Reliable Secure Profile Version 1.0

Committee Specification Draft 01

13 September 2013

Specification URIs

This version:

http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/csd01/ReliableSecureProfile-v1.0-csd01.doc (Authoritative)
http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/csd01/ReliableSecureProfile-v1.0-csd01.html
http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/csd01/ReliableSecureProfile-v1.0-csd01.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/ReliableSecureProfile-v1.0.doc (Authoritative)
http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/ReliableSecureProfile-v1.0.html
http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/ReliableSecureProfile-v1.0.pdf

Technical Committee:

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

Chair:

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

Editors:

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

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- Test Assertions: http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/csd01/testassertions/
- XML schemas: http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/csd01/schemas/

Related work:

This specification is related to:

- WS-I Reliable Secure Profile 1.0 Final Material 2010-11-09. http://www.ws-i.org/Profiles/ReliableSecureProfile-1.0-2010-11-09.html.

Abstract:

This document defines the WS-I Reliable Secure Profile Version 1.0 consisting of a set clarifications, refinements, interpretations and amplifications to a combination of non-proprietary Web services specifications in order to promote interoperability. In particular it is profiling the use of WS-SecureConversation, WS-ReliableMessaging and WS-MakeConnection. This profile extends either one of the Basic Profiles BP1.2 or BP2.0.

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 http://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 (http://www.oasis-open.org/committees/ws-brsp/ipr.php).

Citation format:

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

[ReliableSecureProfile-V1.0]

Reliable Secure Profile Version 1.0. 13 September 2013. OASIS Committee Specification Draft 01. http://docs.oasis-open.org/ws-brsp/ReliableSecureProfile/v1.0/csd01/ReliableSecureProfile-v1.0-csd01.html.

Notices

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 http://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 Introduction

This document defines the WS-I Reliable Secure Profile 1.0 (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 1 introduces the Profile, and explains its relationships to other profiles.

Section 2, "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.

1.1 Relationships to Other Profiles

This Profile is intended to be composed with the WS-I Basic Profile 1.2 **[BP1.2]** , WS-I Basic Profile 2.0 **[BP2.0]**, WS-I Basic Security Profile 1.0 **[BSP1.0]** and WS-I Basic Security Profile 1.1 **[BSP1.1]**. Composability of RSP with the previously mentioned profiles offers the following guarantee to users: conformance of an artifact to RSP does not prevent conformance of this artifact to these other profiles, and vice-versa.

Because the conformance targets defined for RSP may not match exactly the conformance targets for another profile, the following more precise definition of composability is assumed in this profile:

> *A profile P2 is said to be composable with a profile P1 if, for any respective pair of conformance targets (T2, T1) where T1 depends on T2 (see definition below), conformance of an instance of T2 to P2 does not prevent conformance of the related T1 instance(s) to P1, and vice-versa in case T2 depends on T1.*

A target T1 is said to depend on a target T2 if either:

- T2 and T1 are just different names for the same type of artifact (e.g. ENVELOPE in RSP and SOAP_ENVELOPE in BSP)
- or T2 is a specialization (or particular instance) of T1 (e.g. SECURE_ENVELOPE in BSP is a specialization of ENVELOPE in RSP)
- T2 is contained in T1 (e.g. SECURITY_HEADER in BSP is contained in ENVELOPE in RSP)
- more generally, an instance of T2 will restrict in some way the possible values - or behaviors - of T1 instances associated with it.

In order to conform to this profile (RSP):

- If SOAP 1.1 is being used, all requirements defined in BP 1.2 must be complied with.
- If SOAP 1.2 is being used, all requirements defined in BP 2.0 must be complied with.
- Implementations must conform to the WS-Addressing 1.0 - Core and SOAP Binding specifications, and if WSDL is used, the WS-Addressing 1.0 - Metadata specification.

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, such testability is not required. Preferably, testing is achieved in a non-intrusive manner (e.g., 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 a separate Test Assertions (TA) file that contains scripted (XPath 2.0) test assertions for assessing conformance of an endpoint to the RSP 1.0 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 of requirements in the Profile (i.e., those impacting conformance, as outlined in "Conformance Requirements") are 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.

Requirements can be considered to be namespace qualified, in such a way as to be compatible with QNames from Namespaces in XML [xmlNames]. If there is no explicit namespace prefix on a requirement's identifier (e.g., "R9999" as opposed to "bp10:R9999"), it should be interpreted as being in the namespace identified for this Profile.

Extensibility points in underlying specifications (see "Conformance Scope") 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.

- **soap11** - "http://schemas.xmlsoap.org/soap/envelope/"
- **soap12** - "http://www.w3.org/2003/05/soap-envelope"
- **wsdl** - "http://schemas.xmlsoap.org/wsdl/"

- **wsa** - "http://www.w3.org/2005/08/addressing"
- **wsrm** - "http://docs.oasis-open.org/ws-rx/wsrm/200702"
- **wsmc** - "http://docs.oasis-open.org/ws-rx/wsmc/200702"
- **wssc** - "http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
- **wst** - "http://docs.oasis-open.org/ws-sx/ws-trust/200512"
- **wsse** - "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
- **wsu** - "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"

1.5 Terminology

There are no specific terms defined by this Profile.

