated from the WSDL description. **CORENOT_TESTABLE**

4.8 Use of @soapActionRequired in WSDL 1.1 SOAP 1.2 Binding

This section of the Profile incorporates the following specifications by reference:

- WSDL 1.1 Binding Extension for SOAP 1.2 [WSDL11SOAP12]

The WSDL 1.1 Binding Extension for SOAP 1.2 incorporates an attribute called `wsoap12:soapActionRequired` that has a default value of 'true'. This has the effect that, in order to have the action parameter omitted from the `Content-Type` HTTP header field, one would need to explicitly include the attribute in a description with a value of 'false'. The `action` parameter is defined by SOAP 1.2 to be OPTIONAL so as to enable use cases in which the sender of a SOAP message wishes to disallow casual inspection of the clear-text part of the message to infer its purpose or intent. Additionally, implementations are cautioned that they **SHOULD NOT** require the parameter's presence in a message in order to process the message.

The Profile WG believes that the `wsoap12:soapActionRequired` attribute runs counter to the SOAP 1.2 specification's intent. Therefore, the attribute has been profiled such that it **MUST NOT** be used, as it would tend to eliminate the very use cases that lead to the design of the SOAP 1.2 action parameter.

R2756 A **DESCRIPTION** *MUST NOT* include the `wsoap12:soapActionRequired` attribute. **HTTP-TRANSPORTTESTABLE**
BP2756

R2759 A **CONSUMER** *MUST* treat the absence of the `wsoap12:soapActionRequired` attribute as if it had a value of 'false'. **HTTP-TRANSPORTNOT_TESTED**

R2758 A **MESSAGE** *MAY* omit the `action` parameter from the `Content-Type` HTTP header field. **HTTP-TRANSPORTNOT_TESTED**

R2757 A **RECEIVER** *MUST NOT* fault due to the absence or presence of the `action` parameter on the HTTP `Content-Type` header field. **HTTP-TRANSPORTTESTABLE** **BP1757**

Although the WSDL 1.1 Binding Extension for SOAP 1.2 describes how the `action` parameter of the `Content-Type` header field is set by the `wsoap12:soapAction` attribute of the `wsoap12:operation` element, there are implementations that include the `action` parameter in request messages even in cases where the `wsoap12:soapAction` attribute does not appear in the `wsoap12:operation` that describes these requests. Note that, in some of these cases, the `action` parameter might have an empty string ("") value.

4.9 Use of XML Schema

This section of the Profile incorporates the following specifications by reference:

- XML Schema Part 1: Structures [xmlSchema-1]
- XML Schema Part 2: Datatypes [xmlSchema-2]

WSDL 1.1 uses XML Schema as one of its type systems. The Profile mandates the use of XML Schema as the type system for WSDL descriptions of Web Services.

R2800 A **DESCRIPTION** MAY contain any content validated against [xmlSchema-1]. CORENOT_TESTED

R2801 A **DESCRIPTION** MUST use [xmlSchema-2] as the basis of user defined datatypes and structures. CORETESTABLE BP2122

4.10 4.10 WS-Addressing 1.0 - Metadata

WS-Addressing message addressing properties (MAPs) can be specified in a WSDL document. The Profile adds restrictions to such properties specified in WSDL.

The `wsam:Action` attribute is used in WSDL to specify the value of the `wsa:Action` SOAP header in the envelope. A default value computation algorithm is specified for the case where an explicit `wsam:Action` attribute is not specified.

R2900 The value of the `wsa:Action` header block in an **ENVELOPE** MUST equal the value (either actual or computed) of the `wsam:Action` attribute for the corresponding WSDL element (`wsdl:input`, `wsdl:output`, or `wsdl:fault`) contained in the target `wsdl:operation` of the `wsdl:portType`. CORETESTABLE BP1142a BP1142b BP1142c BP1143a BP1143b BP1143c BP1090a BP1090b

R2901 In a **DESCRIPTION**, the actual value of the `wsam:Action` attribute for the `wsdl:input` element contained in the target `wsdl:operation` of the `wsdl:portType` MUST be equal to the value of the `soapAction` attribute, if present, of the `wsoap12:operation` element contained in the target `wsdl:operation` of the `wsdl:binding`. CORETESTABLE BP2801

Note that this requirement is closely related to R1144, which defines the relationship between the value of the `wsa:Action` SOAP header and the value of the `action` parameter of the HTTP Content-Type MIME header.

5 WSDL Corrections

The following sections contain requirements that correct errors and inconsistencies in the Web Services Description Language (WSDL) 1.1 [WSDL1.1]. These have been collected into this common section to serve as a reference point for other specifications.

This section of the Profile incorporates the following specifications by reference:

- Web Services Description Language (WSDL) 1.1 [WSDL1.1]
- XML Schema Part 1: Structures [xmlSchema-1]
- XML Schema Part 2: Datatypes [xmlSchema-2]

These extensibility points are listed, along with any extensibility points from other sections of this Profile, in Appendix A.

5.1 Document Structure

Section 2.1 of [WSDL1.1] [WSDL Document Structure](#) defines the overall structure of an XML document for describing Web services. This Profile mandates the use of that structure, with the following corrections:

5.1.1 WSDL Schema Definitions

The normative schemas for WSDL appearing in Appendix 4 of the WSDL 1.1 specification have inconsistencies with the normative text of the specification. The Profile references new schema documents that have incorporated fixes for known errors.

R2028 A **DESCRIPTION** *using the WSDL namespace (prefixed "wsdl" in this Profile) MUST be valid according to the XML Schema found at "<http://ws-i.org/profiles/basic/1.1/wsdl-2004-08-24.xsd>".* CORE TESTABLE BP2705

R2029 A **DESCRIPTION** *using the WSDL SOAP binding namespace (prefixed "wsoap12" in this Profile) MUST be valid according to the XML Schema found at "<http://schemas.xmlsoap.org/wsdl/soap12/wsdl11soap12-20060302.xsd>".* CORE TESTABLE BP2704

Although the Profile requires WSDL descriptions to be Schema valid, it does not require consumers to validate WSDL documents. It is the responsibility of a WSDL document's author to assure that it is Schema valid.

5.1.2 WSDL and Schema Import

Some examples in WSDL 1.1 incorrectly show the WSDL import statement being used to import XML Schema definitions. The Profile clarifies use of the import mechanisms to keep them consistent and confined to their respective domains. Imported schema documents are also constrained by XML version and encoding requirements consistent to those of the importing WSDL documents.

R2001 A **DESCRIPTION** *MUST only use the WSDL "import" statement to import another WSDL description.* CORE TESTABLE BP2101

R2803 In a **DESCRIPTION**, the namespace attribute of the `wsdl:import` **MUST NOT** be a relative URI. CORETESTABLE BP2803

R2002 To import XML Schema Definitions, a **DESCRIPTION** **MUST** use the XML Schema "import" statement. CORETESTABLE BP2101

R2003 A **DESCRIPTION** **MUST** use the XML Schema "import" statement only within the `xsd:schema` element of the types section. CORETESTABLE BP2103

R2004 In a **DESCRIPTION** the `schemaLocation` attribute of an `xsd:import` element **MUST NOT** resolve to any document whose root element is not "schema" from the namespace "http://www.w3.org/2001/XMLSchema". CORETESTABLE BP2106

R2009 An XML Schema directly or indirectly imported by a **DESCRIPTION** **MAY** include the Unicode Byte Order Mark (BOM). CORENOT_TESTABLE

R2010 An XML Schema directly or indirectly imported by a **DESCRIPTION** **MUST** use either UTF-8 or UTF-16 encoding. CORETESTABLE BP2202

R2011 An XML Schema directly or indirectly imported by a **DESCRIPTION** **MUST** use version 1.0 of the eXtensible Markup Language W3C Recommendation. CORENOT_TESTED

For example,

INCORRECT:

```
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote/definitions"
  xmlns:sq="http://example.com/stockquote"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://example.com/stockquote"
    location="http://example.com/stockquote/schemas/stockquote.xsd"/>

  <message name="GetLastTradePriceInput">
    <part name="body" element="sq:TradePriceRequest"/>
  </message>
  ...
</definitions>
```

CORRECT:

```
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote/definitions"
  xmlns:sq="http://example.com/stockquote"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://example.com/stockquote/definitions"
    location="http://example.com/stockquote/definitions/stockquote.wsdl"/>

  <message name="GetLastTradePriceInput">
```

```

    <part name="body" element="sq:TradePriceRequest"/>
  </message>
  ...
</definitions>
CORRECT:

<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote/definitions"
  xmlns:sq="http://example.com/stockquote"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <xsd:schema targetNamespace="http://example.com/stockquote">
      <xsd:import namespace="http://example.com/stockquote"

schemaLocation="http://example.com/stockquote/schemas/stockquote.xsd"/>
      <xsd:element name="TradePriceRequest" type="sq:PriceRequestType"/>
    </xsd:schema>
    </types>

    <message name="GetLastTradePriceInput">
      <part name="body" element="sq:TradePriceRequest"/>
    </message>
    ...
  </definitions>

```

5.1.3 Placement of WSDL import Elements

Example 3 in WSDL 1.1 Section 3.1 causes confusion regarding the placement of `wsdl:import`.

R2022 When they appear in a **DESCRIPTION**, `wsdl:import` elements **MUST** precede all other elements from the WSDL namespace **except** `wsdl:documentation`. **CORE TESTABLE BP2105**

R2023 When they appear in a **DESCRIPTION**, `wsdl:types` elements **MUST** precede all other elements from the WSDL namespace **except** `wsdl:documentation` **and** `wsdl:import`. **CORE TESTABLE BP2018**

For example,

INCORRECT:

```

<definitions name="StockQuote"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="GetLastTradePriceInput">
    <part name="body" element="sq:TradePriceRequest"/>
  </message>
  ...
  <import namespace="http://example.com/stockquote/definitions"

location="http://example.com/stockquote/definitions/stockquote.wsdl"/>
  ...
  <service name="StockQuoteService">
    <port name="StockQuotePort" binding="tns:StockQuoteSoap">

```



```

    ...
  </port>
</service>
</definitions>

```

CORRECT:

```

<definitions name="StockQuote"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://example.com/stockquote/definitions"
location="http://example.com/stockquote/definitions/stockquote.wsdl"/>

  <message name="GetLastTradePriceInput">
    <part name="body" element="sq:TradePriceRequest"/>
  </message>
  ...
  <service name="StockQuoteService">
    <port name="StockQuotePort" binding="tns:StockQuoteSoap">
      ...
    </port>
  </service>
</definitions>

```

INCORRECT:

```

<definitions name="StockQuote"
  ...
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="GetLastTradePriceInput">
    <part name="body" element="sq:TradePriceRequest"/>
  </message>
  ...
  <service name="StockQuoteService">
    <port name="StockQuotePort" binding="tns:StockQuoteSoap">
      ...
    </port>
  </service>

  <types>
    <xsd:schema targetNamespace="http://example.com/stockquote">
      ...
    </xsd:schema>
  </types>
</definitions>

```

CORRECT:

```

<definitions name="StockQuote"
  ...
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://example.com/stockquote/definitions"
location="http://example.com/stockquote/definitions/stockquote.wsdl"/>

```



```

<types>
  <xsd:schema targetNamespace="http://example.com/stockquote">
    ...
  </xsd:schema>
</types>

<message name="GetLastTradePriceInput">
  <part name="body" element="sq:TradePriceRequest"/>
</message>
...
<service name="StockQuoteService">
  <port name="StockQuotePort" binding="tns:StockQuoteSoap">
    ...
  </port>
</service>
</definitions>

```

5.1.4 WSDL documentation Element

The WSDL 1.1 schema and the WSDL 1.1 specification are inconsistent with respect to where `wsdl:documentation` elements may be placed.

R2030 *In a **DESCRIPTION** the `wsdl:documentation` element MAY be present as the first child element of `wsdl:import`, `wsdl:part` and `wsdl:definitions` in addition to the elements cited in the WSDL 1.1 specification.* CORENOT_TESTED

5.2 Message

Section 2.3 of [WSDL1.1] [Messages](#) defines the `wsdl:message` elements that are used to represent abstract definitions of the data being transmitted. This Profile defines the following corrections on `wsdl:message` elements.

5.2.1 Declaration of part Elements

Examples 4 and 5 in WSDL 1.1 Section 3.1 incorrectly show the use of XML Schema types (e.g. "xsd:string") as a valid value for the `element` attribute of a `wsdl:part` element.

R2206 *A `wsdl:message` in a **DESCRIPTION** containing a `wsdl:part` that uses the `element` attribute MUST refer, in that attribute, to a global element declaration.* CORETESTABLE BP2115

For example,

INCORRECT:

```

<message name="GetTradePriceInput">
  <part name="tickerSymbol" element="xsd:string"/>
</message>

```

CORRECT:

```

<message name="GetTradePriceInput">
  <part name="body" element="tns:SubscribeToQuotes"/>
</message>

```

5.3 SOAP Binding

The WSDL 1.1 Binding Extension for SOAP 1.2 [WSDL11Soap12] defines a binding for SOAP 1.2 endpoints. This Profile mandates the use of the SOAP 1.2 binding as defined in WSDL 1.1 Binding Extension for SOAP 1.2, with the following corrections:

5.3.1 Specifying the transport Attribute

There is an inconsistency between the WSDL 1.1 specification and the WSDL 1.1 schema regarding the `transport` attribute. The WSDL 1.1 specification requires it; however, the schema shows it to be optional.

R2701 The *wSDL:binding* element in a **DESCRIPTION** **MUST** be constructed so that its *wsoap12:binding* child element specifies the *transport* attribute. CORETESTABLE BP2403

5.3.2 SOAP 1.2 Binding Extension

R2179 In a **DESCRIPTION**, a *wsoap12:fault* **MUST** specify the contents of the SOAP 1.2 fault *Detail* element. CORENOT_TESTABLE

The *wsoap12:fault* element is meant to describe the SOAP 1.2 fault detail entries. WSDL 1.1 Binding Extension for SOAP 1.2 incorrectly says that it specifies the contents of the *Reason* element.

5.3.3 Type and Name of SOAP Binding Elements

The WSDL 1.1 schema disagrees with the WSDL 1.1 specification about the name and type of an attribute of the *wsoap12:header* and *wsoap12:headerfault* elements.

R2720 A *wSDL:binding* in a **DESCRIPTION** **MUST** use the *part* attribute with a schema type of "NMTOKEN" on all contained *wsoap12:header* and *wsoap12:headerfault* elements. CORETESTABLE BP2021

R2749 A *wSDL:binding* in a **DESCRIPTION** **MUST NOT** use the *parts* attribute on contained *wsoap12:header* and *wsoap12:headerfault* elements. CORETESTABLE BP2021

The WSDL Schema gives the attribute's name as "parts" and its type as "NMTOKENS". The schema is incorrect since each *wsoap12:header* and *wsoap12:headerfault* element references a single *wSDL:part*.

For example,

CORRECT:

```
<binding name="StockQuoteSoap" type="tns:StockQuotePortType">
  <wsoap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="SubscribeToQuotes">
    <input>
      <wsoap12:body parts="body" use="literal"/>
      <wsoap12:header message="tns:SubscribeToQuotes" part="subscribeheader"
use="literal"/>
    </input>
  </operation>
</binding>
```

5.3.4 name Attribute on Faults

There is inconsistency between the WSDL 1.1 specification and the WSDL 1.1 schema, which does not list the `name` attribute.

R2721 A `wsdl:binding` in a **DESCRIPTION** **MUST** have the `name` attribute specified on all contained `wsoap12:fault` elements. CORETESTABLE BP2022

R2754 In a **DESCRIPTION**, the value of the `name` attribute on a `wsoap12:fault` element **MUST** match the value of the `name` attribute on its parent `wsdl:fault` element. CORETESTABLE BP2032

5.3.5 Omission of the use Attribute

There is inconsistency between the WSDL 1.1 specification and the WSDL 1.1 schema regarding the `use` attribute.

R2722 A `wsdl:binding` in a **DESCRIPTION** **MAY** specify the `use` attribute on contained `wsoap12:fault` elements. CORENOT_TESTED

R2723 If in a `wsdl:binding` in a **DESCRIPTION** the `use` attribute on a contained `wsoap12:fault` element is present, its value **MUST** be "literal". CORETESTABLE BP2406

[WSDL1.1] Section 3.6 indicates that the `use` attribute of `wsoap12:fault` is required while in the schema the `use` attribute is defined as optional. The Profile defines it as optional, to be consistent with `wsoap12:body`.

Since the `use` attribute is optional, the Profile identifies the default value for the attribute when omitted.

Finally, to assure that the Profile is self-consistent, the only permitted value for the `use` attribute is "literal".

5.3.6 Default for use Attribute

There is an inconsistency between the WSDL 1.1 specification and the WSDL 1.1 schema regarding whether the `use` attribute is optional on `wsoap12:body`, `wsoap12:header`, and `wsoap12:headerfault`, and if so, what omitting the attribute means.

R2707 A `wsdl:binding` in a **DESCRIPTION** that contains one or more `wsoap12:body`, `wsoap12:fault`, `wsoap12:header` or `wsoap12:headerfault` elements that do not specify the `use` attribute **MUST** be interpreted as though the value "literal" had been specified in each case. CORENOT_TESTABLE

6 Service Publication and Discovery

When publication or discovery of Web services is required, UDDI is the mechanism the Profile has adopted to describe Web service providers and the Web services they provide. Business, intended use, and Web service type descriptions are made in UDDI terms; detailed technical descriptions are made in WSDL terms. Where the two specifications define overlapping descriptive data and both forms of description are used, the Profile specifies that the descriptions must not conflict.

Registration of Web service instances in UDDI registries is optional. By no means do all usage scenarios require the kind of metadata and discovery UDDI provides, but where such capability is needed, UDDI is the sanctioned mechanism.

Note that the Web services that constitute UDDI V2 are not fully conformant with the Profile 1.0 because they do not accept messages whose envelopes are encoded in either UTF-8 and UTF-16 as required by the Profile. (They accept UTF-8 only.) That there should be such a discrepancy is hardly surprising given that UDDI V2 was designed and, in many cases, implemented before the Profile was developed. UDDI's designers are aware of UDDI V2's nonconformance and will take it into consideration in their future work.

This section of the Profile incorporates the following specifications by reference:

- UDDI Version 2.04 API Specification, Dated 19 July 2002 [UDDI2.04API]
- UDDI Version 2.03 Data Structure Reference, Dated 19 July 2002 [UDDI2.03Data]
- UDDI Version 2 XML Schema [UDDI2schema]

These extensibility points are listed, along with any extensibility points from other sections of this Profile, in Appendix A.

6.1 bindingTemplates

This section of the Profile incorporates the following specifications by reference:

- Section 7 of [UDDI2.03Data] [The bindingTemplate structure](#)

UDDI represents Web service instances as `uddi:bindingTemplate` elements. The `uddi:bindingTemplate` plays a role that is the rough analog of the `wsdl:port`, but provides options that are not expressible in WSDL. To keep the WSDL description of an instance and its UDDI description consistent, the Profile places the following constraints on how `uddi:bindingTemplate` elements may be constructed.

WSDL's `wsoap12:address` element requires the network address of the instance to be directly specified. In contrast, UDDI V2 provides two alternatives for specifying the network address of instances it represents. One, the `uddi:accessPoint`, mirrors the WSDL mechanism by directly specifying the address. The other, the `uddi:hostingRedirector`, provides a Web service-based indirection mechanism for resolving the address, and is inconsistent with the WSDL mechanism.