1.6 Profile Identification and Versioning

This document is identified by a name (in this case, Reliable Secure Profile) and a version number (here, 1.0). Together, they identify a particular *profile instance*.

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

[BP1.2] *Basic Profile 1.2*, OASIS Committee Specification Draft, May 2013. (TBD)

[BP2.0] *WS-I Basic Profile 2.0*, OASIS Committee Specification Draft, May 2013. (TBD)

[BSP1.0] M. McIntosh et al, “WS-I Basic Security Profile 1.0” , March 2007. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2007-03-30.html

[BSP1.1] *Basic Security Profile 1.1* , 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.

[RFC3987] M. Duerst et al., “Internationalized Resource Identifiers (IRIs)”, IETF RFC 3987, January 2005,. http://www.ietf.org/rfc/rfc3987.txt

[SOAP1.2-2] "SOAP Version 1.2 Part 2: Adjuncts (Second Edition)", W3C Recommendation, 27 April 2007,. http://www.w3.org/TR/2007/REC-soap12-part2-20070427/ (Section 6.2 SOAP Request-Response Message Exchange Pattern)

[WSAddrSoap] "WS-Addressing 1.0 – SOAP Binding”, W3C Recommendation, 9 May 2006,.http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/

[WSDL1.1] “Web Services Description Language (WSDL) 1.)", W3C Note, 15 March 2001. hhttp://www.w3.org/TR/2001/NOTE-wsdl-20010315 (Section 2.4 Port Types)

[WSMC1.1] "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

[WSRM1.2] "Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2", OASIS Standard, 2 February 2009,. http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.html

[WSSecCon1.4] "WS-SecureConversation 1.4", OASIS Standard, 2 February 2009,. http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.html

[WSS-Policy1.3] "WS-SecurityPolicy 1.3", OASIS Standard, 2 February 2009,. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html

[xmlNames] T. Bray et al, “Namespaces in XML 1.0” (Second Edition), August 2006. http://www.w3.org/TR/REC-xml-names/

1.8 Non-Normative References

There are no non normative references

2 Conformance

Conformance to the Profile is defined by adherence to the set of *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. 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 considered 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.

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

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) (from Basic Profile 1.1)
- **ENVELOPE** - the serialization of the soap:Envelope element and its content (from Basic Profile 1.1)
- **INSTANCE** - software that implements a wsdl:port or a uddi:bindingTemplate (from Basic Profile 1.1)
- **CONSUMER** - software that invokes an INSTANCE (from Basic Profile 1.1)
- **SENDER** - software that generates a message according to the protocol(s) associated with it (from Basic Profile 1.1)
- **RECEIVER** - software that consumes a message according to the protocol(s) associated with it (e.g., SOAP processors) (from Basic Profile 1.1)
- **MC-SENDER** - software that generates a message containing an EPR that uses the wsmc:MakeConnection Anonymous URI, and generates a MakeConnection message as defined by WS-MakeConnection 1.1 (from WS-MakeConnection 1.1)
- **MC-RECEIVER** - software that consumes a MakeConnection message as defined by WS-MakeConnection 1.1 (from WS-MakeConnection 1.1)
- **RMS** - RM Source as defined by WS-ReliableMessaging 1.2 (from WS-ReliableMessaging 1.2)
- **RMD** - RM Destination as defined by WS-ReliableMessaging 1.2 (from WS-ReliableMessaging 1.2)
- **RM-NODE** - an instance of either an RM Source or an RM Destination (as defined above)

2.3 Conformance Scope

The scope of the Profile delineates the technologies that it addresses; in other words, the Profile only attempts to improve interoperability within its own scope. Generally, the Profile's scope is bounded by the specifications referenced by it.

The Profile's scope is further refined 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.

Note that the Profile may still place requirements 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.

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.

The Profile's scope is defined by the referenced specifications in clause 1.7, as refined 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 2.2.

This Profile is an extension of the Basic Profile (1.2 or 2.0) and therefore conformance to this profile should always be mentioned or claimed in conjunction with a mention or claim about which one of the basic profile (V1.2 or V2.0) is used as foundation, as well as which level of conformance is used for the Basic Profile (CORE or HTTP_TRANSPORT).

In addition, to claim conformance to this Profile, a deployed INSTANCE target must support one or more of the WS-ReliableMessaging (WS-RM), WS-SecureConversation (WS-SC), or WS-MakeConnection (WS-MC) protocols, either individually or in some combination thereof, in a manner that conforms to the requirements set forth in this profile.

The four conformance clauses for RSP1.0 reflect the various ways this profile can use the underlying Basic Profile.

2.4.1 Core Conformance based on BP1.2

A conformance target (as defined above) is said to be conforming to this profile over BP1.2 at the “core” conformance level if both conditions below are satisfied:

(a) this target is conforming with BP1.2 at Core level,

(b) this target fulfills all the RSP1.0 requirements that are relevant to this target type and also relevant to the combination of WS specifications claimed to be supported among WS-ReliableMessaging (WS-RM), WS-SecureConversation (WS-SC), and WS-MakeConnection, except for those requirements that only apply in the context of using HTTP (in 5.2.2 “Binding to HTTP”).

Therefore, claims for “Core conformance based on BP1.2” should also mention the subset of WS specifications supported by the INSTANCE involved.

2.4.2 HTTP-transport Conformance based on BP1.2

A conformance target (as defined above) is said to be conforming to this profile over BP1.2 at the “HTTP-transport” conformance level if both conditions below are satisfied:

(a) this target is conforming with BP1.2 at HTTP-transport level,

(b) this target fulfills all the RSP1.0 requirements that are relevant to this target type and also relevant to the combination of WS specifications claimed to be supported among WS-ReliableMessaging (WS-RM), WS-SecureConversation (WS-SC), and WS-MakeConnection

Therefore, claims for “HTTP-transport conformance based on BP1.2” should also mention the subset of WS specifications supported by the INSTANCE involved.

2.4.3 Core Conformance based on BP2.0

A conformance target (as defined above) is said to be conforming to this profile over BP2.0 at the “core” conformance level if both conditions below are satisfied:

(a) this target is conforming with BP2.0 at Core level,

(b) this target fulfills all the RSP1.0 requirements that are relevant to this target type and also relevant to the combination of WS specifications claimed to be supported among WS-ReliableMessaging (WS-RM), WS-SecureConversation (WS-SC), and WS-

MakeConnection except for those requirements that only apply in the context of using HTTP (in 5.2.2 “Binding to HTTP”).

Therefore, claims for “Core conformance based on BP2.0” should also mention the subset of WS specifications supported by the INSTANCE involved.

2.4.4 HTTP-transport Conformance based on BP2.0

A conformance target (as defined above) is said to be conforming to this profile over BP2.0 at the “HTTP-transport” conformance level if both conditions below are satisfied:

(a) this target is conforming with BP2.0 at HTTP-transport level,

(b) this target fulfills all the RESP1.0 requirements that are relevant to this target type and also relevant to the combination of WS specifications claimed to be supported (among WS-ReliableMessaging (WS-RM), WS-SecureConversation (WS-SC), and WS-MakeConnection)

Therefore, claims for “HTTP-transport conformance based on BP2.0” should also mention the subset of WS specifications supported by the INSTANCE involved.

2.5 Claiming Conformance

Deployed instances can advertise their conformance to this Profile either through the use of mechanisms as described in Conformance Claim Attachment Mechanisms [claimAttachment] (see section 2.4.1) or through the use of mechanisms as described in Web Services Policy - Framework [WS-Policy 1.5] and Web Services Policy - Attachment [WS-Policy Attachment 1.5] specifications (see section 2.4.2). Prior agreements between partners on how Profile conformance is to be advertised or required might exist. When no such prior agreement exists and there is a need to advertise, the use of WS-Policy is RECOMMENDED over the use of the Conformance Claim Attachment Mechanisms.

2.5.1 Claiming Conformance using Conformance Claim Attachment Mechanisms

Mechanisms described in Conformance Claim Attachment Mechanisms [claimAttachment] can be used to advertise conformance to this profile, when the applicable Profile requirements associated with the listed targets have been met:

- **WSDL 1.1 Claim Attachment Mechanism for Web Services Instances** - MESSAGE, INSTANCE, RECEIVER, RMS, RMD

The conformance claim URIs are:

- To claim conformance to the requirements pertaining to WS-ReliableMessaging (Section 3 "Reliable Messaging") defined by this Profile, and to require the use of WS-ReliableMessaging when using the claiming endpoint:

  "http://ws-i.org/profiles/rsp/1.0/ws-rm/"
- To claim conformance to the requirements pertaining to WS-SecureConversation (Section 4 "Secure Conversation") defined by this Profile, and to require the use of WS-SecureConversation when using the claiming endpoint:

"http://ws-i.org/profiles/rsp/1.0/ws-sc/"

- To claim conformance to the requirements pertaining to WS-MakeConnection (Section 5 "Make Connection") defined by this Profile, and to require the use of WS-MakeConnection when using the claiming endpoint:

  "http://ws-i.org/profiles/rsp/1.0/ws-mc/"

Use of more than one or a combination of the above conformance claim URIs is allowed. When the claim URI "`http://ws-i.org/profiles/rsp/1.0/ws-sc/`" is combined with either "`http://ws-i.org/profiles/rsp/1.0/ws-rm/`" or "`http://ws-i.org/profiles/rsp/1.0/ws-mc/`", the requirements pertaining to such a combination as detailed in Section 6 "Secure Reliable Messaging" are also claimed to be conformed with.