R3100 REGDATA of type `uddi:bindingTemplate` representing a conformant INSTANCE MUST contain the `uddi:accessPoint` element. CORENOT_TESTED

For example,

INCORRECT:

```
<bindingTemplate bindingKey="...">
  <description xml:lang="EN">BarSOAPPort</description>
  <hostingRedirector bindingKey="..." />
  <tModelInstanceDetails>
    ...
  </tModelInstanceDetails>
</bindingTemplate>
```

CORRECT:

```
<bindingTemplate bindingKey="...">
  <description xml:lang="EN">BarSOAPPort</description>
  <accessPoint>http://example.org/myBarSOAPPort</accessPoint>
  <tModelInstanceDetails>
    ...
  </tModelInstanceDetails>
</bindingTemplate>
```

6.2 tModels

This section of the Profile incorporates the following specifications by reference:

- Section 8 of [UDDI2.03Data] [The tModel structure](#)

UDDI represents Web service types as `uddi:tModel` elements. (See Section 8.1.1 of [UDDI2.03Data] [Defining the technical fingerprint](#)). These may, but need not, point (using a URI) to the document that contains the actual description. Further, UDDI is agnostic with respect to the mechanisms used to describe Web service types. The Profile cannot be agnostic about this because interoperability is very much complicated if Web service types do not have descriptions or if the descriptions can take arbitrary forms.

The appendix I.1.2.1.1 of [UDDI2.04API] [Taxonomy Values](#) allows but does not require `uddi:tModel` elements that use WSDL to describe the Web service type they represent to state that they use WSDL as the description language. Not doing so leads to interoperability problems because it is then ambiguous what description language is being used.

Therefore the Profile places the following constraints on how `uddi:tModel` elements that describe Web service types may be constructed:

The Profile chooses WSDL as the description language because it is by far the most widely used such language.

R3002 REGDATA of type `uddi:tModel` representing a conformant Web service type **MUST** use WSDL as the description language.

CORENOT_TESTED

To specify that conformant Web service types use WSDL, the Profile adopts the UDDI categorization for making this assertion.

R3003 REGDATA of type `uddi:tModel` representing a conformant Web service type **MUST** be categorized using the `uddi:types` taxonomy and a categorization of "wsdlSpec". CORENOT_TESTED

For the `uddi:overviewURL` in a `uddi:tModel` to resolve to a `wsdl:binding`, the Profile must adopt a convention for distinguishing among multiple `wsdl:bindings` in a WSDL document. The UDDI Best Practice for Using WSDL in a UDDI Registry specifies the most widely recognized such convention.

R3010 **REGDATA** of type `uddi:tModel` representing a conformant Web service type **MUST** follow *V1.08 of the UDDI Best Practice for Using WSDL in a UDDI Registry*. **CORENOT_TESTED**

It would be inconsistent if the `wsdl:binding` that is referenced by the `uddi:tModel` does not conform to the Profile.

R3011 The `wsdl:binding` that is referenced by **REGDATA** of type `uddi:tModel` **MUST** itself conform to the Profile. **CORENOT_TESTED**

7 Security

As is true of all network-oriented information technologies, the subject of security is a crucial one for Web services. For Web services, as for other information technologies, security consists of understanding the potential threats an attacker may mount and applying operational, physical, and technological countermeasures to reduce the risk of a successful attack to an acceptable level. Because an "acceptable level of risk" varies hugely depending on the application, and because costs of implementing countermeasures is also highly variable, there can be no universal "right answer" for securing Web services. Choosing the absolutely correct balance of countermeasures and acceptable risk can only be done on a case by case basis.

While Basic Profile conformance is important for the web services community to ensure interoperability, an instance is expected take whatever security countermeasures it deems necessary to protect itself; even if in that specific case it is acting outside of and is not in conformance with this Profile.

There are common patterns of countermeasures that experience shows reduce the risks to acceptable levels for many Web services. The Profile adopts, but does not mandate use of, the most widely used of these: HTTP secured with either TLS 1.0 or SSL 3.0 (HTTPS). That is, conformant Web services may use HTTPS; they may also use other countermeasure technologies or none at all.

HTTPS is widely regarded as a mature standard for encrypted transport connections to provide a basic level of confidentiality. HTTPS thus forms the first and simplest means of achieving some basic security features that are required by many real-world Web service applications. HTTPS may also be used to provide client authentication through the use of client-side certificates.

See conformance criteria in [Section 2](#) when the HTTP(S) transport protocol is used.

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- RFC2818: HTTP Over TLS [RFC2818]
- RFC2246: The TLS Protocol Version 1.0 [RFC2246]
Extensibility points:
 - E0019 - TLS Cyphersuite - TLS allows for the use of arbitrary encryption algorithms.[HTTP-TRANSPORT](#)
 - E0020 - TLS Extensions - TLS allows for extensions during the handshake phase.[HTTP-TRANSPORT](#)
- The SSL Protocol Version 3.0 [SSLV3]
Extensibility points:
 - E0021 - SSL Cyphersuite - SSL allows for the use of arbitrary encryption algorithms.[HTTP-TRANSPORT](#)
- RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile [RFC2459]
Extensibility points:
 - E0022 - Certificate Authority - The choice of the Certificate Authority is a private agreement between parties.[HTTP-TRANSPORT](#)
 - E0023 - Certificate Extensions - X509 allows for arbitrary certificate extensions.[HTTP-TRANSPORT](#)

These extensibility points are listed, along with any extensibility points from other sections of this Profile, in Appendix A.

7.1 Use of HTTPS

HTTPS is such a useful, widely understood basic security mechanism that the Profile needs to allow it.

R5000 An **INSTANCE** MAY require the use of HTTPS. HTTP-
TRANSPORTNOT_TESTED

R5001 If an **INSTANCE** requires the use of HTTPS, the location attribute of the *wsoap12:address* element in its *wsdl:port* description MUST be a URI whose scheme is "https"; otherwise it MUST be a URI whose scheme is "http". HTTP-
TRANSPORTNOT_TESTED

Simple HTTPS provides authentication of the Web service instance by the consumer but not authentication of the consumer by the instance. For many instances this leaves the risk too high to permit interoperation. Including the mutual authentication facility of HTTPS in the Profile permits instances to use the countermeasure of authenticating the consumer. In cases in which authentication of the instance by the consumer is insufficient, this often reduces the risk sufficiently to permit interoperation.

R5010 An **INSTANCE** MAY require the use of HTTPS with mutual authentication. HTTP-TRANSPORTNOT_TESTED

Appendix A. Extensibility Points

This section identifies extensibility points, as defined in "Scope of the Profile," for the Profile's component specifications.

Agreements on these extensibility points are out of the scope of the Profile and Profile conformance. An initial, non-exhaustive list of these extensibility points is provided here as their use may affect interoperability. In order to avoid interoperability issues not addressed by the Profile, out-of-band agreement on the use of these extensibility points may be necessary between the parties to a Web service.

In SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [SOAP12-1]:

- **E0001 - Header blocks** - Header blocks are an extensibility mechanism in SOAP. CORETESTABLE BP1901
- **E0002 - Processing order** - The order of processing of a SOAP envelope's components (e.g., headers) is unspecified, and therefore may need to be negotiated out-of-band. CORENOT_TESTABLE
- **E0003 - Use of intermediaries** - SOAP Intermediaries is an underspecified mechanism in SOAP 1.2, and their use may require out-of-band negotiation. Their use may also necessitate careful consideration of where Profile conformance is measured. CORENOT_TESTABLE
- **E0004 - SOAP role values** - Values of the SOAP role attribute, other than the special uri 'http://www.w3.org/2003/05/soap-envelope/role/next', represent a private agreement between parties of the web service. CORETESTABLE BP1904
- **E0005 - Fault details** - Faults may have Detail elements. The contents of these elements are not described in SOAP 1.2. CORETESTABLE BP1905
- **E0024 - Namespace Attributes** - Namespace attributes on soap12:Envelope and soap12:Header elements CORETESTABLE
- **E0025 - Attributes on soap12:Body elements** - Neither namespaced nor local attributes are constrained by SOAP 1.2. CORETESTABLE

In SOAP Version 1.2 Part 2: Adjuncts (Second Edition) [SOAP12-2]:

- **E0026 - SOAP envelope in HTTP Response message to WSDL one-way operation** - The SOAP 1.2 Request Response Binding specification does not specify the purpose or processing of such envelopes. HTTP-TRANSPORTTESTABLE

In RFC2616: Hypertext Transfer Protocol -- HTTP/1.1 [RFC2616]:

- **E0007 - HTTP Authentication** - HTTP authentication allows for extension schemes, arbitrary digest hash algorithms and parameters. HTTP-TRANSPORTTESTABLE
- **E0008 - Unspecified Header Fields** - HTTP allows arbitrary headers to occur in messages. HTTP-TRANSPORTTESTABLE
- **E0010 - Content-Encoding** - The set of content-codings allowed by HTTP is open-ended and any besides 'gzip', 'compress', or 'deflate' are an extensibility point. HTTP-TRANSPORTTESTABLE
- **E0011 - Transfer-Encoding** - The set of transfer-encodings allowed by HTTP is open-ended. HTTP-TRANSPORTTESTABLE
- **E0029 - Use of messages other than SOAP 1.2 or XOP messages** - Use of Messages other than a SIMPLE_SOAP_MESSAGE or a XOP_ENCODED_MESSAGE is an extensibility point CORETESTABLE

In WS-Addressing 1.0 - SOAP Binding [WSAddrSoap] (except for sections 4, 5.1.1, 5.2.1 and 6.2):

- **E0027 - Use of SOAP role attribute and WS-Addressing** - WS-Addressing allows multiple instances of headers such as wsa:To, wsa:ReplyTo, and wsa:FaultTo, so long as they are targeted to different SOAP roles. **CORE TESTABLE**
- **E0028 - Endpoint references are extensible** - When extension attributes or elements appear as part of an endpoint reference, the processing model for such extensions is defined by the specification for those extensions. **CORE NOT TESTABLE**

In XML Schema Part 1: Structures [xmlSchema-1]:

- **E0017 - Schema annotations** - XML Schema allows for annotations, which may be used to convey additional information about data structures. **CORE**

In Web Services Description Language (WSDL) 1.1 [WSDL1.1]:

- **E0013 - WSDL extensions** - WSDL allows extension elements and attributes in certain places, including the use and specification of alternate protocol binding extensions; use of such extensions requires out-of-band negotiation. **CORE**
- **E0014 - Validation mode** - whether the parser used to read WSDL and XML Schema documents performs DTD validation or not. **CORE**
- **E0015 - Fetching of external resources** - whether the parser used to read WSDL and XML Schema documents fetches external entities and DTDs. **CORE**
- **E0016 - Relative URIs** - WSDL does not adequately specify the use of relative URIs for the following: wsoap12:body/@namespace, wsoap12:address/@location, wsdl:import/@location, xsd:schema/@targetNamespace and xsd:import/@schemaLocation. Their use may require further coordination; see XML Base for more information. **CORE**

In RFC2246: The TLS Protocol Version 1.0 [RFC2246]:

- **E0019 - TLS Cyphersuite** - TLS allows for the use of arbitrary encryption algorithms. **HTTP-TRANSPORT**
- **E0020 - TLS Extensions** - TLS allows for extensions during the handshake phase. **HTTP-TRANSPORT**

In The SSL Protocol Version 3.0 [SSLV3]:

- **E0021 - SSL Cyphersuite** - SSL allows for the use of arbitrary encryption algorithms. **HTTP-TRANSPORT**

In RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile [RFC2459]:

- **E0022 - Certificate Authority** - The choice of the Certificate Authority is a private agreement between parties. **HTTP-TRANSPORT**
- **E0023 - Certificate Extensions** - X509 allows for arbitrary certificate extensions. **HTTP-TRANSPORT**

Appendix B. Schemas

A copy of the XML Schema (non-normative) for WS-Policy conformance claims is listed below for convenience:

```
<xs:schema targetNamespace='http://ws-i.org/profiles/basic-profile/2.0/'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  elementFormDefault='qualified'
  blockDefault='#all'>
  <xs:element name='Conformant'>
    <xs:complexType>
      <xs:sequence>
        <xs:any namespace='##other' processContents='lax' minOccurs='0'
maxOccurs='unbounded' />
      </xs:sequence>
      <xs:anyAttribute namespace='##other' processContents='lax' />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Appendix C. Testing

C.1 Testability of Requirements

The testability of each Requirement is indicated as informational. It is a non-normative complement to this profile. It is represented by the following tags:

- **TESTABLE:** This means that the requirement could be tested, and that some test assertion(s) has been written for it.
- **TESTABLE_SCENARIO_DEPENDENT:** This means that a specific test scenario is needed in order to exercise the related test assertion, because the test assertion is designed to trigger only on specific input material. Producing this input material requires executing a scenario with specific data that is very unlikely to be produced by systems in production under normal operating conditions (e.g. material known to NEVER be recognizable by an endpoint.)
- **NOT_TESTED:** This is the case for most optional requirements (SHOULD, MAY), and for most Extensibility points as well as for requirements targeting UDDI. Some requirements may also require Schema awareness (ability to process schemas) from the Analyzer test tool. As this conflicted with the ability to use several freely available XSLT20 processors that are not Schema aware, such requirements have been marked "NOT_TESTED" unless this verification was done by tools prior to creating the test log file, which would then just contain some metadata indicating the results of these schema-related tests. A subsequent version may cover untested requirements. In this profile, the core requirements for assessing interoperability of implementations have been initially targeted
- **NOT_TESTABLE:** This means that these requirements cannot be tested based on the technology choices (black-box testing, XPath scripting)

C.2 Structure of Test Assertions

The test assertions are structured in XML, with some elements scripted using XPath2.0 and are automatically processable using the version 2.0 of the WS-I Analyzer tools.

Test Assertion Part	What it means:
Test Assertion ID (required) <i>[markup: testAssertion/@id]</i>	A unique ID for the current test assertion.
Description (optional) <i>[markup: testAssertion/description]</i>	A plain text description of the current test assertion. At minimum expressing the TA predicate.
Comments (optional) <i>[markup: testAssertion/comments]</i>	A plain text comment about the TA script and how well it covers the profile requirement. Explanation material for users, and developers (what could be improved, etc.).
Target (required) <i>[markup: testAssertion/target]</i>	The artifacts to be tested, defined by an XPath expression that returns a list of XML nodes from the log file in input. For every artifact (node) selected by the Target expression, there will be a report entry for this TA in the test report, with a result of either: <ul style="list-style-type: none">• passed

	<ul style="list-style-type: none"> • failed • warning • notApplicable • notRelevant • missingInput • undetermined <p>See the "reporting" item for the meaning of these results.</p>
Cotarget (optional) [markup: testAssertion/cotarget]	<p>Artifact that is related to the target, and that needs be accessed for the testing. Identified by an XPath expression that may refer to the related target node using the variable '\$target'.</p> <p>For example, the target can be a SOAP message and the cotarget the WSDL file that describes this SOAP message.</p> <p>A cotarget must have a @name attribute that identifies it. The value of this attribute can be used as a variable (when prepending '\$' to it) by subsequently defined cotargets, prerequisite and predicate.</p>
Prerequisite (optional) [markup: testAssertion/@preReq] (optional) [markup: testAssertion/prerequisite] (optional)	<p>The pre-condition for evaluating this Test Assertion on this target. If the prerequisite evaluates to "false" then the target does not qualify for this Test Assertion (the test report is "notRelevant")</p> <p>The first part (preReq attribute) is an enumeration of Test Assertion IDs. Each one of the prerequisite TAs must either use the same target (e.g. SOAP Envelope, or WSDL binding, etc.) as this TA, or a target that is of a more general type than the main TA target. The target must "pass" each one of these prerequisite TAs in order to qualify for this TA.</p> <p>(e.g. the target of TA t1 can be a WSDL binding while the target of a TA t2 prerequisite of t1, can be the entire WSDL file).</p> <p>The second part ("prerequisite" element) is an XPath (boolean) expression of the same nature as the predicate. If present, it must evaluate to "true" for the target to qualify. If it fails, the result for the current TA in the report will be "notRelevant". Otherwise, the target can be further evaluated by the predicate of the main TA. The expression may refer to the target explicitly using the variable name "\$target", or to any cotarget using its name as variable name (\$[name]).</p>
Predicate (required) [markup: testAssertion/predicate]	<p>A logical expression that evaluates whether this target is fulfilling the profile requirement addressed by this test Assertion. By default:</p> <ul style="list-style-type: none"> - A result of true means the requirement is fulfilled (reported as a "passed" in the test report). - A result of false means the requirement is violated (reported as a "failed" in the test report). <p>However, in some cases and for testability reasons, the predicate may be designed as a partial indicator e.g. only</p>

	<p>indicates some cases of fulfillment, or some cases of violation. As a result, when "true" indicates fulfillment it may be that "false" is inconclusive, or conversely "false" will indicate violation, but "true" is inconclusive. In such cases, the "Reporting" element specifies the meaning of the predicate result w/r to the profile requirement.</p> <p>The predicate expression implicitly refers to the target (which is its "XPath context") although it may explicitly refer to it using the variable name "\$target". It may refer to any cotarget using its name as variable name (\$[name]).</p>
<p>Prescription (required)</p> <p><i>[markup: testAssertion/prescription/@level]</i></p>	<p>Conveys the level of prescription associated with the profile requirement. At least three values may be used:</p> <ul style="list-style-type: none"> • mandatory: maps to RFC2119 keywords MUST, MUST NOT, SHALL, SHALL NOT, REQUIRED (and sometimes MAY NOT) • preferred: maps to RFC2119 keywords SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED • permitted: maps to RFC2119 keywords MAY, OPTIONAL.
<p>Reporting (optional)</p> <p><i>[markup: testAssertion/reporting]</i></p>	<p>For each possible outcome of the predicate (true or false), specifies how it must be interpreted w/r to the profile feature. Two attributes are used that both must be present, when this element is present:</p> <ul style="list-style-type: none"> • @true attribute: may take values among {passed, failed, warning, undetermined} (default is 'passed') • @false attribute: may take values among {passed, failed, warning, undetermined} (default is 'failed') <p>The reported outcomes have the following meaning:</p> <ul style="list-style-type: none"> • passed: the target passes the test and can be considered as fulfilling the profile feature. • failed: the target fails the test and can be considered as violating (or not exhibiting) the profile feature. • warning: the test result is inconclusive. There is a possibility of profile requirement violation, that deserves further investigation. • undetermined: the test result is inconclusive for this predicate value. <p>NOTES: the predicate of the TA may be worded in a negative way so that @false='passed' although that is not recommended. The result of a test should not be related to the prescription level, e.g. a "preferred" or "permitted" level should not imply that @false='warning'.</p> <p>Other test results that are automatically generated and not</p>

controlled by the "reporting" element are:

- **notRelevant**: the target failed the prerequisite condition and therefore does not qualify for further testing (i.e. the predicate expression is NOT evaluated on it).
- **missingInput**: a cotarget expression returned an empty node set.
- **notApplicable**: this target was not even selected by the target XPath expression, while being of the same general artifact type (e.g. message type).

C.3 Test Log Conventions

The test assertions designed for this test suite are written to work over "test log" files that are assumed to follow some rules in their structure and content. These rules are more completely stated in the documentation associated with the log file description. Some of these rules are:

- Every message in the log must be uniquely identified: it must have a unique pair of values for: {message/@conversation, message/@id}, where @id is unique within each conversation. Typically, a conversation is used to identify an HTTP connection and the group of messages over this connection.
- A response message (for WSDL request-responses as well as RM lifecycle messages) always appears after the request message in the log file.
- A wsa:RelatesTo reference always refers to a message that has been logged before.
- A Fault message always appears after the message-in-error.
- An RM acknowledgement always appears after the messages it acknowledges.
- There should not be both a doc-lit and an rpc-lit bindings for the same portType.
- Imports must be resolved locally to the log file.

Appendix D. Test Assertions

This is a non-normative complement to this profile

Test Assertion:	BP1901
Description:	The soap:Envelope contains other header blocks.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope[soap12:Header[some \$hblock in child::* satisfies not(fn:namespace-uri(\$hblock) = 'http://www.w3.org/2005/08/addressing') and not(\$hblock/attribute::*:IsReferenceParameter)]]
Predicate:	fn:true()
Reporting:	true=warning, false=failed
Prescription:	permitted
Error Message:	Extensibility Point WARNING: The soap12:Envelope makes use of extra SOAP headers
Diagnostic Data:	

Test Assertion:	BP1904
Description:	The soap12:Envelope contains @role with unspecified values.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope[.//attribute::*:role]
Predicate:	every \$rval in .//*/attribute::*:role satisfies (\$rval = 'http://www.w3.org/2003/05/soap-envelope/role/next' or \$rval = 'http://www.w3.org/2003/05/soap-envelope/role/none')
Reporting:	true=passed, false=warning
Prescription:	permitted
Error Message:	Extensibility Point WARNING: The soap12:Envelope contains @role with unspecified values
Diagnostic Data:	

Test Assertion:	BP1905
Description:	The soap12:Envelope contains a Fault with unspecified Detail content.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope[soap12:Body/soap12:Fault/soap12:Detail]
Predicate:	fn:true()
Reporting:	true=warning, false=passed
Prescription:	permitted
Error Message:	Extensibility Point WARNING: The soap12:Fault makes use of optional Detail element
Diagnostic Data:	

Test Assertion:	BP1015
Description:	A receiver must not fault messages with envelopes that contain an XML Declaration.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents[@containsXmlDecl = 'true']/soap12:Envelope
Predicate:	not (/wsil:testLog/wsil:messageLog/wsil:message[(@type = 'response' and @conversation = \$target/../../@conversation) or (.//soap12:Envelope/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] = \$target/soap12:Header/wsa:MessageID)]/wsil:messageContents/soap12:Envelope/soap12:Body/soap12:Fault)

Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	A soap Fault has been generated when receiving a message with an XML declaration: verify that this Fault has another cause - it must not be caused by the XML declaration.
Diagnostic Data:	{SOAP message}

Test Assertion:	BP1306
Description:	A RECEIVER MUST NOT fault due to the presence of a UTF-8 Unicode Byte Order Mark. NOTE: This is a "scenario-dependent" test assertion, that will only trigger if the SOAP Header contains a header block of the form *:R4006 or *:R1019 (with any namespace prefix).
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents[fn:starts-with(fn:lower-case(@encoding), 'utf-8')]/soap12:Envelope[soap12:Header/*:R1019 or soap12:Header/*:R4006]
Predicate:	not (/wsil:testLog/wsdl:messageLog/wsdl:message[(@type = 'response' and @conversation = \$target/../../@conversation) or (./soap12:Envelope/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] = \$target/soap12:Header/wsa:MessageID)]/wsdl:messageContents/soap12:Envelope/soap12:Body/soap12:Fault)
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	A SOAP Fault has been generated when receiving a message with a UTF-8 Unicode Byte Order Mark. Please verify that this fault has another cause - it must not be caused by the presence of the B.O.M.
Diagnostic Data:	{SOAP message}

Test Assertion:	BP1307
Description:	A RECEIVER MUST NOT fault due to the presence of a UTF-16 Unicode Byte Order Mark. NOTE: This is a "scenario-dependent" test assertion, that will only trigger if the SOAP Header contains a header block of the form *:R4007 (with any namespace prefix).
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents[fn:starts-with(fn:lower-case(@encoding), 'utf-16')]/soap12:Envelope[soap12:Header/*:R4007]
Predicate:	not (/wsil:testLog/wsdl:messageLog/wsdl:message[(@type = 'response' and @conversation = \$target/../../@conversation) or (./soap12:Envelope/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] = \$target/soap12:Header/wsa:MessageID)]/wsdl:messageContents/soap12:Envelope/soap12:Body/soap12:Fault)
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	A soap Fault has been generated when receiving a message with a UTF-16 Unicode Byte Order Mark. Please verify that this Fault has another cause - it must not be caused by the presence of the B.O.M.
Diagnostic Data:	{SOAP message}

Test Assertion:	BP1019
Description:	The soap12:Envelope in the message is a well-formed XML 1.0 document.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope
co-Target: metadata	\$target/../../wsil:messageContents[@validXml and @xmlVersion]
Predicate:	\$target/../../wsil:messageContents[@validXml = fn:true() and @xmlVersion = '1.0']
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The soap12:Envelope does not conform to XML 1.0.
Diagnostic Data:	{SOAP message}{any XML parser error messages}

Test Assertion:	BP1018
Description:	The logged SOAP envelope is a UTF-8 transcript of an envelope originally encoded as UTF-8 or UTF-16. The HTTP Content-Type header's charset value is either UTF-8 or UTF-16. Looking at the messageContent element of the logged message, either (1) it has a BOM attribute which maps the charset value in the Content-Type header, or (2) it has it has an XML declaration which matches the charset value in the Content-Type header, or (3) there is no BOM attribute and no XML declaration, and the charset value is UTF-8.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope
Predicate:	\$target/../../wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'charset' and (fn:starts-with(fn:lower-case(@value), 'utf-8') or fn:starts-with(fn:lower-case(@value), 'utf-16'))] or fn:starts-with(fn:lower-case(\$target/../../wsil:messageContents/@encoding), 'utf-8') or fn:starts-with(fn:lower-case(\$target/../../wsil:messageContents/@encoding), 'utf-16')
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Either (1a) the message contains a Content-Type header but no charset value, or (1b) the charset value is neither UTF-8 nor UTF-16, or (2) there is a BOM attribute in the messageContent element, but its value does not match the charset value, or (3) there is an XML declaration, and the charset value does not match its value, or (4) there is no BOM attribute, no XML declaration, and the charset value in Content-Type header is not UTF-8.
Diagnostic Data:	Complete message.

Test Assertion:	BP1020
Description:	A XOP_ENCODED_MESSAGE MUST include the start-info parameter in the Content-Type header of the enclosing multipart/related MIME entity body.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[wsil:httpHeaders/wsdl:contentTypeHeader/@type = 'multipart' and wsil:httpHeaders/wsdl:contentTypeHeader/@subtype = 'related' and wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'type']/@value = 'application/xop+xml']
Predicate:	wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'start-info' and @value != '']
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The XOP_ENCODED_MESSAGE does not include the start-info parameter with proper @value.
Diagnostic Data:	

Test Assertion:	BP1021
Description:	A XOP_ENCODED_MESSAGE SHOULD include the full value of the type parameter from the root entity body part inside the start-info parameter of the enclosing multipart/related MIME entity body part's Content-Type header.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[wsil:httpHeaders/wsdl:contentTypeHeader/@type = 'multipart' and wsil:httpHeaders/wsdl:contentTypeHeader/@subtype = 'related' and wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'type']/@value = 'application/xop+xml']
Predicate:	some \$rootContentID in string(wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'start']/@value) satisfies string(wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'start-info']/@value) = string(wsil:messageAttachments/*:mimeHeaders[*:mimeHeader[fn:lower-case(@key) = 'content-id' and @value = \$rootContentID]]/wsil:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'type']/@value)
Reporting:	true=passed, false=failed
Prescription:	preferred

Error Message:	The XOP_ENCODED_MESSAGE SHOULD include the full value of the type parameter
Diagnostic Data:	

Test Assertion:	BP1600
Description:	The envelope conforms to the structure specified in SOAP 1.2 Part 1, Section 5.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope
co-Target: metadata	\$target/../../@schemaValid
Predicate:	\$target/../../@schemaValid = fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The envelope does not conform to the structure specified in SOAP 1.2 Part 1, Section 5
Diagnostic Data:	SOAP envelope.

Test Assertion:	BP1881
Description:	The envelope must have exactly zero or one child elements of the soap12:Body
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope
Predicate:	count(soap12:Body/*) le 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The envelope does not have exactly zero or one child elements of the soap12:Body
Diagnostic Data:	SOAP envelope.

Test Assertion:	BP1202
Description:	Each child element (if any) of the soap12:Body element is namespace qualified (not the grandchildren).
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope[count(soap12:Body/*) > 0]
Predicate:	fn:namespace-uri-from-QName(fn:node-name(soap12:Body/*)) != "
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A child element of the soap12:Body element is not namespace qualified.
Diagnostic Data:	SOAP envelope.

Test Assertion:	BP1007
Description:	DTDs relating to soap12:Header or soap12:Body documents, are not present in the envelope: no DOCTYPE element is present.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope
co-Target: metadata	\$target/../../@containsDTD
Predicate:	not(../../@containsDTD = fn:true())
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The soap12:Header or soap12:Body elements in the envelope, were described with an included DTD.
Diagnostic Data:	SOAP envelope.

Test Assertion:	BP1208
Description:	The SOAP envelope does not include XML processing instructions.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope
co-Target: metadata	\$target/./@containsProcessingInstructions
Predicate:	not(./@containsProcessingInstructions = fn:true())
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	a SOAP envelope contains XML processing instructions.
Diagnostic Data:	SOAP envelope.

Test Assertion:	BP1033
Description:	The SOAP envelope does not contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope
Predicate:	not(//*[fn:namespace-uri(.) = 'http://www.w3.org/XML/1998/namespace'])
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	The SOAP envelope contains the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".
Diagnostic Data:	SOAP envelope

Test Assertion:	BP1032
Description:	Test to see if the soap12:Envelope, soap12:Header, and soap12:Body elements have any attributes in either the SOAP 1.1 or the SOAP 1.2 namespace. Although this tests for additional constraints beyond those in R1032 (namely attributes in the SOAP 1.1 namespace), it is felt that the existence of such attributes represent an error.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope
Predicate:	every \$attrib in (self::node()/attribute::*, ./soap12:Header/attribute::*, ./soap12:Body/attribute::*) satisfies fn:namespace-uri(\$attrib) != 'http://www.w3.org/2003/05/soap-envelope' and fn:namespace-uri(\$attrib) != 'http://schemas.xmlsoap.org/soap/envelope/'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The message has a soap12:Envelope, soap12:Header, or soap12:Body element with one or more attributes in either the "http://schemas.xmlsoap.org/soap/envelope/" or the "http://www.w3.org/2003/05/soap-envelope" namespace.
Diagnostic Data:	SOAP envelope

Test Assertion:	BP1035
Description:	The value of the wsa:Action element is http://www.w3.org/2005/08/addressing/soap/fault
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:Action and (fn:ends-with(soap12:Body/soap12:Fault/soap12:Code/soap12:Value,'MustUnderstand') or fn:ends-with(soap12:Body/soap12:Fault/soap12:Code/soap12:Value, 'VersionMismatch'))]
Predicate:	normalize-space(string(./soap12:Header/wsa:Action)) eq 'http://www.w3.org/2005/08/addressing/soap/fault'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A fault response does not contain the expected wsa:Action value of http://www.w3.org/2005/08/addressing/soap/fault.
Diagnostic Data:	Complete message.

Test Assertion:	BP1040a
Description:	If an endpoint requires use of WS-Addressing by use of a wsam:Addressing policy assertion, an ENVELOPE sent by a SENDER MUST carry all required WS-Addressing SOAP headers. NOTE: this TA is only verifying the presence of the wsa:Action header. The presence of other headers (once wsa:Action is present) is verified by the family of TAs BP1142x and BP1143x (related to R1142), which also contribute at testing R1040.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:testLog/wsdl:descriptionFiles/*:feature[@name = 'http://www.w3.org/2007/05/addressing/metadata/Addressing' and @mode = 'required']]/wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)]
Predicate:	\$target/soap12:Header/wsa:Action
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The use of WS-Addressing was required, but the message did not include wsa:Action.
Diagnostic Data:	Complete message.

Test Assertion:	BP1040b
Description:	This assertions tests one aspect of R1040; if an instance requires the use of WS-Addressing with anonymous response EPRs, any envelope sent by a sender must not contain response EPRs that include anything other than "http://www.w3.org/2005/08/addressing/anonymous" as the value of their wsa:Address.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request'] /wsil:messageContents/soap12:Envelope [not(soap12:Header/wsa:RelatesTo) and not(soap12:Header/wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/none') and not(soap12:Header/wsa:FaultTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/none')] [/wsil:testLog/wsdl:descriptionFiles/wsdl:feature[@name = "http://www.w3.org/ns/ws-policy/Policy"]/wsil:alternative/wsdl:feature[@name = "http://www.w3.org/2007/05/addressing/metadata/Addressing"]/wsil:alternative/wsdl:feature[@name = "http://www.w3.org/2007/05/addressing/metadata/AnonymousResponses" and @mode = "required"]]