The conformance claim URI to claim conformance to all requirements defined by this Profile is

"http://ws-i.org/profiles/rsp/1.0/"

NOTE: Because there is no requirement targeting WSDL constructs in RSP, a conformance claim attached to a wsdl:port only indicates conformance of the service instance to this Profile.

2.5.2 Claiming Conformance using WS-Policy and WS-PolicyAttachment

Mechanisms described in Web Services Policy - Framework [WS-Policy 1.5] and Web Services Policy - Attachment [WS-Policy Attachment 1.5] specifications can be used to advertise conformance to this profile.

The Profile defines the following policy assertion for this purpose:

```
<rsp:Conformant xmlns:rsp="http://ws-i.org/profiles/rsp/1.0/"/>
```

The presence of this assertion indicates that policy subject supports one or more of the WS-RM, WS-SC, or WS-MC protocols in a manner that conforms to RSP 1.0. This assertion only has meaning when used in a policy alternative that also contains at least one of wsrmp:RMAssertion, wsmc:MCSupported, or wsp:SecureConversationToken. The semantics of this assertion apply only to those protocols (WS-RM, WS-SC, or WS-MC) whose use is indicated by a policy assertion within the same policy alternative as this assertion. This assertion also requires that clients MUST use the effected protocols in a way that conforms to RSP 1.0. The absence of this assertion says nothing about RSP 1.0 conformance; it simply indicates the lack of an affirmative declaration of and requirement for RSP 1.0 conformance.

The `rsp: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 `rsp:Conformant` policy assertion MUST NOT be attached to a `wsdl:portType`.

Use of this policy assertion to claim conformance is highly encouraged.

This following example shows a policy expression that indicates/requires support for RSP 1.0 conformant WS-RM.

For example,

CORRECT:

```
<wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy"
            xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"
            xmlns:rsp="http://ws-i.org/profiles/rsp/1.0/">
  <wsrmp:RMAssertion>
    <wsp:Policy/>
  </wsrmp:RMAssertion>
  <rsp:Conformant/>
</wsp:Policy>
```

In the following example, the use of WS-RM is advertised as supporting/requiring conformance with RSP 1.0, but it is indeterminate whether or not the implementation of WS-MC supports or requires RSP 1.0 conformance.

For example,

CORRECT:

```
<wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy"
            xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"
            xmlns:rsp="http://ws-i.org/profiles/rsp/1.0/">
  <wsp:ExactlyOne>
    <wsp:All>
      <wsrmp:RMAssertion>
        <wsp:Policy/>
      </wsrmp:RMAssertion>
      <rsp:Conformant/>
    </wsp:All>
    <wsp:All>
      <wsmc:MCSupported/>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

3 Reliable Messaging

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

- Web Services Reliable Messaging 1.2 [WSRM1.2]
  Extensibility points:
  - E0001 - CreateSequence element and attribute extensions - Extending CreateSequence, via additional elements or attributes, is the primary mechanism for negotiating supplemental semantics to be applied to the requested and/or offered Sequence. Note this extensiblity point does not cover the pre-defined use of the /wsrm:CreateSequence/wsse:SecurityTokenReference element.
  - E0002 - CreateSequenceResponse element and attribute extensions - Extending CreateSequenceResponse, via additional elements or attributes, may be used to signal the acceptance of the supplemental semantics requested by the use of E0001 or it may be used in its own right to request or signal additional semantics to be applied to either requested and/or offered Sequence.
  - E0003 - CloseSequence element and attribute extensions - The CloseSequence element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the closure of the Sequence.
  - E0004 - CloseSequenceResponse element and attribute extensions - The CloseSequenceResponse element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the closure of the Sequence.
  - E0005 - TerminateSequence element and attribute extensions - The TerminateSequence element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the termination of the Sequence.
  - E0006 - TerminateSequenceResponse element and attribute extensions - The TerminateSequenceResponse element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the termination of the Sequence.
  - E0007 - Sequence element and attribute extensions - The Sequence header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the Sequence identified by the header.
  - E0008 - AckRequested element and attribute extensions - The AckRequest header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the request.
  - E0009 - SequenceAcknowledgment element and attribute extensions - The SequenceAcknowledgment header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the acknowledgment.
  - E0010 - SequenceFault element and attribute extensions - The SequenceFault element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the fault.
- Internationalized Resource Identifiers (IRIs) [RFC3987]
- Web Services Addressing 1.0 - SOAP Binding [WSAddrSoap]

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

3.1 WS-ReliableMessaging Support

3.1.1 Requiring WS-ReliableMessaging

As noted in Section 2, support for WS-ReliableMessaging by a specific service is optional. However, a service may require the use of WS-ReliableMessaging, in which case, for successful interaction with that service, a client will need to support it.

R0002 *If an endpoint requires the use of WS-ReliableMessaging, any* **ENVELOPE** *sent to that endpoint MUST conform to Section 3 of this Profile.* TESTABLERSP0002aRSP0002b

R0003 *If an endpoint requires or supports the use of WS-ReliableMessaging, the corresponding* **INSTANCE** *MUST behave in accordance with Section 3 of this Profile.* TESTABLERSP0003

Note that two RSP compliant web services implementations might both support the use of WS-ReliableMessaging yet fail to agree on a common set of features necessary to interact with one another. For example, a client might require the use of the "InOrder" Delivery Assurance, yet the service might not support this Delivery Assurance.

3.2 Use of Extension Elements and Attributes in Messages

The protocol elements defined by WS-ReliableMessaging contain extension points wherein implementations MAY add child elements and/or attributes.

3.2.1 Ignore Unknown Extension Elements

To ensure the ability to safely extend the protocol, it is necessary that adding an extension does not create the risk of impacting interoperability with non-extended implementations.

R0001 *A* **RECEIVER** *MUST NOT generate a fault as a consequence of receiving a message (e.g. wsrm:CreateSequence) that contains extension elements and/or attributes that it does not recognize, unless that extension is a SOAP Header with a mustUndestand="1" attribute. Any exceptions to this rule are clearly identified in requirements below or the specifications underlying the profile.* TESTABLE_SCENARIO_DEPENDENTRSP0001

While the extensibility points of the profiled specifications can be used, per R0001 they MUST be ignored if they are not understood. However if a SENDER wishes to ensure that the RECEIVER understands and will comply with any such extensions, they need to include a SOAP header, marked with `mustUnderstand="1"`, in the request message that requires adherence to the semantics of those extensions.

3.3 SOAP Version Considerations

In general, it is not expected that the service descriptions for applications that use WS-ReliableMessaging will include bindings of the WS-RM protocol itself. This being the case, there is some uncertainty about which version of SOAP should be used to carry Sequence Lifecycle Messages.

3.3.1 SOAP Version Selection for Sequence Lifecycle Messages

For messages that flow from the RMS to the RMD, the version(s) of SOAP used for Sequence Lifecycle Messages are constrained to the version(s) of SOAP that are supported by the target endpoint (i.e. the endpoint to which the client is attempting to reliably communicate). For example, if a client is attempting to communicate reliably to an endpoint who's service description indicates that it only supports SOAP 1.1, the RMS should only send Sequence Lifecycle Messages using SOAP 1.1. Sequence Lifecycle Response Messages (CreateSequenceResponse, TerminateSequenceResponse, and CloseSequenceResponse) should use the version of SOAP used by their corresponding request message (CreateSequence, TerminateSequence, and CloseSequence respectively); this applies to WS-RM fault messages as well. For messages that flow from the RMD to the RMS (SequenceAck messages with an empty body, unsolicited CloseSequence messages, and unsolicited TerminateSequence messages) this profile adheres to and expands upon WS-RM's statement that "The SOAP version used for the CreateSequence message SHOULD be used for all subsequent messages in or for that Sequence, sent by either the RM Source or the RM Destination".