Predicate:	((not (\$target/soap12:Header/wsa:ReplyTo) or (\$target/soap12:Header/wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous')) and (not (\$target/soap12:Header/wsa:FaultTo) or (\$target/soap12:Header/wsa:FaultTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous')))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The service requires request messages to use response endpoint EPRs that contain the anonymous URI as the value of wsa:Address. The client sent a request with a response EPR that contained something other than the anonymous URI.
Diagnostic Data:	Complete message.

Test Assertion:	BP1040c
Description:	This assertions tests one aspect of R1040; if an instance requires the use of WS-Addressing with non-anonymous response EPRs, any envelope sent by a sender must not contain response EPRs that include anything other than "http://www.w3.org/2005/08/addressing/anonymous" as the value of their wsa:Address. Note also that the absence of a ReplyTo header in the envelope is not sufficient to pass this assertion. "http://www.w3.org/2005/08/addressing/anonymous".
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request'] /wsil:messageContents/soap12:Envelope [not(soap12:Header/wsa:RelatesTo) and not(soap12:Header/wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/none') and not(soap12:Header/wsa:FaultTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/none')] [/wsil:testLog/wsdl:descriptionFiles/wsdl:feature[@name = "http://www.w3.org/ns/ws-policy/Policy"]/wsil:alternative/wsdl:feature[@name = "http://www.w3.org/2007/05/addressing/metadata/Addressing"]/wsil:alternative/wsdl:feature[@name = "http://www.w3.org/2007/05/addressing/metadata/AnonymousResponses" and @mode = "required"]]
Predicate:	((\$target/soap12:Header/wsa:ReplyTo) and not (\$target/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous') and not (\$target/soap12:Header/wsa:FaultTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous'))
Reporting:	true=passed, false=failed
Prescription:	mandatory

Error Message:	The service requires request messages to use response endpoint EPRs that contain something other than the anonymous URI as the value of the anonymous URI or without the ReplyTo header, which implies a default [reply endpoint] with the anonymous URI.
Diagnostic Data:	Complete message.

Test Assertion:	BP1041
Description:	An ENVELOPE that is a MustUnderstand SOAP fault, sent from an endpoint that has a policy alternative containing the wsam:Addressing assertion attached to its WSDL endpoint subject, MUST NOT contain a NotUnderstood SOAP header block with the qname attribute value that identifies a WS-Addressing defined SOAP header block.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope[soap12:Body/soap12:Fault[some \$code in ./soap12:Value satisfies fn:contains(\$code,'MustUnderstand')]] [/wsil:testLog/wsdl:descriptionFiles/*:feature[@name = 'http://www.w3.org/2007/05/addressing/metadata/Addressing' and (@mode = 'required' or @mode = 'supported')]]
Predicate:	\$target/soap12:Header[not(soap12:NotUnderstood[fn:namespace-uri-for-prefix(fn:substring-before(@qname, ':'), .) = 'http://www.w3.org/2005/08/addressing'])]
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A MustUnderstand SOAP fault, sent from an endpoint that has a policy alternative containing the wsam:Addressing assertion attached to its WSDL endpoint subject, contains a NotUnderstood SOAP header block with the qname attribute value that identifies a WS-Addressing defined SOAP header block.
Diagnostic Data:	Complete message.

Test Assertion:	BP1142a
Description:	The envelope must conform to WS-Addressing WSDL Binding, Section 5.1. For a request message, with a wsa:Action element, defined using an rpc/lit binding, the following must hold: - If there is an EPR attached to the corresponding wsdl:port definition, then the wsa:To element must be present - the wsa:RelatesTo/@RelationshipType must be present with value "http://www.w3.org/2005/08/addressing/reply" (this is covered by 2nd cotarget presence) - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl - If the operation binding includes a wsdl:output element then there must be wsa:messageID and a wsa:ReplyTo elements present in the soap12:Header
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsdl:messageContents/soap12:Envelope [soap12:Header[wsdl:Action and not (wsdl:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)])] and soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsdl:output[@style = 'rpc']/wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]]
Predicate:	(if (/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/ wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/./@name]/wsa:EndpointReference) then (some \$portEpr in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/ wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/./@name]/wsa:EndpointReference satisfies (\$target/soap12:Header/wsa:To)) else fn:true()) and (if (\$myOpBinding/wsdl:output) then (\$target/soap12:Header/wsa:MessageID) else fn:true())) and (: correct wsdl:Action value :) (some \$opmsg in

	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[\$myOpBinding/@name = @name]/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace,'/'))) then '/' else "", \$opmsg/../../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/../../@name, 'Request'))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope does not conform to WS-Addressing WSDL Binding, Section 5.1. For a request message, with a wsa:Action element, defined using an rpc/lit binding, the following must hold: - If there is an EPR attached to the corresponding wsdl:port definition, then the wsa:To element must be present - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl - If the operation binding includes a wsdl:output element then there must be wsa:messageID and a wsa:ReplyTo elements present in the soap12:Header
Diagnostic Data:	Complete message.

Test Assertion:	BP1142b
Description:	The envelope must conform to WS-Addressing WSDL Binding, Section 5.1. http://www.w3.org/2005/08/addressing/reply IRI. For a response message with a wsa:Action element, defined using an rpc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have the anonymous address - the wsa:To element, if present, must have the anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'response']/wsil:messageContents/soap12:Envelope[soap12:Header[wsa:Action] and soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']]/ wsdl:operation [fn:string-join((@name, 'Response')," ") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
co-Target: related-Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	((not (\$related-Request/soap12:Header/wsa:ReplyTo) or (\$related-Request/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous')) and (\$target/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]) and (\$target/soap12:Header[not (wsa:To) or (wsa:To = 'http://www.w3.org/2005/08/addressing/anonymous')]) and (: correct wsa:Action value :) (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[\$myOpBinding/@name = @name]/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace,'/'))) then '/' else "", \$opmsg/../../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/../../@name, 'Response')))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope does not conform to WS-Addressing WSDL Binding, Section 5.1. For a response message with a wsa:Action element, defined using an rpc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have the anonymous address - the wsa:To element, if present, must have the anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl
Diagnostic Data:	Complete message.

Test Assertion:	BP1142c
Description:	The envelope must conform to WS-Addressing WSDL Binding, Section 5.1. http://www.w3.org/2005/08/addressing/reply IRI. For a non-anonymous response message with a

	wsa:Action element, defined using an rpc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have a non-anonymous address - the wsa:To element must be present, and must have a non-anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope [soap12:Body[not (soap12:Fault)] and soap12:Header [wsa:Action and (wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)])]] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']/ wsdl:operation [fn:string-join((@name, 'Response')," ") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
co-Target: related-Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	(((\$related-Request/soap12:Header/wsa:ReplyTo) and not (\$related-Request/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous')) and (\$target/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]) and (\$target/soap12:Header[(wsa:To) and not (wsa:To = 'http://www.w3.org/2005/08/addressing/anonymous')]) and (: correct wsa:Action value :) (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[\$myOpBinding/@name = @name]/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace,'/')) then '/' else "", \$opmsg/../../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/../../@name, 'Response'))))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope does not conform to WS-Addressing WSDL Binding, Section 5.1. For a non-anonymous response message with a wsa:Action element, defined using an rpc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have a non-anonymous address - the wsa:To element must be present, and must have a non-anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl
Diagnostic Data:	Complete message.

Test Assertion:	BP1143a
Description:	The envelope must conform to WS-Addressing WSDL Binding, Section 5.1. For a request message, with a wsa:Action element, defined using a doc/lit binding, the following must hold: - If there is an EPR attached to the corresponding wsdl:port definition, then the wsa:To element must be present - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl (this is covered by the cotarget presence) - If the operation binding includes a wsdl:output element then there must be wsa:messageID and a wsa:ReplyTo present in the soap12:Header
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/ soap12:Envelope [soap12:Body[not (soap12:Fault)] and soap12:Header [wsa:Action and not (wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)])]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/../../@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace,'/')) then '/' else "", \$opmsg/../../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/../../@name, 'Request')))) and (\$opBinding/@name = \$opmsg/../../@name)) else fn:true())]

Predicate:	(if (/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/ wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/..@name]/wsa:EndpointReference) then (some \$portEpr in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/ wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/..@name]/wsa:EndpointReference satisfies (\$target/soap12:Header/wsa:To)) else fn:true()) and (if (\$myOpBinding/wsdl:output) then (\$target/soap12:Header/wsa:MessageID) else fn:true()))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope does not conform to WS-Addressing WSDL Binding, Section 5.1. For a request message, with a wsa:Action element, defined using a doc/lit binding, the following must hold: - If there is an EPR attached to the corresponding wsdl:port definition, then the wsa:To element must be present - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl (this is covered by the cotarget presence) - If the operation binding includes a wsdl:output element then there must be wsa:messageID and a wsa:ReplyTo elements present in the soap12:Header
Diagnostic Data:	Complete message.

Test Assertion:	BP1143b
Description:	The envelope must conform to WS-Addressing WSDL Binding, Section 5.1. http://www.w3.org/2005/08/addressing/reply IRI. For a response message with a wsa:Action element, defined using a doc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have the anonymous address - the wsa:RelatesTo/@RelationshipType must be present with value "http://www.w3.org/2005/08/addressing/reply" (this is covered by 2nd cotarget presence) - the wsa:To element, if present, must have the anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl (this is covered by the cotarget presence)
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'response']/wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)] and soap12:Header[wsa:Action]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/..@name) else fn:true()) (: and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/..@name, ' ', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/..@name, 'Response'))) and (\$opBinding/@name = \$opmsg/..@name)) else fn:true()) :)]
co-Target: related-Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo/@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	(not (\$related-Request/soap12:Header/wsa:ReplyTo) or (\$related-Request/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous')) and (\$target/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]) and (\$target/soap12:Header[not (wsa:To) or (wsa:To = 'http://www.w3.org/2005/08/addressing/anonymous')])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope does not conform to WS-Addressing WSDL Binding, Section 5.1. For a response message with a wsa:Action element, defined using a doc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have the anonymous address - the wsa:To element, if present, must have the anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl (this is covered by the cotarget presence)
Diagnostic Data:	Complete message.

Test Assertion:	BP1143c
Description:	The envelope must conform to WS-Addressing WSDL Binding, Section 5.1. http://www.w3.org/2005/08/addressing/reply IRI. For a non anon response message with a wsa:Action element, defined using a doc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have a non-anonymous address - the wsa:RelatesTo/@RelationshipType must be present with value "http://www.w3.org/2005/08/addressing/reply" (this is covered by 2nd cotarget presence) - the wsa:To element, must be present and not anonymous - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl (this is covered by the cotarget presence)
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)] and soap12:Header [wsa:Action and (wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType))]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./@targetNamespace, if (not(fn:ends-with(\$opmsg/./@targetNamespace, '/')) then '/' else ' ', \$opmsg/./@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Response')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]
co-Target: related-Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	((\$related-Request/soap12:Header/wsa:ReplyTo) and not (\$related-Request/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous')) and (\$target/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]) and (\$target/soap12:Header[(wsa:To) and not (wsa:To = 'http://www.w3.org/2005/08/addressing/anonymous')])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope does not conform to WS-Addressing WSDL Binding, Section 5.1. For a non anon response message with a wsa:Action element, defined using a doc/lit binding, the following must hold: - the wsa:ReplyTo EPR in the related Request message must have a non-anonymous address - the wsa:To element, must be present with a non-anonymous value - The wsa:Action value in the soap12:Header must equal to that specified from the wsdl (this is covered by the cotarget presence)
Diagnostic Data:	Complete message.

Test Assertion:	BP1043a
Description:	This assertion tests one aspect of R1143. It verifies that, if the ReplyTo EPR contains reference parameters, these referece parameters are properly echoed any response that claims to have originated from an instance that supports WS-Addressing.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope[soap12:Header/wsa:ReplyTo/wsa:ReferenceParameters]
co-Target: related-Response	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or .wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo[not(@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Header/wsa:RelatesTo[not(@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] = \$target/soap12:Header/wsa:MessageID]
Predicate:	(if ((\$related-Response/soap12:Header/wsa:Action) and (not(\$related-Response/soap12:Body/soap12:Fault) or (not(\$target/soap12:Header/wsa:FaultTo) and \$related-Response/soap12:Header/wsa:Action ne 'http://www.w3.org/2005/08/addressing/fault')) then (every \$refP in \$target/soap12:Header/wsa:ReplyTo/wsa:ReferenceParameters/* satisfies some \$header in \$related-Response/soap12:Header/*[@wsa:lsReferenceParameter='1' or @wsa:lsReferenceParameter = 'true'] satisfies fn:resolve-QName(fn:name(\$refP), \$refP) = fn:resolve-QName(fn:name(\$header), \$header)) else fn:true())

Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The response to a message that contained reference parameters in the ReplyTo EPR is missing one or more of those reference parameters in its SOAP headers.
Diagnostic Data:	Complete message.

Test Assertion:	BP1043b
Description:	This assertion tests one aspect of R1143. It verifies that, if the FaultTo EPR contains reference parameters, these reference parameters are properly echoed any fault response that claims to have originated from an instance that supports WS-Addressing.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope[soap12:Header/wsa:FaultTo/wsa:ReferenceParameters]
co-Target: related-Response	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or . /wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelationshipType[not(@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Header/wsa:RelatesTo[not(@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] = \$target/soap12:Header/wsa:MessageID]
Predicate:	(if ((\$related-Response/soap12:Header/wsa:Action) and (\$related-Response/soap12:Body/soap12:Fault)) then (every \$refP in \$target/soap12:Header/wsa:FaultTo/wsa:ReferenceParameters/* satisfies some \$header in \$related-Response/soap12:Header/*[@wsa:isReferenceParameter='1' or @wsa:isReferenceParameter = 'true'] satisfies fn:resolve-QName(fn:name(\$refP), \$refP) = fn:resolve-QName(fn:name(\$header), \$header)) else fn:true())
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The fault response to a message that contained reference parameters in the FaultTo EPR is missing one or more of those reference parameters in its SOAP headers.
Diagnostic Data:	Complete message.

Test Assertion:	BP1153a
Description:	For a request message, defined using a doc/lit binding, the following must hold: - If there is no EPR attached to the corresponding wsdl:port definition, then in case there is no wsa:To element, the message should not be faulted.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsdl:messageContents/ soap12:Envelope [soap12:Body[not (soap12:Fault)] and soap12:Header[not(wsa:To)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/ @type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action))) and (some \$port in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port satisfies not(wsa:EndpointReference))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmsg/ @name] satisfies \$opBinding/ @name = \$dopmsg/.. / @name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/ @wsam:Action) then \$opmsg/ @wsam:Action else fn:concat(\$opmsg/.. /.. / @targetNamespace, if (not(fn:ends-with(\$opmsg/.. /.. / @targetNamespace, '))) then ' ' else ' ', \$opmsg/.. /.. / @name, ' ', if(\$opmsg/ @name) then \$opmsg/ @name else fn:concat(\$opmsg/.. / @name, 'Request')))) and (\$opBinding/ @name = \$opmsg/.. / @name)) else fn:true())]
Prerequisite:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/.. / @name and not(wsa:EndpointReference)]
Predicate:	(if (/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/ wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/.. / @name and

	not(wsa:EndpointReference))) then (not (/wsil:testLog/wsil:messageLog/wsil:message[(@type = 'response' and @conversation = \$target/../../@conversation) or (./soap12:Envelope/soap12:Header/wsa:RelatesTo = \$target/soap12:Header/wsa:MessageID)] /wsil:messageContents/soap12:Envelope/soap12:Body/soap12:Fault)) else fn:true())
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	An envelope for a doc/lit bound request message, without wsa:To in a message bound to a port that is NOT extended with wsa:EndpointReference has generated a Fault in response. Verify that this fault is not caused by the absence of wsa:To.
Diagnostic Data:	Complete message.

Test Assertion:	BP1153b
Description:	For a request message, defined using a rpc/lit binding, the following must hold: - If there is no EPR attached to the corresponding wsdl:port definition, then in case there is no wsa:To element, the message should not be faulted.
Target:	/wsil:testLog/wsil:messageLog/wsil:message[@type = 'request']/wsil:messageContents/soap12:Envelope [soap12:Header[not (wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]) and not(wsa:To)] and soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies (some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)) and (some \$port in /wsil:testLog/wsil:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port satisfies not(wsa:EndpointReference))]
co-Target: myOpBinding	/wsil:testLog/wsil:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsdl:port[@style = 'rpc']]/wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Prerequisite:	/wsil:testLog/wsil:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/./@name and not(wsa:EndpointReference)]
Predicate:	(if (/wsil:testLog/wsil:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/ wsdl:port[fn:local-name-from-QName(fn:resolve-QName(xsd:string(@binding), .)) = \$myOpBinding/./@name and not(wsa:EndpointReference)]) then (not (/wsil:testLog/wsil:messageLog/wsil:message[(@type = 'response' and @conversation = \$target/../../@conversation) or (./soap12:Envelope/soap12:Header/wsa:RelatesTo = \$target/soap12:Header/wsa:MessageID)] /wsil:messageContents/soap12:Envelope/soap12:Body/soap12:Fault)) else fn:true())
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	An envelope for a rpc/lit bound request message, without wsa:To in a message bound to a port that is NOT extended with wsa:EndpointReference has generated a Fault in response. Verify that this fault is not caused by the absence of wsa:To.
Diagnostic Data:	Complete message.

Test Assertion:	BP1125a
Description:	When a CONSUMER is non-addressable, a SOAP ENVELOPE, that is described by the input message of a WSDL doc-lit operation supported by an INSTANCE, MUST be bound to a HTTP Request message.
Target:	/wsil:testLog/wsil:messageLog/wsil:message/wsdl:messageContents/ soap12:Envelope [soap12:Body[not (soap12:Fault)] and not(soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]) and ((some \$repto in soap12:Header/wsa:ReplyTo satisfies fn:contains(\$repto, 'anonymous')) or not(soap12:Header/wsa:ReplyTo))] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies (not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action

	<pre>))) and (some \$port in /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port satisfies not(wsa:EndpointReference))] </pre>
co-Target: myOpBinding	<pre> /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve- QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./@targetNamespace, if (not(fn:ends-with(\$opmsg/./@targetNamespace, '))) then ' ' else ' ', \$opmsg/./@name, ' ', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Request')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())] </pre>
Predicate:	\$target/./@type = 'request'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope for a WSDL doc/lit bound request message, with a wsa:ReplyTo anonymous, was sent over an HTTP response.
Diagnostic Data:	Complete message.

Test Assertion:	BP1125b
Description:	When a CONSUMER is non-addressable, a SOAP ENVELOPE, that is described by the input message of a WSDL rpc-lit operation supported by an INSTANCE, MUST be bound to a HTTP Request message.