R0900 *Unless otherwise specified (e.g. through some WSDL or WS-Policy designator), the* **RMD** *MUST send Sequence Lifecycle Messages destined to the CreateSequence/AcksTo EPR with the same SOAP version that was used in the CreateSequence message.* TESTABLERSP0900

R0901 *Unless otherwise specified (e.g. through some WSDL or WS-Policy designator), the* **RMS** *of an Offered Sequence MUST send Sequence Lifecycle Messages destined to the CreateSequence/Offer/Endpoint EPR with the same SOAP version that was used in the CreateSequence message.* TESTABLERSP0901

3.4 Targeting Sequence Lifecycle Messages

WS-ReliableMessaging is silent on where certain Sequence Lifecycle Messages (such as CreateSequence) should be sent.

3.4.1 CreateSequence Target

The WS-RM specification is silent on exactly where an RMS should send a CreateSequence message to establish a Sequence. This is true for the case of a client-side RMS creating a Sequence to carry request messages as well as the case of a server-side RMS creating a Sequence to carry response messages. This is an interoperability issue because, unless the respective RMS and RMD implementations agree on the expected target for CreateSequence messages, the intended recipient may not configure the necessary infrastructure (WS-RM message handlers, etc.) and the CreateSequence message may either cause a fault or be ignored.

R0800 *Baring some out of band agreement, an* **ENVELOPE** *carrying a CreateSequence message MUST be addressed to the same destination as one of the Sequence Traffic Message for that Sequence.* TESTABLERSP0800

This requirement applies equally to cases in which the first Sequence Traffic Message is addressed to a URI (as may happen when the target endpoint is retrieved from a WSDL document) or to an EPR (as may happen when the target endpoint is the wsa:ReplyTo address of the corresponding request message).

3.4.2 Use of the Offer Element

The use of the Offer element within a CreateSequence message is an optional feature of WS-ReliableMessaging. Using Offer avoids the exchange of CreateSequence and CreateSequenceResponse messages to establish a new sequence for response messages. However, WS-RM does not define a mechanism by which an RMS can determine if an Offer is desired by the RMD. This creates a potential interoperability issue in cases where an RMS that either doesn't wish to use or cannot support the use of Offer attempts to create a Sequence with an RMD that requires the use of Offer. To ensure interoperability, the Offer feature must be optional for both the initiator of the Sequence (the RMS) as well as the RMD.

Conversely, when an RMS includes an Offer within a CreateSeqence and the RMD rejects that Offer (e.g. if it only has input-only operations and concludes it has no need for the offered Sequence), if the RMD indicates this choice by faulting the CreateSequence the RMS has no programmatic means of determining that the fault was due to the presence of an Offer. To ensure interoperability in these cases, the RMD, rather than faulting the CreateSequence, must instead simply not accept the offered Sequence by not including an Accept element in the CreateSequenceResponse.

R0010 *An* **RMD** *MUST NOT fault a CreateSequence due to the absence of the Offer element.* TESTABLERSP0010

R0011 *An* **RMD** *MUST NOT fault a CreateSequence due to the presence of the Offer element.* TESTABLERSP0011

3.5 Sequence Identifiers

Under certain conditions it is possible for the `CreateSequence` or `CreateSequenceResponse` messages to be lost or delayed. Depending upon the timing of the attempts to resend such messages, it is possible to receive duplicate `CreateSequence` or `CreateSequenceResponse` messages (in fact, it is possible to receive duplicate messages even without retries). This creates the potential for `CreateSequence` and `CreateSequenceResponse` messages that contain duplicate Sequence Identifiers. Furthermore there are situations in which one party (RMS or RMD) may erroneously send a `CreateSequence` or `CreateSequenceResponse` message with a duplicate Sequence Identifier. Due to the crucial role of Sequence Identifiers in the WS-RM protocol, the handling of duplicate Sequence Identifiers needs to be further refined to prevent interoperability problems.

3.5.1 Duplicate Identifier in CreateSequenceResponse

Regardless of the causative circumstances, the existence of two, non-terminated Sequences with the same Identifier makes it difficult for the RMS to correctly function, therefore the RMS should take steps to prevent this condition.

R0700 *The* **RMS** *MUST generate a fault when it receives a* `CreateSequenceResponse` *that contains a Sequence Identifier that is the same as the Identifier of a non-terminated Sequence.* NOT_TESTABLECOM0700

Note that this requirement does not differentiate between duplicate Identifiers created by "the same" RMD or "different" RMDs; the simple fact that the RMS already has an active Sequence with the same Identifier is enough to trigger this requirement.

3.6 Sequence Termination

Termination of sequences must be done in a way to ensure that both the RMS and RMD share a common understanding of the final status of the sequence. The Profile places the following requirements on termination procedures:

3.6.1 Sequence Termination from the Destination

An RMS may need to get a final sequence acknowledgment, for supporting a particular delivery assurance. This is only possible after the sequence is closed and before it is terminated. When the termination is decided by the RMD, the RMS must also be made aware of this closure so that it can request a final acknowledgement.

R0200 *In the case where an* **RMD** *decides to discontinue a sequence, it MUST close the Sequence and MUST attempt to send a wsrm:CloseSequence message to the AcksTo EPR.* NOT_TESTABLECOM0200