Target:	<pre> /wsdl:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/ soap12:Envelope [soap12:Body[not (soap12:Fault)] and not(soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]) and ((some \$repto in soap12:Header/wsa:ReplyTo satisfies fn:contains(\$repto, 'anonymous')) or not(soap12:Header/wsa:ReplyTo))] [some \$myenv in . satisfies (some \$message in /wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)) and (some \$port in /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port satisfies not(wsa:EndpointReference))] </pre>
co-Target: myOpBinding	<pre> /wsdl:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsdl:operation[@style = 'rpc']/wsdl:operation[@name = fn:local-name-from-QName(node- name(\$target/soap12:Body/*[1]))] </pre>
Predicate:	\$target/./@type = 'request'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope for a WSDL rpc/lit bound request message, with a wsa:ReplyTo anonymous, was sent over an HTTP response.
Diagnostic Data:	Complete message.

Test Assertion:	BP1126a
Description:	When a CONSUMER is non-addressable, a SOAP ENVELOPE, that is described by the output message of a WSDL doc-lit operation supported by an INSTANCE, MUST be bound to a HTTP Response message.
Target:	<pre> /wsdl:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope [soap12:Body[not (soap12:Fault)] and soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and (some \$respenv in . satisfies some \$req in /wsdl:testLog/wsdl:messageLog/wsdl:message satisfies \$req/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$respenv/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' </pre>

	or not (@RelationshipType)] and \$req/wsdl:messageContents/soap12:Envelope/soap12:Header[fn:contains(wsa:ReplyTo[1]/wsa:Address , 'http://www.w3.org/2005/08/addressing/anonymous') or not(wsa:ReplyTo))] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./@targetNamespace, if (not(fn:ends-with(\$opmsg/./@targetNamespace,'')) then '' else ' ', \$opmsg/./@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Response')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]
Predicate:	\$target/./@type = 'response'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope for a WSDL doc/lit bound response message, related to a request containing a wsa:ReplyTo anonymous, was sent over an HTTP request.
Diagnostic Data:	Complete message.

Test Assertion:	BP1126b
Description:	When a CONSUMER is non-addressable, a SOAP ENVELOPE, that is described by the output message of a WSDL rpc-lit operation supported by an INSTANCE, MUST be bound to a HTTP Response message.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message/wsdl:messageContents/soap12:Envelope [soap12:Body[not (soap12:Fault)] and soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and (some \$respenv in . satisfies some \$req in /wsil:testLog/wsdl:messageLog/wsdl:message satisfies \$req/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$respenv/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and \$req/wsdl:messageContents/soap12:Envelope/soap12:Header[fn:contains(wsa:ReplyTo[1]/wsa:Address, 'http://www.w3.org/2005/08/addressing/anonymous') or not(wsa:ReplyTo))] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsdl:operation [fn:string-join((@name, 'Response'), " ") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	\$target/./@type = 'response'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An envelope for a WSDL rpc/lit bound response message, related to a request containing a wsa:ReplyTo anonymous, was sent over an HTTP request.
Diagnostic Data:	Complete message.

Test Assertion:	BP1144
Description:	The message must specify an action HTTP header with either a value that is an absolute URI that has the same value as the value of the wsa:Action SOAP header, or a value of "" (empty string).
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:Action]
Prerequisite:	BP1761
Predicate:	(wsil:messageContents/soap12:Envelope/soap12:Header/wsa:Action/text() = wsil:httpHeaders/wsdl:httpHeader[fn:lower-case(@key) = 'action']/@value) or

	not(wsil:httpHeaders/wsdl:httpHeader[fn:lower-case(@key) = 'action'])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A message did not specify an action HTTP header with either a value that is an absolute URI that has the same value as the value of the wsdl:Action SOAP header, or a value of "" (empty string).
Diagnostic Data:	Complete message.

Test Assertion:	BP1002
Description:	If it is a request, the arg #2 of POST is <HTTP/1.1> or <HTTP/1.0>. If absent, first line of the body is: HTTP-Version = HTTP/1.1. (or HTTP/1.0). If it is a response, it starts with <HTTP/1.1> or <HTTP/1.0>
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request' and fn:contains(wsdl:httpHeaders/wsdl:requestLine/text(), 'HTTP')]
Predicate:	fn:contains(wsdl:httpHeaders/wsdl:requestLine/text(), 'HTTP/1.0') or fn:contains(wsdl:httpHeaders/wsdl:requestLine/text(), 'HTTP/1.1')
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The message that is sent over HTTP, is not sent using HTTP/1.1 or HTTP/1.0.
Diagnostic Data:	All HTTP headers.

Test Assertion:	BP1001
Description:	If it is a request, the arg #2 of POST is <HTTP/1.1>. If absent, first line of the body is: HTTP-Version = HTTP/1.1. If it is a response, it starts with <HTTP/1.1>
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message
Prerequisite:	BP1002
Predicate:	contains(wsdl:httpHeaders/wsdl:requestLine/text(), 'HTTP/1.1')
Reporting:	true=passed, false=warning
Prescription:	preferred
Error Message:	The message is not sent using HTTP/1.1.
Diagnostic Data:	All HTTP headers.

Test Assertion:	BP1006
Description:	The action header contains a quoted string of any value, including "".
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request'][wsdl:httpHeaders/wsdl:contentTypeHeader]
Prerequisite:	BP1761
Predicate:	every \$par in wsdl:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter satisfies not(\$par/@quoted) or not(fn:lower-case(\$par/@key) eq 'type' or fn:lower-case(\$par/@key) eq 'start-info' or fn:lower-case(\$par/@key) eq 'action' or fn:lower-case(\$par/@key) eq 'boundary') or \$par/@quoted = fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	action HTTP header does not contain a quoted string.
Diagnostic Data:	All HTTP headers.

Test Assertion:	BP1100
Description:	The message uses a "200 OK" HTTP status code.
Target:	//wsdl:message[@type='response'][wsdl:messageContents/soap12:Envelope/soap12:Body[not(soap12:Fault)]]
Predicate:	contains(wsdl:httpHeaders/wsdl:requestLine, "200")
Reporting:	true=passed, false=failed

Prescription:	preferred
Error Message:	The envelope of the response message does not contain a soap12:Fault and the message does not use a "200 OK" HTTP status code.
Diagnostic Data:	Complete message.

Test Assertion:	BP1101
Description:	The response message, if successfully processed at HTTP level, is sent using either a "200 OK" or "202 Accepted" HTTP status code.
Target:	//wsil:message[@type='response'][wsil:messageContents[not(soap12:Envelope)]] [not(matches(wsil:httpHeaders/wsil:requestLine, "4[0-9][0-9]"))]
Predicate:	contains(wsil:httpHeaders/wsil:requestLine, "200") or contains(wsil:httpHeaders/wsil:requestLine, "202 Accepted")
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	A response message without a SOAP message, is not using either a "200 OK" or "202 Accepted" HTTP status code, though successful at HTTP level.
Diagnostic Data:	Complete message.

Test Assertion:	BP1150
Description:	The message must be a response sent in the entity body of the HTTP Response using the same HTTP connection as the request message of that operation.
Target:	/wsil:testLog/wsil:messageLog/wsil:message[*[contains(., 'MustUnderstand') or contains(., 'VersionMismatch')]]
Predicate:	wsil:messageContents/soap12:Envelope/soap12:Body/soap12:Fault/soap12:Code/soap12:Value[contains(., 'MustUnderstand') or contains(., 'VersionMismatch')]
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A response message was not sent in the entity body of the HTTP Response using the same HTTP connection as the request message of that operation.
Diagnostic Data:	Complete message

Test Assertion:	BP1151
Description:	The message must be sent in the entity body of an HTTP request in a separate HTTP connection specified by the response EPR using the SOAP 1.1 Request Optional Response HTTP binding.
Target:	/wsil:testLog/wsil:messageLog/wsil:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]] [./wsil:messageContents/soap12:Envelope [not (soap12:Body/soap12:Fault [contains(soap12:Code/soap12:Value, 'MustUnderstand') or contains(soap12:Code/soap12:Value, 'VersionMismatch')])]]]
co-Target: myRequestMsg	/wsil:testLog/wsil:messageLog/wsil:message [@type = 'request' and (if (not(./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID)) then (@conversation = \$target/./@conversation) else (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]))]
Prerequisite:	\$myRequestMsg[1]/wsil:messageContents/soap12:Envelope/soap12:Header[wsa:ReplyTo and not(wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous')]
Predicate:	(\$target/@type = 'request') and (\$target/wsil:messageContents/soap12:Envelope/soap12:Header [(wsa:To) and not (wsa:To = 'http://www.w3.org/2005/08/addressing/anonymous')])
Reporting:	true=passed, false=failed

Prescription:	mandatory
Error Message:	A message was not sent in the entity body of an HTTP request in a separate HTTP connection specified by the response EPR using the SOAP 1.1 Request Optional Response HTTP binding.
Diagnostic Data:	Complete message

Test Assertion:	BP1146
Description:	A non-faulting response must be sent to the endpoint referred to by the wsa:ReplyTo header or generate a fault instead.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [(@type = 'response' and (some \$resp in . satisfies some \$req in /wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request' and @conversation = \$resp/@conversation] satisfies \$req/wsa:*)) or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]] [wsil:messageContents/soap12:Envelope[not(soap12:Body/soap12:Fault)]]
co-Target: myRequestMsg	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and (if (not(./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID)) then (@conversation = \$target/./@conversation) else (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]))]
Predicate:	((not(\$myRequestMsg/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:ReplyTo) or \$myRequestMsg/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous') and \$target[@type = 'response']/wsil:messageContents/soap12:Envelope/soap12:Header [wsa:To = 'http://www.w3.org/2005/08/addressing/anonymous' or not(wsa:To)]) or (\$target/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:To = \$myRequestMsg/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:ReplyTo/wsa:Address)
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A response message was sent to a destination other than the expected endpoint, i.e. different from wsa:ReplyTo in the request message.
Diagnostic Data:	Complete message

Test Assertion:	BP1152a
Description:	The message must be sent in the entity body of an HTTP request in a separate HTTP connection specified by the response EPR using the SOAP 1.1 Request Optional Response HTTP binding.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and wsil:messageContents/soap12:Envelope/soap12:Body[not(soap12:Fault)]] and (some \$req in /wsil:testLog/wsdl:messageLog/wsdl:message satisfies \$req/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and \$req/wsdl:messageContents/soap12:Envelope/soap12:Header[wsa:ReplyTo[1] and not(fn:contains(wsa:ReplyTo[1]/wsa:Address, 'http://www.w3.org/2005/08/addressing/anonymous'))]]
co-Target: myRequestMsg	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$target/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	(\$target/@type = 'request') and (\$target/wsdl:messageContents/soap12:Envelope/soap12:Header [(wsa:To) and not (fn:contains(wsa:To[1], 'http://www.w3.org/2005/08/addressing/anonymous'))])

Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A response message was not sent in the entity body of an HTTP request in a separate HTTP connection as required by the (non-anonymous) response EPR.
Diagnostic Data:	Complete message

Test Assertion:	BP1152b
Description:	The message must be sent in the entity body of an HTTP request in a separate HTTP connection specified by the response EPR using the SOAP 1.1 Request Optional Response HTTP binding.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and wsil:messageContents/soap12:Envelope/soap12:Body[soap12:Fault] and wsil:messageContents/soap12:Envelope/soap12:Header[not(wsa:Action) or not(fn:starts-with(wsa:Action,'http://www.w3.org/2005/08/addressing/'))]] and (some \$req in /wsil:testLog/wsdl:messageLog/wsdl:message satisfies \$req/wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)] and \$req/wsil:messageContents/soap12:Envelope/soap12:Header[wsa:FaultTo[1] and not(fn:contains(wsa:FaultTo[1]/wsa:Address, 'http://www.w3.org/2005/08/addressing/anonymous'))]]]]
co-Target: myRequestMsg	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$target/wsdl:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]]
Predicate:	(\$target/@type = 'request') and (\$target/wsdl:messageContents/soap12:Envelope/soap12:Header [(wsa:To) and not (fn:contains(wsa:To[1], 'http://www.w3.org/2005/08/addressing/anonymous'))])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A response message was not sent in the entity body of an HTTP request in a separate HTTP connection as required by the (non-anonymous) response EPR.
Diagnostic Data:	Complete message

Test Assertion:	BP1152c
Description:	The message must be sent in the entity body of an HTTP request in a separate HTTP connection specified by the response EPR using the SOAP 1.1 Request Optional Response HTTP binding.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[not (wsil:messageContents/soap12:Envelope/soap12:Body/soap12:Fault) and wsil:messageContents/soap12:Envelope/soap12:Header [(wsa:To) and not (fn:contains(wsa:To[1], 'http://www.w3.org/2005/08/addressing/anonymous'))]]]
Predicate:	\$target/@type = 'request'
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A message with a non-anonymous destination was not sent in the entity body of an HTTP request in a separate HTTP connection as required by the (non-anonymous) destination EPR.
Diagnostic Data:	Complete message

Test Assertion:	BP2703
Description:	The wsdl:definitions is a well-formed XML 1.0 document. The wsdl:definitions namespace has value: http://schemas.xmlsoap.org/wsdl/.

Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile[fn:prefix-from-QName(fn:node-name(*:definitions)) eq 'wsdl' or wsdl:definitions]
Predicate:	fn:namespace-uri-from-QName(fn:node-name(*:definitions)) eq 'http://schemas.xmlsoap.org/wsdl/' or /*:bindingTemplate
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsdl:definitions is not a well-formed XML 1.0 document, or WSDL definition does not conform to the schema located at http://schemas.xmlsoap.org/wsdl/soap/2003-02-11.xsd for some element using the WSDL-SOAP binding namespace, or does not conform to the schema located at http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd for some element using the WSDL namespace.
Diagnostic Data:	Error message from the XML parser

Test Assertion:	BP2756
Description:	A DESCRIPTION MUST NOT include the wsoap12:soapActionRequired attribute
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions
Predicate:	not (./wsoap12:operation/@soapActionRequired)
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A DESCRIPTION include the wsoap12:soapActionRequired attribute
Diagnostic Data:	

Test Assertion:	BP1761
Description:	A SOAP1.2 SENDER SHOULD NOT include the SOAPAction HTTP header
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[wsil:httpHeaders]
Predicate:	not(wsil:httpHeaders/wsdl:httpHeader[fn:lower-case(@key) = 'soapaction'])
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	A SOAP1.2 SENDER has included the SOAPAction HTTP header (instead of using the 'action' header)
Diagnostic Data:	

Test Assertion:	BP1757
Description:	A receiver must not fault messages that do not have the action parameter on the Content-Type HTTP header field.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message[not(wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'action'])]/wsil:messageContents/soap12:Envelope
Prerequisite:	BP1761
Predicate:	not (/wsil:testLog/wsdl:messageLog/wsdl:message[(@type = 'response' and @conversation = \$target/./../@conversation) or (./soap12:Envelope/soap12:Header/wsa:RelatesTo = \$target/soap12:Header/wsa:MessageID)]/wsil:messageContents/soap12:Envelope/soap12:Body/soap12:Fault)
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	A soap Fault has been generated when receiving a message with no action parameter on the Content-Type HTTP header field: verify that this Fault has another cause - it must not be caused by this parameter absence.
Diagnostic Data:	{SOAP message}

Test Assertion:	BP2704
Description:	The wsdl:definitions using the WSDL SOAP binding namespace (prefixed "wsoap12" in this Profile) MUST be valid according to the XML Schema found at " http://schemas.xmlsoap.org/wsdl/soap12/wsdl11soap12.xsd ".

Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[./wsoap12:*]
co-Target: metadata	\$target/./@schemaValid
Predicate:	\$target/./@schemaValid = fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsdl:definitions is using the WSDL SOAP binding namespace (prefixed "wsoap12" in this Profile) in a way that is not valid according to the XML Schema found at " http://schemas.xmlsoap.org/wsdl/soap12/wsdl11soap12.xsd ".
Diagnostic Data:	Error message from the XML parser

Test Assertion:	BP2101
Description:	Each wsdl:import statement is only used to import another WSDL description.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[wsdl:import]
Predicate:	every \$wsdlImp in \$target/wsdl:import satisfies some \$wsdlFile in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile[@filename=\$wsdlImp/@location] satisfies ((fn:namespace-uri(\$wsdlFile/*[1]) = 'http://schemas.xmlsoap.org/wsdl/') and (fn:local-name(\$wsdlFile/*[1]) = 'definitions'))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:import element does not contain a reference to another WSDL description.
Diagnostic Data:	wsdl:import element(s) that does not reference a WSDL description.

Test Assertion:	BP2803
Description:	The "namespace" attribute's value is not a relative URI.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[wsdl:import]
Prerequisite:	BP2703
Predicate:	every \$imp in wsdl:import[@namespace] satisfies (starts-with(\$imp/@namespace, 'http://') or starts-with(\$imp/@namespace, 'HTTP://') or starts- with(\$imp/@namespace, 'urn:') or starts-with(\$imp/@namespace, 'URN:') or starts- with(\$imp/@namespace, 'ft') or starts-with(\$imp/@namespace, 'FT'))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsdl:import element's "namespace" attribute value is a relative URI.
Diagnostic Data:	Defective wsdl:import element.

Test Assertion:	BP2103
Description:	For the referenced definitions as well as all imported descriptions, The XML schema import statement is only used within an xsd:schema element.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[descendant::xsd:import]
Predicate:	every \$imp in descendant::xsd:import satisfies (\$imp/ancestor::xsd:schema and \$imp/ancestor::wsdl:types)
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A XML schema import element was found outside of an xsd:schema element.
Diagnostic Data:	Defective XML schema import element.

Test Assertion:	BP2202
------------------------	---------------

Description:	All imported schema use UTF-8 or UTF-16 for the encoding.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[wsdl:types/xsd:schema/xsd:import]
Predicate:	every \$schImp in \$target/wsdl:types/xsd:schema/xsd:import[@schemaLocation] satisfies (some \$schFile in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile[@filename=\$schImp/@schemaLocation] satisfies (not (\$schFile/@encoding) or \$schFile/@encoding = 'UTF-8' or \$schFile/@encoding = 'UTF-16' or \$schFile/@encoding = 'utf-8' or \$schFile/@encoding = 'utf-16'))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The XML declaration statement within an imported XML Schema does not use expected encoding (UTF-8 or UTF-16).
Diagnostic Data:	XML declaration statement.

Test Assertion:	BP2098
Description:	The "location" attribute is specified for the wsdl:import element, and has a non-empty value.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:import
Predicate:	@location and @location != ""
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:import element does not have a "location" attribute, or has an empty value for the location attribute.
Diagnostic Data:	Defective wsdl:import element.

Test Assertion:	BP2105
Description:	wsdl:import elements occur either as first children elements in the WSDL namespace of the wsdl:definitions element, or they are only preceded by wsdl:documentation elements.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[wsdl:import]
Predicate:	not(wsdl:import/preceding-sibling::wsdl:*[(local-name-from-QName(node-name(.)) != 'documentation') and (local-name-from-QName(node-name(.)) != 'import')])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:import element in the WSDL namespace under the wsdl:definitions element, is preceded by child elements other than wsdl:documentation elements.
Diagnostic Data:	Display the WSDL import element(s) that failed the assertion.

Test Assertion:	BP2018
Description:	The wsdl:types elements occur either as first children in the WSDL namespace of the wsdl:definitions element, or they are only preceded by wsdl:documentation element(s) and/or wsdl:import(s) element(s).
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions[wsdl:types]
Prerequisite:	BP2703
Predicate:	not(wsdl:types/preceding-sibling::wsdl:*[(local-name-from-QName(node-name(.)) != 'documentation') and (local-name-from-QName(node-name(.)) != 'import') and (local-name-from-QName(node-name(.)) != 'types')])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	wsdl:types element(s) in the WSDL namespace of the wsdl:definitions element were preceded by child elements other than wsdl:documentation or wsdl:import elements.
Diagnostic Data:	

Test Assertion:	BP2700
Description:	The wsdl:definitions is a well-formed XML 1.0 document.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions
co-Target: metadata	\$target/./@validXml
Predicate:	./@validXml=fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsdl:definitions is not a well-formed XML 1.0 document.
Diagnostic Data:	Error message from the XML parser

Test Assertion:	BP2034
Description:	The candidate description does not contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions
Prerequisite:	BP2703
Predicate:	not(descendant-or-self::*:namespace::node())[1] = 'xmlns:xml="http://www.w3.org/XML/1998/namespace"'
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	The candidate description contains the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".
Diagnostic Data:	

Test Assertion:	BP2201
Description:	The XML declaration statement uses UTF-8 or UTF-16 for the encoding.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions
co-Target: metadata	\$target/./@encoding
Predicate:	fn:starts-with(fn:lower-case(./@encoding), 'utf-8') or fn:starts-with(fn:lower-case(./@encoding), 'utf-16')
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	XML declaration statement within WSDL document does not use expected encoding (UTF-8 or UTF-16).
Diagnostic Data:	XML declaration statement.

Test Assertion:	BP2104
Description:	The targetNamespace attribute on the wsdl:definitions element for the imported WSDL description has the same value as the namespace attribute on the wsdl:import element that imported the WSDL description.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:import
Prerequisite:	BP2101
Predicate:	some \$wimp in . satisfies some \$descFile in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile[fn:ends-with(\$wimp/@location, @filename)] satisfies (\$descFile/wsdl:definitions/@targetNamespace = \$wimp/@namespace)
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The targetNamespace attribute on the wsdl:definitions element for an imported WSDL description does not have the same value as the namespace attribute on the wsdl:import element that imported the WSDL description.