3.6.2 Last Message Number

Among other benefits, the use of Sequence Message Numbers makes an RMD aware of gaps - messages it has not received - in a sequence. For this awareness to apply to messages missing from the end of a sequence the RMD must be aware of the highest message number sent.

R0210 *Any* **ENVELOPE** *from an RMS containing either a wsrm:CloseSequence or a wsrm:TerminateSequence element MUST also contain a wsrm:LastMsgNumber element if the Sequence in question contains at least one Sequence Traffic Message.* TESTABLERSP0210

There is a corner case for sequences in which no messages have been sent (i.e. empty sequences). In these cases it is permissable to omit wsrm:LastMsgNumber since there is no valid value for this element.

3.6.3 Sequence Lifecycle Independence

WS-ReliableMessaging is unclear about the relationship, if any, between the lifecycles of a Sequence and its corresponding Offered Sequence. Considering that such a relationship is not necessary for the proper functioning of the WS-RM protocol and that the existence of a such a relationship would create unnecessary and undesirable interdependencies between the RMS and the RMD, this profile makes the clarifying requirement that no such relationship exists.

R0220 *An* **RM-NODE** *(RMD or RMS) MUST NOT assume that the termination (or closure) of a Sequence implicitly terminates (or closes) any other Sequence.* NOT_TESTED

3.7 Sequence Faults

This Profile adds the following requirement to the handling of faults that are generated as the result of processing WS-RM Sequence Lifecycle messages.

3.7.1 WS-ReliableMessaging Faults

The use of WS-ReliableMessaging for faults that are themselves related to the WS-RM protocol is undefined and unlikely to be interoperable. Accordingly this profile prohibits the assignment of WS-RM fault messages to a WS-RM Sequence.

R0620 *An* **ENVELOPE** *that has wsrm:SequenceTerminated, wsrm:UnknownSequence, wsrm:InvalidAcknowledgement, wsrm:MessageNumberRollover, wsrm:CreateSequenceRefused, wsrm:SequenceClosed, or wsrm:WSRMRequired as the value of either the SOAP 1.2 /S:Fault/S:Code/S:Subcode/S:Value element or the /wsrm:SequenceFault/wsrm:FaultCode element MUST NOT contain a wsrm:Sequence header block.* TESTABLE RSP0620a RSP0620b

3.8 Sequence Assignment

WS-ReliableMessaging is silent on the mechanism for assigning messages (either request messages or response messages) to a particular Sequence. While this flexibility is beneficial from a general web services specification perspective, it creates some interoperability issues.

3.8.1 Reliable Response Messages

Given a scenario in which a consumer and a provider engage in a series of reliable request/response exchanges, it is important for the consumer and provider to have a consistent use of reliable messaging for response messages. From a reliable messaging perspective, all responses messages (both faults or non-faulting replies) are to be treated the same - meaning, either reliable messaging is enaged for both types of responses or it is turned off for both types of responses.

R0600 *An INSTANCE MUST NOT differentiate between faulting responses and non-faulting responses when determining whether to use WS-ReliableMessaging for a response message.* NOT_TESTED

3.8.2 Scope of an RM Node

WS-ReliableMessaging does not define the scope of an RM node other that to say that the scope is not restricted. For example, with respect to R0600 above, an Offered Sequence should be used to carry the response to any message sent over the Sequence corresponding to the CreateSequence request that included the Offer. However, it should not be assumed that the Sequence Traffic Messages carried over the Offered Sequence must be addressed to a particular response endpoint.

R0610 *The scope of an RM Node is an implementation choice that MUST NOT be constrained by the remote* **RM-NODE***. For example, the RMD MUST NOT constrain the values used by the RMS in the wsa:ReplyTo EPRs used by the RMS to be the same for all request messages (Sequence and Lifecycle messages).* TESTABLE RSP0610

Within this context the phrase "scope of an RM node" is defined as "the set of all EPRs that address a given RM node".

3.9 Retransmission of Messages

WS-ReliableMessaging protocol requires retransmission of messages. The Profile places the following restrictions and refinements on such retransmissions:

3.9.1 Retransmission of Unacknowledged Messages

To ensure reliable delivery of messages within a Sequence, it is necessary for the RMS to retransmit unacknowledged messages and for the RMD to accept them.

R0101 *An* **RMS** *MUST continue to retransmit unacknowledged messages until the Sequence is closed or terminated.* TESTABLERSP0101

R0102 *An* **RMD** *MUST accept unacknowledged messages until the Sequence is closed or terminated.* TESTABLERSP0102

Note: there are cases where it may be obvious that retransmitting a message is unlikey to result in an outcome that is any different from the previous, failed transmission(s). For example, in the case of HTTP, a 401 status code may indicate that access to an endpoint has been refused for the credentials that accompanied the request. Unless some action is taken to grant access to those credentials, retransmitting the request is likely to result in the same error and may cause negative side-effects such as the locking of an account due to "excessive failed login attempts".

3.9.2 Retransmission of Sequence Lifecycle Messages

WS-ReliableMessaging Section 2.1 defines the messages that affect the created/closing/closed/terminating state of a Sequence as "Sequence Lifecycle Messages". WS-RM is silent on what a SENDER (RMS or RMD) is expected to do when it either fails to send one of the messages or does not receive the corresponding response message (e.g. an RMS sends a CreateSequence message but does not receive a CreateSequenceResponse message).

R0110 *When a* **SENDER** *fails to successfully send a Sequence Lifecycle Message or it does not receive the corresponding response message (if one exists), it is RECOMMENDED that the SENDER attempt to resend the message. The frequency and number of these retries are implementation dependent.* NOT_TESTED