Diagnostic Data:	wsdl:import element.

Test Assertion:	BP2123
Description:	Contained WSDL extension elements do not use the wsdl:required attribute value of "true".
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/descendant::* [namespace-uri(.) != 'http://schemas.xmlsoap.org/wsdl/' and not(namespace-uri(.) = 'http://schemas.xmlsoap.org/wsdl/soap/' and (local-name(.) = 'binding' or local-name(.) = 'operation' or local-name(.) = 'body' or local-name(.) = 'header' or local-name(.) = 'headerfault' or local-name(.) = 'fault' or local-name(.) = 'address')) and (./ancestor::wsdl:portType or ./ancestor::wsdl:binding or ./ancestor::wsdl:message or ./ancestor::wsdl:types or ./ancestor::wsdl:import)]
Predicate:	not(@wsdl:required = fn:true())
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	An extension element within a WSDL element that claims conformance to the Basic Profile has a wsdl:required attribute with a value of "true".
Diagnostic Data:	Display the extension element that failed the assertion.

Test Assertion:	BP2416
Description:	A DESCRIPTION MUST NOT use QName references to WSDL components in namespaces that have been neither imported, nor defined in the referring WSDL document.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions
Predicate:	(every \$qref in \$target/wsdl:service/wsdl:port/@binding satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))) and (every \$qref in \$target/wsdl:binding/@type satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))) and (every \$qref in \$target/wsdl:portType/wsdl:operation/wsdl:input/@message satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))) and (every \$qref in \$target/wsdl:portType/wsdl:operation/wsdl:output/@message satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))) and (every \$qref in \$target/wsdl:portType/wsdl:operation/wsdl:fault/@message satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))) and (every \$qref in \$target/wsdl:binding/wsdl:operation/wsdl:*/*soap12:header/@message satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))) and (every \$qref in \$target/wsdl:binding/wsdl:operation/wsdl:*/*soap12:headerfault/@message satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$wimp in \$target/wsdl:import satisfies \$wimp/@namespace = \$ns)))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	This DESCRIPTION uses QName references to WSDL components in namespaces that are neither imported, nor defined in the referring WSDL document
Diagnostic Data:	Defective QName(s).

Test Assertion:	BP2417
Description:	A QName reference to a Schema component in a DESCRIPTION MUST use the namespace defined in the targetNamespace attribute on the xsd:schema element, or to a namespace defined in the namespace attribute on an

	xsd:import element within the xsd:schema element.
Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions
Predicate:	(every \$qref in \$target/wsdl:message/wsdl:part/@element satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/wsdl:types/xsd:schema/@targetNamespace = \$ns) or (some \$simp in \$target/wsdl:types/xsd:schema/xsd:import satisfies \$simp/@namespace = \$ns))) and (every \$qref in \$target/wsdl:message/wsdl:part/@type satisfies (some \$ns in fn:namespace-uri-from-QName(fn:resolve-QName (xsd:string(\$qref),\$qref/..)) satisfies ((\$target/@targetNamespace = \$ns) or (some \$simp in \$target/wsdl:types/xsd:schema/xsd:import satisfies \$simp/@namespace = \$ns) or (\$ns = 'http://www.w3.org/2001/XMLSchema'))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A QName reference to a Schema component in a DESCRIPTION does not use the namespace defined in the targetNamespace attribute on the xsd:schema element, or a namespace defined in the namespace attribute on an xsd:import element within the xsd:schema element.
Diagnostic Data:	Defective QName(s).

Test Assertion:	BP2106
Description:	In a DESCRIPTION the schemaLocation attribute of an xsd:import element MUST NOT resolve to any document whose root element is not "schema" from the namespace "http://www.w3.org/2001/XMLSchema".
Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions[wsdl:types/xsd:schema/xsd:import]
Predicate:	every \$schemaImp in \$target/wsdl:types/xsd:schema/xsd:import[@schemaLocation] satisfies some \$impSchemaFile in /wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile[@filename=\$schemaImp/@schemaLocation] satisfies (fn:namespace-uri(\$impSchemaFile/*[1])='http://www.w3.org/2001/XMLSchema' and fn:local-name(\$impSchemaFile/*[1])='schema')
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	xsd:import element does not resolve to document whose root element is not "schema" from xsd namespace
Diagnostic Data:	Defective XML schema import element.

Test Assertion:	BP2107
Description:	The xsd:schema element contains a targetNamespace attribute with a valid and non-null value unless the xsd:schema element has xsd:import and/or xsd:annotation as its only child element(s).
Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions[wsdl:types/xsd:schema]
Predicate:	wsdl:types/xsd:schema[@targetNamespace != " or (count(xsd:import) + count(xsd:annotation)) = count(child::*)]
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A xsd:schema element contained in a wsdl:types element does not have a targetNamespace attribute with a valid and non-null value, while having child element(s) other than xsd:import or xsd:annotation.
Diagnostic Data:	Defective xsd:schema element(s).

Test Assertion:	BP2108b
Description:	The type soapenc:Array does not appear in these declarations, and the wsdl:arrayType attribute is not used in the type declaration.

Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions/wsdl:types[xsd:schema]
Predicate:	not (xsd:schema[descendant::xsd:extension/@base = "soapenc:Array" or descendant::xsd:restriction/@base = "soapenc:Array"])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An Array declaration uses - restricts or extends - the soapenc:Array type, or the wsdl:arrayType attribute is used in the type declaration.
Diagnostic Data:	Defective declaration(s).

Test Assertion:	BP2108a
Description:	The type soapenc:Array does not appear in these declarations, and the wsdl:arrayType attribute is not used in the type declaration.
Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions/wsdl:types[xsd:schema]
Predicate:	not (xsd:schema[descendant::*/@wsdl:arrayType])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An Array declaration uses - restricts or extends - the soapenc:Array type, or the wsdl:arrayType attribute is used in the type declaration.
Diagnostic Data:	Defective declaration(s).

Test Assertion:	BP2110
Description:	The declaration does not use the naming convention ArrayOfXXX.
Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions[wsdl:types/xsd:schema]
Predicate:	not (wsdl:types/xsd:schema[descendant::xsd:element[fn:starts-with(@name, 'ArrayOf')]])
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	A declaration is using the convention ArrayOfXXX.
Diagnostic Data:	Defective declaration(s).

Test Assertion:	BP1204
Description:	The soap12:Body of the envelope does not contain the soapenc:arrayType attribute.
Target:	//soap12:Envelope
Predicate:	not (descendant::*[@soapenc:arrayType])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The soap12:Body of an envelope contains the soapenc:arrayType attribute.
Diagnostic Data:	SOAP envelope.

Test Assertion:	BP2124
Description:	No two global element declarations share the same qualified name.
Target:	/wsil:testLog/wsil:descriptionFiles/wsil:descriptionFile/wsdl:definitions/wsdl:types/xsd:schema
Predicate:	count (xsd:element[@name = preceding-sibling::element/@name]) = 0
Reporting:	true=passed, false=failed
Prescription:	preferred

Error Message:	Two or more global element declarations share the same qualified name.
Diagnostic Data:	The wsdl:operation elements containing the repeated global element declarations.

Test Assertion:	BP2125
Description:	No two type definitions share the same qualified name.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:types/xsd:schema
Predicate:	(count (xsd:ComplexType[@name = preceding-sibling::ComplexType/@name]) = 0) and (count (xsd:SimpleType[@name = preceding-sibling::SimpleType/@name]) = 0)
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	Two or more type definitions share the same qualified name.
Diagnostic Data:	The elements containing the repeated qualified name.

Test Assertion:	BP2111
Description:	If the "parts" attribute is present, then the wsoap12:body element(s) have at most one part listed in the parts attribute.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[not(../wsoap12:*[@style = 'rpc'])]
Prerequisite:	BP2017
Predicate:	not(../wsoap12:body[@parts and contains(@parts, " ")])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	One or more wsoap12:body element(s) in a document-literal soabinding does not have at most one part listed in the parts attribute.
Diagnostic Data:	Defective wsoap12:body element(s).

Test Assertion:	BP2119
Description:	If it does not specify the parts attribute on a wsoap12:body element, the corresponding abstract wsdl:message defines zero or one wsdl:part.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[not(../wsoap12:*[@style = 'rpc'])]
Prerequisite:	BP2017
Predicate:	every \$sbody in wsdl:operation//wsoap12:body[not(@parts)] satisfies some \$msgname in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$sbody/../../@name]/*[local-name(.) = local-name(\$sbody/..)]/@message satisfies count(/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = \$msgname]/wsdl:part) le 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A document-literal binding which does not specify the parts attribute, has more than one wsdl:part in the associated wsdl:message element.
Diagnostic Data:	Defective wsdl:binding element.

Test Assertion:	BP2013
Description:	The binding (in wsoap12:body elements) only refers to part elements that have been defined using the

	"type" attribute
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wssoap12:*[@style = 'rpc']]
Prerequisite:	BP2017
Predicate:	every \$sbody in wsdl:operation/wssoap12:body satisfies some \$msgname in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$sbody/././@name]*/[local-name(.) = local-name(\$sbody/./)]/@message satisfies every \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = \$msgname]/wsdl:part satisfies \$mpart/@type or \$sbody/@parts and not(contains(\$sbody/@parts, \$mpart/@name))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	soabinding (in wssoap12:body elements) refers to part(s) that do not have the "type" attribute.
Diagnostic Data:	{binding}{message with failed part}

Test Assertion:	BP2012
Description:	A doc/wssoap12:body/@use != 'literal') and (count(\$myOpBinding/./wssoap12:body) = count(\$myOpBinding/./wssoap12:body/@use)) and not(\$myOpBinding/./wssoap12:*[@style = 'rpc'] and \$myOpBinding/wsdl:output/wssoap12:body[@parts and (@parts = " or @parts = ' ')] ument-literal binding only refers in wssoap12:body elements to part elements that have been defined using the "element" attribute.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[not(./wssoap12:*[@style = 'rpc'])]
Prerequisite:	BP2017
Predicate:	every \$sbody in wsdl:operation/wssoap12:body satisfies some \$msgname in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$sbody/././@name]*/[local-name(.) = local-name(\$sbody/./)]/@message satisfies every \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = \$msgname]/wsdl:part satisfies \$mpart/@element or \$sbody/@parts and not(contains(\$sbody/@parts, \$mpart/@name))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The binding (in wssoap12:body elements) refers to part(s) of a soap12:Body element that do not have the "element" attribute.
Diagnostic Data:	wssoap12:body element(s) that have non "element" parts attributes.

Test Assertion:	BP1212a
Description:	The Envelope of request message contains exactly one part accessor element for each of the wsdl:part elements bound to the envelope's corresponding wssoap12:body element.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in /wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wssoap12:*[@style = 'rpc']]/wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	some \$mbody in ./soap12:Body satisfies some \$dbody in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation[@name = fn:local-name(\$mbody/*[1])]/wsdl:input/wssoap12:body satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = fn:local-name(\$mbody/*[1])]/[fn:local-name(.) = 'input'] satisfies some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = fn:tokenize(xsd:string(\$dopmsg/@message), ':') [fn:last()]] satisfies (not(wsdl:part) and not(\$mbody/*/*)) or (every \$mpart in \$dmesg/wsdl:part satisfies fn:count(\$mbody/*/*[fn:local-name(.) = \$mpart/@name]) eq 1

	or (\$body/@parts and not(contains(\$body/@parts, \$mpart/@name))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the soap12:Body element of request message is inconsistent with its description. The envelope does not contain exactly one part accessor element for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Diagnostic Data:	

Test Assertion:	BP1212b
Description:	The Envelope of response message contains exactly one part accessor element for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']]/ wsdl:operation [fn:string-join((@name, 'Response'),") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	some \$body in ./soap12:Body satisfies some \$body in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation[fn:string-join((@name, 'Response'),") = fn:local-name(\$body/*[1])]/wsdl:output/wsoap12:body satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[fn:string-join((@name, 'Response'),") = fn:local-name(\$body/*[1])/*[fn:local-name(.) = 'output'] satisfies some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = fn:tokenize(xsd:string(\$dopmsg/@message), '.')][fn:last()] and wsdl:part/@type] satisfies every \$mpart in \$dmesg/wsdl:part satisfies fn:count(\$body/*[fn:local-name(.) = \$mpart/@name]) eq 1 or (\$body/@parts and not(contains(\$body/@parts, \$mpart/@name)))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the soap12:Body element of response message is inconsistent with its description. The envelope does not contain exactly one part accessor element for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Diagnostic Data:	

Test Assertion:	BP1213a
Description:	The request message envelope has no element content in the soap12:Body element if the value of the parts attribute of the wsoap12:body is an empty string in the corresponding doc-literal description.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies (not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action))]]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./../@targetNamespace, if (not(fn:ends-with(\$opmsg/./../@targetNamespace,'/')) then '/' else "", \$opmsg/./../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else

	fn:concat(\$opmsg/./@name, 'Request'))) and (\$opBinding/./@name = \$opmsg/./@name)) else fn:true())]
Prerequisite:	some \$myOpBind in \$myOpBinding satisfies (\$myOpBind/./wsoap12:binding[not (@style = 'rpc')] and \$myOpBind/wsoap12:operation[not (@style = 'rpc')] and \$myOpBind/wsd:input/wsoap12:body[@parts and (@parts = " or @parts = ' ')])
Predicate:	count(/./soap12:Body/*) eq 0
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The request message soap12:Body element must be empty when, in the corresponding doc-literal description, the value of the parts attribute of wsoap12:body is an empty string.
Diagnostic Data:	SOAP envelope

Test Assertion:	BP1213b
Description:	The response message envelope has no element content in the soap12:Body element if the value of the parts attribute of the wsoap12:body is an empty string in the corresponding doc-literal description.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsd:message satisfies (not(\$message/wsd:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:binding/wsd:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:portType/wsd:operation/wsd:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:portType/wsd:operation/wsd:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./@name, ' ', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Response')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]
Prerequisite:	some \$myOpBind in \$myOpBinding satisfies (\$myOpBind/./wsoap12:binding[not (@style = 'rpc')] and \$myOpBind/wsoap12:operation[not (@style = 'rpc')] and \$myOpBind/wsd:input/wsoap12:body[@parts and (@parts = " or @parts = ' ')])
Predicate:	count(/./soap12:Body/*) eq 0
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The response message soap12:Body element must be empty when, in the corresponding doc-literal description, the value of the parts attribute of wsoap12:body is an empty string.
Diagnostic Data:	SOAP envelope

Test Assertion:	BP1214a
Description:	The request message envelope does not contain any part accessor elements if the value of the parts attribute of the wsoap12:body is an empty string in the corresponding rpc-literal description
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsd:message satisfies (\$message/wsd:part[1]/@type)]]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:binding [./wsoap12:*[@style = 'rpc']/wsdl:operation [@name = fn:local-name-from-

	QName(node-name(\$target/soap12:Body/*[1]))]
Prerequisite:	\$myOpBinding/wSDL:input/wsoap12:body[@parts and (@parts = " or @parts = ' ')]
Predicate:	count(/.soap12:Body/*/*) eq 0
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the request message soap12:Body element is inconsistent with its description. The envelope must not have any part accessor elements when, in the corresponding rpc-literal description, the value of the parts attribute of wsoap12:body is an empty string.
Diagnostic Data:	SOAP envelope

Test Assertion:	BP1214b
Description:	The response message envelope does not contain any part accessor elements if the value of the parts attribute of the wsoap12:body is an empty string in the corresponding rpc-literal description
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:/*[@style = 'rpc'] / wsdl:operation [fn:string-join((@name, 'Response'), ") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Prerequisite:	\$myOpBinding/wSDL:output/wsoap12:body[@parts and (@parts = " or @parts = ' ')]
Predicate:	count(/.soap12:Body/*/*) eq 0
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the response message soap12:Body element is inconsistent with its description. The envelope must not have any part accessor elements when, in the corresponding rpc-literal description, the value of the parts attribute of wsoap12:body is an empty string.
Diagnostic Data:	SOAP envelope

Test Assertion:	BP2113
Description:	When they contain references to message parts, the wsoap12:header, wsoap12:headerfault and wsoap12:fault elements only refer to wsdl:part element(s) that have been defined using the "element" attribute.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:header[@part] or ./wsoap12:headerfault[@part] or ./wsdl:fault/wsoap12:fault[@name]]
Predicate:	(every \$shhf in wsdl:operation//wsoap12:/*[local-name(.) = 'header' or local-name(.) = 'headerfault'] satisfies some \$msguse in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$shhf/../../@name]/*[local-name(.) = local-name(\$shhf/..)] satisfies every \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = local-name-from-QName(resolve-QName(xsd:string(\$msguse/@message), \$msguse))]/wsdl:part satisfies \$mpart/@element or \$shhf/@part and not(\$shhf/@part = \$mpart/@name)) and (every \$sf in wsdl:operation/wsdl:fault/wsoap12:fault satisfies some \$msguse in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$sf/../../@name]/wsdl:fault[@name = \$sf/@name] satisfies every \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = local-name-from-QName(resolve-QName(xsd:string(\$msguse/@message), \$msguse))]/wsdl:part satisfies \$mpart/@element)
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsoap12:header, wsoap12:headerfault or wsoap12:fault elements refer to wsdl:part element(s) that are not defined using only the "element" attribute.
Diagnostic	Defective wsdl:binding and wsdl:part elements.

Data:	
-------	--

Test Assertion:	BP2114
Description:	Every wsdl:part from each wsdl:message in the associated wsdl:portType is referenced either by the wsoap12:body, wsoap12:header, wsoap12:fault, or wsoap12:headerfault.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding
Predicate:	<p>some \$bing in self::node() satisfies every \$op in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType[local-name-from-QName(resolve-QName(xsd:string(\$bing/@type),\$bing)) = @name]/wsdl:operation satisfies some \$bop in \$bing/wsdl:operation[@name = \$op/@name] satisfies (</p> <p>(</p> <p>(\$bop/wsdl:input/wsoap12:body[not(@parts)]) or</p> <p>(every \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = local-name-from-QName(resolve-QName(xsd:string(\$op/wsdl:input/@message),\$op/wsdl:input))]/wsdl:part satisfies</p> <p>(</p> <p>fn:exists(fn:index-of(tokenize(\$bop/wsdl:input/wsoap12:body/@parts,'s+'), \$mpart/@name))</p> <p>or</p> <p>(some \$extra in \$bing/wsdl:operation/wsdl:*[wsoap12:*[local-name-from-QName(resolve-QName(xsd:string(@message),.) = \$mpart/./@name] satisfies \$mpart/@name = \$extra/@part))</p> <p>)</p> <p>)</p> <p>and</p> <p>(not(\$op/wsdl:output) or</p> <p>(\$bop/wsdl:output/wsoap12:body[not(@parts)]) or</p> <p>(every \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = local-name-from-QName(resolve-QName(xsd:string(\$op/wsdl:output/@message),\$op/wsdl:output))]/wsdl:part satisfies</p> <p>(</p> <p>fn:exists(fn:index-of(tokenize(\$bop/wsdl:output/wsoap12:body/@parts,'s+'), \$mpart/@name)) or</p> <p>(some \$extra in \$bing/wsdl:operation/wsdl:*[wsoap12:*[local-name-from-QName(resolve-QName(xsd:string(@message),.) = \$mpart/./@name] satisfies \$mpart/@name = \$extra/@part))</p> <p>)</p> <p>)</p> <p>and</p> <p>(not(\$op/wsdl:fault) or</p> <p>(every \$flt in \$op/wsdl:fault satisfies \$flt/@name = \$bop/wsdl:fault/wsoap12:fault/@name))</p> <p>)</p>
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:binding does not bind every wsdl:part of a wsdl:message in the wsdl:portType to which it refers to one of wsoap12:body, wsoap12:header, wsoap12:fault or wsoap12:headerfault.
Diagnostic Data:	{Defective wsdl:binding element}{message with part(s) unbound}

Test Assertion:	BP2115
Description:	An "element" attribute on any wsdl:part element refers to a global element declaration.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part/@element]
Predicate:	<p>every \$part in wsdl:part[@element] satisfies some \$ged in //xsd:schema/xsd:element satisfies ((</p> <p>\$ged/./@targetNamespace = namespace-uri-from-QName(resolve-QName(xsd:string(\$part/@element),\$part)) or (not (\$ged/./@targetNamespace) and not (namespace-uri-from-QName(resolve-QName(xsd:string(\$part/@element),\$part))))) and local-name-from-QName(resolve-QName(xsd:string(\$part/@element),\$part)) = \$ged/@name)</p>

Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:message element containing a wsdl:part element that uses the "element" attribute does not refer, via that attribute, to a global element declaration.
Diagnostic Data:	Defective wsdl:message element.

Test Assertion:	BP1012a
Description:	The content of the request message envelope matches the definition in the WSDL document for rpc-lit binding. The order of the part elements in the soap12:Body of the wired message, is same as that of the wsdl:partS, in the wsdl:message that describes it for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsoap12:*[@style = 'rpc']]/wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	some \$tbody in ./soap12:Body satisfies some \$dbody in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation[@name = fn:local-name(\$tbody/*[1])/wsdl:input/wsoap12:body satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = fn:local-name(\$tbody/*[1])/*[fn:local-name(.) = 'input'] satisfies some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name = fn:tokenize(xsd:string(\$dopmsg/@message), ':')[fn:last()] satisfies (not(wsdl:part) and not(\$tbody/*/*)) or (every \$tst in (for \$i in 1 to fn:count(\$tbody/*/*) return //wsdl:message/wsdl:part[not(\$tbody/@parts and not(fn:contains(\$tbody/@parts, @name))))][\$i]/@name = fn:local-name(\$tbody/*/*[\$i])) satisfies \$tst = fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the request message envelope does not match the wsdl:message definition for rpc/lit binding. The order of parts in soap12:Body does not match the order of wsdl:partS in wsdl:message that describes it for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Diagnostic Data:	Non-matching WSDL operation and envelope.

Test Assertion:	BP1012b
Description:	The content of the response message envelope matches the definition in the WSDL document for rpc-lit binding. The order of the part elements in the soap12:Body of the wired message, is same as that of the wsdl:partS, in the wsdl:message that describes it for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']]/ wsdl:operation [fn:string-join((@name, 'Response'),") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	some \$tbody in ./soap12:Body satisfies some \$dbody in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation[fn:string-join((@name, 'Response'),") = fn:local-name(\$tbody/*[1])/wsdl:output/wsoap12:body satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[fn:string-join((@name, 'Response'),") = fn:local-name(\$tbody/*[1])/*[fn:local-name(.) = 'output'] satisfies some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[@name =

	fn.tokenize(xsd:string(\$dopmsg/@message), ' ')[fn:last()] and wsdl:part/@type] satisfies every \$tst in (for \$i in 1 to fn:count(\$mbody/*/*) return //wsdl:message/wsdl:part[not(\$dbody/@parts and not(fn:contains(\$dbody/@parts,@name))][(\$i)/@name = fn:local-name(\$mbody/*/*[\$i])]) satisfies \$tst = fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the response message envelope does not match the wsdl:message definition for rpc/lit binding. The order of parts in soap12:Body does not match the order of wsdl:partS in wsdl:message that describes it for each of the wsdl:part elements bound to the envelope's corresponding wsoap12:body element.
Diagnostic Data:	Non-matching WSDL operation and envelope.

Test Assertion:	BP2208
Description:	The wsdl:operation element is either a WSDL request/response or a one-way operation (no Notification or Solicit-Response).
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation
Prerequisite:	BP2703
Predicate:	not(*[local-name-from-QName(node-name(.)) = 'output' and position() = 1])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	wsdl:operation was not a request/response or one-way operation.
Diagnostic Data:	Operation.

Test Assertion:	BP2010
Description:	name attributes of Operations are unique in the wsdl:portType definition
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType
Prerequisite:	BP2703
Predicate:	not(wsdl:operation[@name = preceding-sibling::*/@name]) and not(wsdl:operation[@name = following-sibling::*/@name])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	name attributes are not unique within the portType definition.
Diagnostic Data:	list of duplicate name(s) and of elements that use them.

Test Assertion:	BP2014
Description:	The parameterOrder attribute omits at most 1 part from an output wsdl:message.
Target:	/wsil:testLog/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@parameterOrder]
Predicate:	for \$op in self::node() return count(for \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/*wsdl:message[@name = \$op/wsdl:output/@message]/wsdl:part return index-of(tokenize(\$op/@parameterOrder,'s+'), \$mpart/@name)) ge count(/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/*wsdl:message[@name = \$op/wsdl:output/@message]/wsdl:part) - 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An operation associated with an rpc-literal binding has a parameterOrder attribute that omits more than 1 part.
Diagnostic Data:	{defective portType}{output wsdl:message}

Test Assertion:	BP2116
Description:	The wsdl:message element does not contain part elements that use both "type" and "element" attributes.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part]
Predicate:	not(wsdl:part[@type and @element])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:message element has at least one wsdl:part element that contains both type and element attributes.
Diagnostic Data:	Defective wsdl:message element.

Test Assertion:	BP2402
Description:	The wsdl:binding element has a wsoap12:binding child element.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding
Prerequisite:	BP2703
Predicate:	child::wsoap12:binding
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsdl:binding element does not use a wsoap12:binding element as defined in section "3 SOAP Binding." of the WSDL 1.1 specification.
Diagnostic Data:	wsdl:binding.

Test Assertion:	BP2403
Description:	The contained soabinding element has a "transport" attribute
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[wsoap12:binding]
Prerequisite:	BP2703 BP2402
Predicate:	wsoap12:binding[@transport]
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Transport attribute of the soabinding does not exist.
Diagnostic Data:	soabinding element.

Test Assertion:	BP2404
Description:	The contained soabinding element has a "transport" attribute, which has value: http://schemas.xmlsoap.org/soap/http.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[wsoap12:binding]
Prerequisite:	BP2703 BP2402 BP2403
Predicate:	wsoap12:binding[@transport='http://schemas.xmlsoap.org/soap/http']
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Transport attribute of the soabinding does not contain http://schemas.xmlsoap.org/soap/http.
Diagnostic Data:	soabinding element.

Test	BP2017
-------------	---------------

Assertion:	
Description:	The wsdl:binding is either an rpc-literal binding or a document-literal binding.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[wssoap12:binding]
Prerequisite:	BP2404
Predicate:	not(../wssoap12:body/@use != 'literal') and (count(../wssoap12:body) = count(../wssoap12:body/@use)) and ((not(../wssoap12:*/@style != 'rpc') and not(../wssoap12:operation[not(@style) and not(../wssoap12:binding/@style)])) or (not(../wssoap12:*/@style != 'document'))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The "style" attribute of an operation in soapbinding, does not have the same value of "document" or "rpc", as other operations of the binding.
Diagnostic Data:	defective soapbinding element.

Test Assertion:	BP2406
Description:	The use attribute has a value of "literal".
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[wssoap12:binding]
Prerequisite:	BP2703
Predicate:	not(../*[@use and @use != "literal"])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The use attribute of a wssoap12:body, wssoap12:fault, wssoap12:header and wssoap12:headerfault does not have value of "literal".
Diagnostic Data:	Defective wssoap12:body, wssoap12:fault, wssoap12:header, or wssoap12:headerfault elements.

Test Assertion:	BP2705
Description:	The wsdl:definitions in namespace http://schemas.xmlsoap.org/wsdl/ is valid in accordance with the wsi version of the wsdl 1.1 schema at http://ws-i.org/profiles/basic/1.1/wsdl-2004-08-24.xsd .
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions]
co-Target: metadata	\$target/@schemaValid
Predicate:	\$target/@schemaValid = fn:true()
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The description file contains a WSDL definition which does not conform to the schema located at http://ws-i.org/profiles/basic/1.1/wsdl-2004-08-24.xsd
Diagnostic Data:	Error message from the XML parser

Test Assertion:	BP2120b
Description:	Each operation referenced by the binding results in a unique wire signature.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [../wssoap12:*[@style = 'rpc']]
co-Target: imagesig	for \$bop_1 in \$target/wsdl:operation return fn:string-join((\$bop_1/wsdl:input/wssoap12:body/@namespace , \$bop_1/@name , for \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType [@name = fn:tokenize(xsd:string(\$target/@type), ':')fn:last()]/wsdl:operation[@name = \$bop_1/@name]/ wsdl:input return ((if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace, '/')) then '/' else "", \$opmsg/../../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/../../@name, 'Request')))), '')

Prerequisite:	BP2017
Predicate:	fn:count(\$imagesig) = fn:count(fn:distinct-values(\$imagesig))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A binding has operations that are not unique.
Diagnostic Data:	Defective wsdl:operation element(s).

Test Assertion:	BP2120a
Description:	Each operation referenced by the binding results in a unique wire signature.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [not(./wsoap12:*[@style = 'rpc'])]
co-Target: imagesig	for \$bop_1 in \$target/wsdl:operation return fn:string-join((for \$part in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [@name = (fn:tokenize(/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType [@name = fn:tokenize(xsd:string(\$target/@type), ':')][fn:last()]) /wsdl:operation[@name = \$bop_1/@name] / wsdl:input/@message, '.') [fn:last()]]) /wsdl:part[1] return (fn:namespace-uri-from-QName(fn:resolve-QName(xsd:string(\$part/@element), \$part)) , fn:local-name-from-QName(fn:resolve-QName(xsd:string(\$part/@element), \$part))) , for \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType [@name = fn:tokenize(xsd:string(\$target/@type), ':')][fn:last()]) /wsdl:operation[@name = \$bop_1/@name] / wsdl:input return ((if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace, '/')) then '/' else "", \$opmsg/../../@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/../../@name, 'Request')))),','))
Prerequisite:	BP2017
Predicate:	fn:count(\$imagesig) = fn:count(fn:distinct-values(\$imagesig))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A binding has operations that are not unique.
Diagnostic Data:	Defective wsdl:operation element(s).

Test Assertion:	BP2711
Description:	don't know why this is listed as Not testable.
Target:	///wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:service/wsdl:port
Predicate:	not(wsoap12:address/@location = preceding::wsdl:port/wsoap12:address/@location) and not(wsoap12:address/@location = following::wsdl:port/wsoap12:address/@location)
Reporting:	true=passed, false=failed
Prescription:	preferred
Error Message:	
Diagnostic Data:	

Test Assertion:	BP1011a
Description:	The content of the envelope matches the definition in the WSDL document. In case of a doc-lit binding, the child element of soap12:body is an instance of the global element declaration referenced by the corresponding wsdl:part. It is assumed the doc-lit binding has only 1 part. (NOTE: BP211 assumed prereq for the cotarget)
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (

	@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/././@targetNamespace, if (not(fn:ends-with(\$opmsg/././@targetNamespace, '/')) then ' ' else ' ', \$opmsg/././@name, ' ', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Request')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]]
Predicate:	count(soap12:Body/*) le 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the envelope associated with a doc-lit WSDL binding does not match the wsdl:message definition. the child element of soap12:body is not an instance of the global element declaration referenced by the corresponding wsdl:part.
Diagnostic Data:	Non-matching WSDL operation and envelope.

Test Assertion:	BP1011b
Description:	The content of the response message envelope matches the definition in the WSDL document. In case of a doc-lit binding, the child element of soap12:Body is an instance of the global element declaration referenced by the corresponding wsdl:part. It is assumed the doc-lit binding has only 1 part. (NOTE: BP211 assumed prereq for the cotarget)
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/././@targetNamespace, if (not(fn:ends-with(\$opmsg/././@targetNamespace, '/')) then ' ' else ' ', \$opmsg/././@name, ' ', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Response')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]]
Predicate:	count(soap12:Body/*) le 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the envelope does not match the wsdl:message definition. the child element of soap12:Body is not an instance of the global element declaration referenced by the corresponding wsdl:part.
Diagnostic Data:	Non-matching WSDL operation and envelope.

Test Assertion:	BP1111a
------------------------	----------------

Description:	The content of the request message envelope matches the definition in the WSDL document. In case of a doc-lit binding, the child element of soap12:body is an instance of the global element declaration referenced by the corresponding wsdl:part. It is assumed the doc-lit binding has only 1 part. (NOTE: BP211 assumed prereq for the cotarget)
Target:	<code>/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (. /wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]</code>
co-Target: myOpBinding	<code>/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .) = \$dmesg/@name) satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./././@targetNamespace, if (not(fn:ends-with(\$opmsg/./././@targetNamespace, '/')) then '/' else ' ', \$opmsg/./././@name, '/' , if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Request')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]</code>
Prerequisite:	some \$myOpBind in \$myOpBinding satisfies (not(\$myOpBind/./wsdl:body/@use != 'literal') and (count(\$myOpBind/./wsdl:body) = count(\$myOpBind/./wsdl:body/@use)) and \$myOpBind/./wsdl:binding[not (@style = 'rpc')] and \$myOpBind/wsdl:operation[not (@style = 'rpc')])
Predicate:	count(soap12:Body/*) le 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the request message envelope associated with a doc-lit WSDL binding does not match the wsdl:message definition. the child element of soap12:body is not an instance of the global element declaration referenced by the corresponding wsdl:part.
Diagnostic Data:	Non-matching WSDL operation and envelope.

Test Assertion:	BP1111b
Description:	The content of the response message envelope matches the definition in the WSDL document. In case of a doc-lit binding, the child element of soap12:Body is an instance of the global element declaration referenced by the corresponding wsdl:part. It is assumed the doc-lit binding has only 1 part. (NOTE: BP211 assumed prereq for the cotarget)
Target:	<code>/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or . /wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]</code>
co-Target: myOpBinding	<code>/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .) = \$dmesg/@name) satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./././@targetNamespace, if (not(fn:ends-with(\$opmsg/./././@targetNamespace, '/')) then '/' else ' ', \$opmsg/./././@name, '/' , if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Response')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]</code>
Prerequisite:	some \$myOpBind in \$myOpBinding satisfies (not(\$myOpBind/./wsdl:body/@use != 'literal') and (count(\$myOpBind/./wsdl:body) = count(\$myOpBind/./wsdl:body/@use)) and

	\$myOpBind/./wsoap12:binding[not (@style = 'rpc')] and \$myOpBind/wsoap12:operation[not (@style = 'rpc')]
Predicate:	count(soap12:Body/*) le 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The content of the response message envelope does not match the wsdl:message definition. the child element of soap12:Body is not an instance of the global element declaration referenced by the corresponding wsdl:part.
Diagnostic Data:	Non-matching WSDL operation and envelope.

Test Assertion:	BP2019
Description:	For doc/lit binding, the "namespace" attribute is not specified in any contained wsoap12:body, wsoap12:header, wsoap12::headerfault, wsoap12:fault elements
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[not(./wsoap12:*[@style = 'rpc'])]
Prerequisite:	BP2406
Predicate:	every \$wopbind in (\$target//wsoap12:body \$target//wsoap12:fault \$target//wsoap12:header \$target//wsoap12:headerfault) satisfies not(\$wopbind/@namespace)
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The binding is of style "document" and use "literal", and the "namespace" attribute is specified in some wsoap12:body, wsoap12:header, wsoap12::headerfault, wsoap12:fault element
Diagnostic Data:	Contained element with namespace attribute.

Test Assertion:	BP2020
Description:	The namespace attribute is specified on all wsoap12:body elements and the value of the namespace attribute is an absolute URI.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']]
Prerequisite:	BP2017
Predicate:	every \$wbody in ./wsoap12:body satisfies fn:count(fn:tokenize(\$wbody/@namespace,':')) > 1
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsoap12:body element does not have a namespace attribute, or the namespace attribute value is not an absolute URI.
Diagnostic Data:	wsoap12:body elements that failed the assertion

Test Assertion:	BP2117
Description:	The rpc-literal binding does not have a namespace attribute specified on a contained wsoap12:header, wsoap12:headerfault, and wsoap12:fault element.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']]
Prerequisite:	BP2017 BP2406
Predicate:	not(./wsoap12:fault[@namespace]) and not(./wsoap12:header[@namespace]) and not(./wsoap12:headerfault[@namespace])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	An rpc-literal binding has the namespace attribute specified on contained wsoap12:header, wsoap12:headerfault and wsoap12:fault elements.

Diagnostic Data:	{Defective wsdl:binding}{defective wsoap12:header, wsoap12:headerfault, or wsoap12:fault element}
-------------------------	---

Test Assertion:	BP2118
Description:	The list (or set) of wsdl:operation elements for the contained wsdl:binding is the same as the list of wsdl:operations for the referred wsdl:portType.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding
co-Target: myporttype	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType[@name = fn:local-name-from-QName(fn:resolve-QName(xsd:string(\$target/@type), \$target))]
Predicate:	(every \$operation in \$myporttype/wsdl:operation satisfies (some \$bop in \$target/wsdl:operation satisfies \$bop/@name = \$operation/@name)) and (count(wsdl:operation) = count(\$myporttype/wsdl:operation)) and not(wsdl:operation[@name = preceding-sibling::wsdl:operation/@name])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	A wsdl:binding does not have the same list of wsdl:operations as the wsdl:portType to which it refers.
Diagnostic Data:	{unmatching wsdl:binding element}{unmatching portType element}

Test Assertion:	BP2021
Description:	The wsdl:input element and wsdl:output element of each operation uses the attribute name "part" with a Schema type of "NMTOKEN" and does not use "parts", for both wsoap12:header elements and wsoap12:headerfault elements.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation[wsdl:input/wsoap12:header or wsdl:input/wsoap12:headerfault or wsdl:output/wsoap12:header or wsdl:output/wsoap12:headerfault]
Prerequisite:	BP2703
Predicate:	(not(wsdl:input/wsoap12:header) or (wsdl:input/wsoap12:header[@part] and not(wsdl:input/wsoap12:header[@parts]) and not(wsdl:input/wsoap12:header[contains(@part, ' ')]) and not(wsdl:input/wsoap12:header[contains(@part, ',')])))) and (not(wsdl:input/wsoap12:headerfault) or (wsdl:input/wsoap12:headerfault[@part] and not(wsdl:input/wsoap12:headerfault[@parts]) and not(wsdl:input/wsoap12:headerfault[contains(@part, ' ')]) and not(wsdl:input/wsoap12:headerfault[contains(@part, ',')])))) and (not(wsdl:output/wsoap12:header) or (wsdl:output/wsoap12:header[@part] and not(wsdl:output/wsoap12:header[@parts]) and not(wsdl:output/wsoap12:header[contains(@part, ' ')]) and not(wsdl:output/wsoap12:header[contains(@part, ',')])))) and (not(wsdl:output/wsoap12:headerfault) or (wsdl:output/wsoap12:headerfault[@part] and not(wsdl:output/wsoap12:headerfault[@parts]) and not(wsdl:output/wsoap12:headerfault[contains(@part, ' ')]) and not(wsdl:output/wsoap12:headerfault[contains(@part, ',')]))))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The wsdl:input element or wsdl:output element of an operation does not use the attribute name "part" with a Schema type of "NMTOKEN" for wsoap12:header elements or wsoap12:headerfault elements, or it uses "parts".
Diagnostic Data:	wsdl:input element or wsdl:output element of the defective operation.

Test Assertion:	BP2022
Description:	the name attribute is specified on the wsoap12:fault element.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation/wsdl:fault/wsoap12:fault
Predicate:	@name and @name != "
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Defective wsoap12:fault element: the name attribute is not specified on the wsoap12:fault element.
Diagnostic Data:	Defective wsoap12:fault element

Test Assertion:	BP2032
Description:	the name attribute that is specified on the wsoap12:fault element matches the value specified on the parent element wsd:fault.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:binding/wsd:operation/wsd:fault
Predicate:	@name = wsoap12:fault/ @name
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Defective wsoap12:fault element: the "name" attribute value does not match the value of the "name" attribute on the parent element wsd:fault.
Diagnostic Data:	Defective wsoap12:fault element

Test Assertion:	BP1005
Description:	The envelope has a wrapper element with a name equal to the name attribute on the wsd:operation element suffixed with string "Response".
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/ soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in /wsdl:definitions/wsd:message satisfies (\$message/wsd:part[1]/@type)]
co-Target: related- Request	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and (if (not(./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID)) then (@conversation = \$target/./../@conversation) else (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo [@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]))]/wsil:messageContents/soap12:Envelope[soap12:Body/*]
co-Target: myOpBindings	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:binding[./wsoap12:* [@style = 'rpc']/wsdl:operation[@name = fn:local-name(\$related- Request[1]/soap12:Body/*[1])]]
Predicate:	fn:concat(fn:local-name(\$related-Request[1]/soap12:Body/*[1]), 'Response') = fn:local- name(\$target/soap12:Body/*[1])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Wrapper element in the envelope does not have a value equal to the name attribute on the wsd:operation element suffixed with string "Response".
Diagnostic Data:	

Test Assertion:	BP1008a
Description:	The envelope has part accessor elements for parameters and return value, in no namespaces, but the descendants of these are namespace qualified as defined by the schema in which their types are defined.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body//soap12:Fault)] [some \$myenv in . satisfies some \$message in /wsdl:definitions/wsd:message satisfies (\$message/wsd:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsd:definitions/wsd:binding [./wsoap12:*[@style = 'rpc']/wsdl:operation[@name = fn:local-name-from- QName(node-name(\$target/soap12:Body/*[1]))]]
Predicate:	every \$accessor in \$target/soap12:Body/*/* satisfies (fn:namespace-uri-from- QName(fn:node-name(\$accessor)) = " or fn:namespace-uri-from-QName(fn:node-

	name(\$accessor)) = 'http://www.ws-i.org/testing/2008/02/log')
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The request envelope has part accessor elements for parameters and return value, within namespaces.
Diagnostic Data:	

Test Assertion:	BP1008b
Description:	The response envelope has part accessor elements for parameters and return value, in no namespaces.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/ @type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsdl:binding[@style = 'rpc']/ wsdl:operation [fn:string-join((@name, 'Response'), " ") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	every \$accessor in ./soap12:Body/*/* satisfies (fn:namespace-uri-from-QName(fn:node-name(\$accessor)) = " or fn:namespace-uri-from-QName(fn:node-name(\$accessor)) = 'http://www.ws-i.org/testing/2008/02/log')
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The response envelope has part accessor elements for parameters and return value, within namespaces.
Diagnostic Data:	

Test Assertion:	BP1755a
Description:	Each part accessor element in the envelope has a local name of the same value as the name attribute of the corresponding wsdl:part element.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/ @type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsdl:binding[@style = 'rpc']/ wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	(some \$mbody in ./soap12:Body satisfies every \$accessor in \$mbody/*/* satisfies some \$dbody in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [@name = fn:local-name(\$mbody/*[1])]/wsdl:input/wsdl:body satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation [@name = fn:local-name(\$mbody/*[1])]/*[fn:local-name(.) = 'input'] satisfies some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [@name = fn:tokenize(xsd:string(\$dopmsg/@message), ':') [fn:last()]] satisfies some \$mpart in \$dmesg/wsdl:part satisfies \$mpart/@name = fn:local-name(\$accessor))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The envelope referenced by an rpc-literal binding has a part accessor element with a local name that is not the same value as the name attribute of the corresponding wsdl:part element.
Diagnostic Data:	

	\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [wsdl:part{fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])}] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/././@targetNamespace, if (not(fn:ends-with(\$opmsg/././@targetNamespace,'/')) then '/' else '', \$opmsg/././@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Request')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]
Prerequisite:	\$myOpBinding/wsdl:input/wsoap12:header
Predicate:	some \$myOpBind in \$myOpBinding satisfies (every \$inputhdr in \$myOpBind/wsdl:input/wsoap12:header satisfies some \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [@name = fn:local-name-from-QName(fn:resolve-QName(xsd:string(\$inputhdr/@message), \$inputhdr))] /wsdl:part[@name = \$inputhdr/@part] satisfies some \$headerblk in \$target/soap12:Header/* satisfies (fn:resolve-QName(xsd:string(\$mpart/@element), \$mpart) = fn:node-name(\$headerblk)))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Envelope does not include all wsoap12:headers specified in the wsdl:input (of request) of its bound operation.
Diagnostic Data:	

Test Assertion:	BP1009b
Description:	The Envelope includes all wsoap12:headers specified in the wsdl:input (if request) or wsdl:output (if response) of the operation referred to by its wsdl:binding, and may also include headers that were not specified.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message[wsdl:part{fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])}] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:output satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/././@targetNamespace, if (not(fn:ends-with(\$opmsg/././@targetNamespace,'/')) then '/' else '', \$opmsg/././@name, '/', if(\$opmsg/@name) then \$opmsg/@name else fn:concat(\$opmsg/./@name, 'Response')))) and (\$opBinding/@name = \$opmsg/./@name)) else fn:true())]
Prerequisite:	\$myOpBinding/wsdl:output/wsoap12:header
Predicate:	some \$myOpBind in \$myOpBinding satisfies (every \$outputhdr in \$myOpBind/wsdl:output/wsoap12:header satisfies some \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [@name = fn:local-name-from-QName(fn:resolve-QName(xsd:string(\$outputhdr/@message), \$outputhdr))] /wsdl:part[@name = \$outputhdr/@part] satisfies some \$headerblk in \$target/soap12:Header/* satisfies (fn:resolve-QName(xsd:string(\$mpart/@element), \$mpart) = fn:node-name(\$headerblk)))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Envelope does not include all wsoap12:headers specified in the wsdl:input (if request) or wsdl:output (if response) of its bound operation.
Diagnostic	

Data:	
--------------	--

Test Assertion:	BP1009c
Description:	The Envelope includes all wsoap12:headers specified in the wsdl:input (for request) of the operation referred to by its wsdl:binding, and may also include headers that were not specified.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body//soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsoap12:*[@style = 'rpc']] / wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Prerequisite:	\$myOpBinding/wsdl:input/wsoap12:header
Predicate:	some \$myOpBind in \$myOpBinding satisfies (every \$inputhdr in \$myOpBind/wsdl:input/wsoap12:header satisfies some \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [@name = fn:local-name-from-QName(fn:resolve-QName(xsd:string(\$inputhdr/@message), \$inputhdr))] /wsdl:part[@name = \$inputhdr/@part] satisfies some \$headerblk in \$target/soap12:Header/* satisfies (fn:resolve-QName(xsd:string(\$mpart/@element), \$mpart) = fn:node-name(\$headerblk)))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Envelope does not include all wsoap12:headers specified in the wsdl:input (if request) or wsdl:output (if response) of its bound operation.
Diagnostic Data:	

Test Assertion:	BP1009d
Description:	The Envelope includes all wsoap12:headers specified in the wsdl:output (for response) of the operation referred to by its wsdl:binding, and may also include headers that were not specified.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body//soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsoap12:*[@style = 'rpc']] /wsdl:operation [fn:string-join((@name, 'Response'),") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Prerequisite:	\$myOpBinding/wsdl:output/wsoap12:header
Predicate:	some \$myOpBind in \$myOpBinding satisfies (every \$outputhdr in \$myOpBind/wsdl:output/wsoap12:header satisfies some \$mpart in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [@name = fn:local-name-from-QName(fn:resolve-QName(xsd:string(\$outputhdr/@message), \$outputhdr))] /wsdl:part[@name = \$outputhdr/@part] satisfies some \$headerblk in \$target/soap12:Header/* satisfies (fn:resolve-QName(xsd:string(\$mpart/@element), \$mpart) = fn:node-name(\$headerblk)))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Envelope does not include all wsoap12:headers specified in the wsdl:input (if request) or wsdl:output (if response) of its bound operation.
Diagnostic Data:	

Test Assertion:	BP1116a
------------------------	----------------

Description:	The action parameter on the Content-Type header if present has value equal to the value of the corresponding wsoap12:operation/@soapAction attribute, and an empty string ("") if there is no such attribute.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'action'] (: and @type = 'request' :) and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body[not (soap12:Fault)]] [some \$myenv in . satisfies (every \$message in //wsdl:definitions/wsdl:message satisfies ((not(\$message/wsdl:part[1]/@type)) and (\$myenv/soap12:Body/*[1] or \$myenv/soap12:Header/wsa:Action)))]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation [some \$opBinding in . satisfies (if (\$target/soap12:Body/*[1]) then (some \$dmesg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [wsdl:part[fn:resolve-QName(xsd:string(@element), .) = fn:node-name(\$target/soap12:Body/*[1])]] satisfies some \$dopmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input [fn:local-name-from-QName(fn:resolve-QName(xsd:string(@message), .)) = \$dmesg/@name] satisfies \$opBinding/@name = \$dopmsg/./@name) else fn:true()) and (if (\$target/soap12:Header/wsa:Action) then (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation/wsdl:input satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/@wsam:Action) then \$opmsg/@wsam:Action else fn:concat(\$opmsg/./@name, 'Request'))) and (\$opmsg/./@targetNamespace, if (not(fn:ends-with(\$opmsg/./@targetNamespace, 'Request')) then 'Request') else 'Request')))) and (\$opmsg/./@name = \$opmsg/./@name)) else fn:true())]
Prerequisite:	BP1006 BP1761
Predicate:	some \$myOpBind in \$myOpBinding satisfies ((\$myOpBind/wsoap12:operation/@soapAction = \$target/./wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter [fn:lower-case(@key) = 'action']/@value) or (not (\$myOpBind/wsoap12:operation/@soapAction) and \$target/./wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'action']/@value = "))
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The action parameter on the Content-Type header if present does not match wsoap12:operation/@soapAction attribute.
Diagnostic Data:	{All HTTP headers}{soapAction value from the soabinding}.

Test Assertion:	BP1116b
Description:	The action parameter on the Content-Type header if present has value equal to the value of the corresponding wsoap12:operation/@soapAction attribute, and an empty string ("") if there is no such attribute.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'action'] (: and @type = 'request' :) and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsoap12:*[@style = 'rpc']/wsdl:operation[@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Prerequisite:	BP1006 BP1761
Predicate:	some \$myOpBind in \$myOpBinding satisfies ((\$myOpBind/wsoap12:operation/@soapAction = \$target/./wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter [fn:lower-case(@key) = 'action']/@value) or (not (\$myOpBind/wsoap12:operation/@soapAction) and \$target/./wsil:httpHeaders/wsdl:contentTypeHeader/wsdl:parameter[fn:lower-case(@key) = 'action']/@value = "))
Reporting:	true=passed, false=failed
Prescription:	mandatory

Prescription:	mandatory
Error Message:	A Fault response message defined using a doc/lit binding has a wsa:Action value that is not equal to that specified in the WSDL. This may be due to the fact that the fault was generated by some infrastructure component (e.g. WS-Addressing, WS-RM, etc.). Verify that the fault was generated by some WS-* infrastructure component and not by the application.
Diagnostic Data:	Complete message.

Test Assertion:	BP1090b
Description:	For a Fault response message with a wsa:Action element, defined using a doc/lit binding, the following must hold: - The wsa:Action value in the soap12:Header must equal to that specified from the WSDL.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[(soap12:Body/soap12:Fault) and soap12:Header[wsa:Action]] [some \$myenv in . satisfies some \$message in /wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: relatedRequest	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/ wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]][1]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsoap12:*[@style = 'rpc']/wsdl:operation[@name = fn:local-name-from- QName(node-name(\$relatedRequest[1]/soap12:Body/*[1]))]]
Predicate:	(: correct wsa:Action value :) some \$detailQName in fn:node- name(\$target/soap12:Body/soap12:Fault/soap12:Detail/*[1]) satisfies some \$msgDef in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:message [wsdl:part[1][fn:resolve-QName(xsd:string(@element),.) = \$detailQName] satisfies (some \$opmsg in /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/ wsdl:operation[\$myOpBinding/ @name = @name]/ wsdl:fault[fn:local-name-from- QName(fn:resolve-QName(xsd:string(./@message),.) = \$msgDef/ @name) satisfies (\$target/soap12:Header/wsa:Action = (if (\$opmsg/ @wsam:Action) then \$opmsg/ @wsam:Action else fn:concat(\$opmsg/../../@targetNamespace, if (not(fn:ends-with(\$opmsg/../../@targetNamespace,'/'))) then '/' else "", \$opmsg/../../@name, '/', \$opmsg/../../@name, '/Fault', \$opmsg/ @name))))
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	A Fault response message defined using a rpc/lit binding has a wsa:Action value that is not equal to that specified in the WSDL. This may be due to the fact that the fault was generated by some infrastructure component (e.g. WS-Addressing, WS-RM, etc.). Verify that the fault was generated by some WS-* infrastructure component and not by the application.
Diagnostic Data:	Complete message.

Test Assertion:	BP2801
Description:	The value (either actual or computed) of wsam:Action attribute for the wsdl:input element contained in the target wsdl:operation of the wsdl:portType MUST equal the value of the soapAction attribute, if present, inside the wsoap12:operation element contained in the target wsdl:operation of the wsdl:binding.
Target:	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding/wsdl:operation[wsoap12:operation[@soapAction] and wsoap12:operation/ @soapAction ne " and (some \$bop in . satisfies /wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$bop/ @name]/wsdl:input[@wsam:Action])]
co-Target: myOpDefinition	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:portType/wsdl:operation[@name = \$target/ @name]
Predicate:	\$target/wsoap12:operation/ @soapAction = \$myOpDefinition/wsdl:input/ @wsam:Action

Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The value of wsam:Action attribute for the wsdl:input element contained in the target wsdl:operation of the wsdl portType, does not match the @soapAction value in this operation binding.
Diagnostic Data:	Operation.

Test Assertion:	BP1211a
Description:	Part accessor elements in the envelope do not have an xsi:nil attribute with a value of "1" or "true".
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'request' and not (./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')])] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body//soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding [./wsdl:operation [@style = 'rpc']]/wsdl:operation [@name = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	not (soap12:Body/*/*[attribute::xsi:nil = '1' or attribute::xsi:nil = fn:true()])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	Envelope referenced by an rpc-literal binding has part accessor elements with an xsi:nil attribute with a value of "1" or "true".
Diagnostic Data:	

Test Assertion:	BP1211b
Description:	Part accessor elements in the response envelope do not have an xsi:nil attribute with a value of "1" or "true".
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope[soap12:Body/* and not (soap12:Body/soap12:Fault)] [some \$myenv in . satisfies some \$message in //wsdl:definitions/wsdl:message satisfies (\$message/wsdl:part[1]/@type)]
co-Target: myOpBinding	/wsil:testLog/wsdl:descriptionFiles/wsdl:descriptionFile/wsdl:definitions/wsdl:binding[./wsdl:operation [@style = 'rpc']]/ wsdl:operation [fn:string-join((@name, 'Response'), " ") = fn:local-name-from-QName(node-name(\$target/soap12:Body/*[1]))]
Predicate:	not (soap12:Body/*/*[attribute::xsi:nil = '1' or attribute::xsi:nil = fn:true()])
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	response Envelope referenced by an rpc-literal binding has part accessor elements with an xsi:nil attribute with a value of "1" or "true".
Diagnostic Data:	

Test Assertion:	BP1149a
Description:	This assertion tests one aspect of R1149. Specifically, if a consumer sends a request message with a non-anonymous response EPR to an instance which advertises that it only supports anonymous EPRs, any response from that instance must be the SOAP fault described by Section 6.4.1.7 of the Web Services Addressing 1.0 - SOAP Binding . Other possible fault conditions, such as invalid addresses, are not tested for by this assertion.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or ./wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]]/wsil:messageContents/soap12:Envelope

	[/wsil:testLog/wsdl:descriptionFiles/wsdl:feature[@name="http://www.w3.org/ns/ws-policy/Policy"]/wsil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/Addressing"]/wsil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/AnonymousResponses" and @mode="required"]]
co-Target: related-Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]]
Predicate:	(if ((\$related-Request/soap12:Header/wsa:ReplyTo and not (\$related-Request/soap12:Header/wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous')) or (\$related-Request/soap12:Header/wsa:FaultTo and not (\$related-Request/soap12:Header/wsa:FaultTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous'))) then (not (\$target/soap12:Body/soap12:Fault) or (fn:ends-with(\$target/soap12:Body/soap12:Fault/soap12:Code/soap12:Value, 'Sender') and fn:ends-with(\$target/soap12:Body/soap12:Fault/soap12:Code/soap12:Subcode/soap12:Value, 'InvalidAddressingHeader') and fn:ends-with(\$target/soap12:Body/soap12:Fault/soap12:Code/soap12:Subcode/soap12:Subcode/soap12:Value, 'OnlyAnonymousAddressSupported'))) else fn:true())
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	The service requires the use of WS-Addressing with anonymous response EPRs. The response to a request with non-anonymous response EPRs must be the fault defined by Section 6.4.1.7 of the "WS-Addressing 1.0 - SOAP Binding" specification. Although the transmitted response is a fault, it does not match the definition in Section 6.4.1.7. This could be due to some other, overriding error such as a MustUnderstand fault. Inspect the response message and verify that this is the case. If not, the service is acting incorrectly by not transmitting the proper fault.
Diagnostic Data:	Complete message.

Test Assertion:	BP1149b
Description:	This assertion tests one aspect of R1149. Specifically, if a consumer sends a request message with a non-anonymous response EPR to an instance which advertises that it only supports anonymous EPRs, any response from that instance must be the SOAP fault described by Section 6.4.1.7 of the Web Services Addressing 1.0 - SOAP Binding . Other possible fault conditions, such as invalid addresses, are not tested for by this assertion.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or /wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope [/wsil:testLog/wsdl:descriptionFiles/wsdl:feature[@name="http://www.w3.org/ns/ws-policy/Policy"]/wsil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/Addressing"]/wsil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/AnonymousResponses" and @mode="required"]]
co-Target: related-Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]]
Predicate:	(if ((\$related-Request/soap12:Header/wsa:ReplyTo and not (\$related-Request/soap12:Header/wsa:ReplyTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous')) or (\$related-Request/soap12:Header/wsa:FaultTo and not (\$related-Request/soap12:Header/wsa:FaultTo/wsa:Address = 'http://www.w3.org/2005/08/addressing/anonymous'))) then (\$target/soap12:Body/soap12:Fault) else fn:true())
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The service requires the use of WS-Addressing with anonymous response EPRs. Any response to a request with non-anonymous response EPRs must be the fault defined by Section 6.4.1.7 of the "WS-Addressing 1.0 - SOAP Binding" specification. The transmitted response is either not a fault or does not match the definition of this fault.
Diagnostic	Complete message.

Data:	
--------------	--

Test Assertion:	BP1149c
Description:	This assertion tests one aspect of R1149. Specifically, if a consumer sends a request message with an anonymous response EPR to an instance which advertises that it only supports non-anonymous EPRs, any response from that instance must be the SOAP fault described by Section 6.4.1.8 of the Web Services Addressing 1.0 - SOAP Binding . Other possible fault conditions, such as invalid addresses, are not tested for by this assertion.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or /wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope [wsil:testLog/wsdl:descriptionFiles/wsdl:feature[@name="http://www.w3.org/ns/ws- policy/Policy"]/wsil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/Addressing"]/w sil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/NonAnonymousResponses" and @mode="required"]]
co-Target: related- Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/ wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	(if (not (\$related-Request/soap12:Header/wsa:ReplyTo) or \$related- Request/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous' or \$related-Request/soap12:Header/wsa:FaultTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous') then (not (\$target/soap12:Body/soap12:Fault) or (fn:ends- with(\$target/soap12:Body/soap12:Fault/soap12:Code/soap12:Value, 'Sender') and fn:ends- with(\$target/soap12:Body/soap12:Fault/soap12:Code/soap12:Subcode/soap12:Value, 'InvalidAddressingHeader') and fn:ends-with(\$target/soap12:Body/soap12:Fault/soap12:Code/soap12:Subcode/soap12:Subcode/soap12:Value, 'OnlyNonAnonymousAddressSupported'))) else fn:true())
Reporting:	true=passed, false=warning
Prescription:	mandatory
Error Message:	The service requires the use of WS-Addressing with non-anonymous response EPRs. The response to a request with anonymous response EPRs must be the fault defined by Section 6.4.1.8 of the "WS-Addressing 1.0 - SOAP Binding" specification. Although the transmitted response is a fault, it does not match the definition in Section 6.4.1.8. This could be due to some other, overriding error such as a MustUnderstand fault. Inspect the response message and verify that this is the case. If not, the service is acting incorrectly by not transmitting the proper fault.
Diagnostic Data:	Complete message.

Test Assertion:	BP1149d
Description:	This assertion tests one aspect of R1149. Specifically, if a consumer sends a request message with an anonymous response EPR to an instance which advertises that it only supports non-anonymous EPRs, any response from that instance must be the SOAP fault described by Section 6.4.1.8 of the Web Services Addressing 1.0 - SOAP Binding . Other possible fault conditions, such as invalid addresses, are not tested for by this assertion.
Target:	/wsil:testLog/wsdl:messageLog/wsdl:message [@type = 'response' or /wsil:messageContents/soap12:Envelope/soap12:Header/wsa:RelatesTo [not (@RelationshipType) or (@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply')]] /wsil:messageContents/soap12:Envelope [wsil:testLog/wsdl:descriptionFiles/wsdl:feature[@name="http://www.w3.org/ns/ws- policy/Policy"]/wsil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/Addressing"]/w sil:alternative/wsdl:feature[@name="http://www.w3.org/2007/05/addressing/metadata/NonAnonymousResponses" and @mode="required"]]
co-Target: related- Request	/wsil:testLog/wsdl:messageLog/wsdl:message[@type = 'request']/ wsil:messageContents/soap12:Envelope[soap12:Header/wsa:MessageID = \$target/soap12:Header/wsa:RelatesTo[@RelationshipType = 'http://www.w3.org/2005/08/addressing/reply' or not (@RelationshipType)]]
Predicate:	(if (not (\$related-Request/soap12:Header/wsa:ReplyTo) or \$related- Request/soap12:Header/wsa:ReplyTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous' or \$related-Request/soap12:Header/wsa:FaultTo/wsa:Address='http://www.w3.org/2005/08/addressing/anonymous') then

	(\$target/soap12:Body/soap12:Fault) else fn:true())
Reporting:	true=passed, false=failed
Prescription:	mandatory
Error Message:	The service requires the use of WS-Addressing with non-anonymous response EPRs. The response to a request with anonymous response EPRs must be the fault defined by Section 6.4.1.8 of the "WS-Addressing 1.0 - SOAP Binding" specification. The transmitted response is not a SOAP Fault message.
Diagnostic Data:	Complete message.

Appendix E. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Antonio Campanile, Bank of America
Robin Cover, OASIS
Doug Davis, IBM
Jacques Durand, Fujitsu
Pim van der Eijk, Sonnenglanz Consulting
Chet Ensign, OASIS
Joel Fleck II, Hewlett-Packard
Micah Hainline, Asynchrony Solutions, Inc.
Gershon Janssen, Individual
Ram Jeyaraman, Microsoft
Sarosh Niazi, Cisco Systems
Tom Rutt, Fujitsu Limited
Alessio Soldano, Red Hat

In addition, the Technical Committee thanks members of the WS-I Basic Profile Working Group whose work provided the foundation for this document, and in particular the former editorial team:

Robert Chumbley , IBM
Gilbert Pilz , Oracle
Prasad Yendluri, webMethods
Tom Rutt, Fujitsu

Appendix F. Revision History

Revision	Date	Editor	Changes Made
[WD01]	[3/6/2013]	[Tom Rutt]	[Moved referenced specs annex into Normative references]
[WD02]	[5/6/2013]	[jacques Durand]	Aligned references in specification body Added conformance clauses.
[WD06]	[3/17/2014]	[Tom Rutt]	Resolves all PR comments