3.9.3 Message Identity

In cases where `wsa:MessageID` is being used, retransmission must not alter its value, because other headers (possibly occuring in other messages - such as `wsa:RelatesTo`) may rely on it for message correlation.

R0120 *For any two* **ENVELOPE***S that contain WS-RM Sequence headers in which the value of their wsrm:Identifier and wsrm:MessageNumber elements are equal, it MUST be true that neither of the envelopes contains a wsa:MessageID or that both messages contain a wsa:MessageID and the value of the wsa:MessageID elements are equal.* TESTABLERSP0120

3.10 Piggybacking

WS-ReliableMessaging allows for the addition of some WS-RM-defined headers to messages that are targeted to the same endpoint to which those headers are to be sent; a concept it refers to as "piggybacking". There are a number of interoperability issues with the practice of piggybacking.

3.10.1 Endpoint Comparison for Piggybacked SequenceAcknowledgment Headers

Because there is no standard mechanism for comparing EPRs, it is possible for different implementations to have dissimilar assumptions about which messages are and are not valid carriers for piggybacked SequenceAcknowledgement headers. For example, an implementation of the RMS may assume that the ReferenceParameters (if any) of the EPRs will be compared as part of the determination of whether a message is targeted to "the same" endpoint as the AcksTo endpoint. Meanwhile an implementation of the RMD may assume that a simple comparison of the Address IRIs is sufficient for making this determination. This creates the possibility for misdirected, dropped, and otherwise lost acknowledgements to the detriment and possible malfunctioning of the WS-RM protocol.

R0500 *An* **RMD** *MUST NOT piggyback a wsrm:SequenceAcknowledgement Header onto another message in cases where the destination property of the carrier message contains a wsa:Address IRI that differs (based on a simple string comparison) from the wsa:Address IRI of the wsm:AcksTo EPR corresponding to the wsrm:SequenceAcknowledgement.* TESTABLERSP0500

R0501 *In cases where the AcksTo EPR of a Sequence has an Address value equal to the WS-Addressing 1.0 Anonymous URI, the* **RMD** *MUST also limit piggybacking as described in section 3.9 of the WS-ReliableMessaging specification.* TESTABLERSP0501

These requirements establish a minimum baseline for an RMD to correctly piggyback SequenceAcknowledgement headers. Both endpoints should expect that at minimum, an RMD can compare address IRIs based on a simple string comparison algorithm, as indicated in the RFC 3987 section 5.3.1, in order to make the decision to piggyback or not. Individual RMD implementations may choose to consider and/or compare additional elements of the EndpointReference (e.g. the value of any ReferenceParameters elements).

3.10.2 Treatment of ReferenceParameters in AcksTo EPRs

There exists an interoperability problem for Sequences in which the AcksTo EPR contains ReferenceParameters. According to the processing rules defined by Web Services Addressing 1.0 - SOAP Binding [WSAddrSoap], the RMS should expect that any acknowledgements for the Sequence will be accompanied by the contents of the wsrm:AcksTo/wsa:ReferenceParameters promoted as headers in the message carrying that acknowledgement. However, in the case of piggybacked acknowledgments, the carrier message's [destination] EPR may contain Reference Parameters that conflict in some way with the wsrm:AcksTo/ReferenceParameters.

R0510 *If the algorithm used by the* **RMD** *to determine if a SequenceAcknowledgment can be piggybacked onto another message does not include a comparison of the value of the ReferenceParameters element (when present), then the RMD MUST NOT piggyback SequenceAcknowledgement headers for Sequences in which the AcksTo EPR contains ReferenceParameters.* TESTABLERSP0510

This requirement ensures any RMS implementation that includes ReferenceParameters in its AckTo EPRs of the following invariant: regardless of whether or not the acknowledgments for such Sequences are piggybacked, any message containing the SequenceAcknowledgement header(s) for such Sequences will also contain the AcksTo/wsa:ReferenceParameters in its SOAP headers. Note, this requirement applies equally to Sequences for which AcksTo/wsa:Address is anonymous and Sequences for which AcksTo/wsa:Address is not anonymous.

3.10.3 Preventing Piggybacked Acknowledgements

In situations where an RMD exercises the opportunity to piggyback most or all of the wsrm:SequenceAcknowledgement headers for a particular Sequence to an RMS which does not support the processing of piggybacked acknowledgments, it is likely that the operation of the WS-RM protocol will be severely impacted. This situation can be avoided if the RMS takes steps to ensure that the AcksTo EPRs for any Sequence's it creates are sufficiently unique as to cause the RMD to rule out the possibility of piggybacking acknowledgments for these Sequences.

R0520 *An* **RMS** *that does not support the processing of piggybacked SequenceAcknowledgement headers MUST differentiate the AcksTo EPRs for any Sequence's it creates from other EPRs.* NOT_TESTABLE

The term "differentiate" in the above requirement refers to the process of altering the information in the EPR in such a way as to cause the RMD to rule out the possibility of piggybacking acknowledgments for these Sequences while preserving the RMDs ability to connect to the proper transport endpoint. For example, suppose a particular instance of a web services stack maintains a generic, asynchronous callback facility at http://b2b.foo.com/async/AsyncResponseService. In general, all the EPRs minted by this instance for the purpose of servicing callbacks will have this URI as the value of their wsa:Address element. However, if this web services stack does not support the processing piggybacked acknowledgements, the use of this value in the AcksTo EPR creates the potential for the problem described above. The RMS implementation of this web services stack could fulfill this requirement by specifying http://b2b.foo.com/async/AsyncResponseService?p={*unique value*} as the address of the AcksTo EPR for any sequences it creates. Since each sequence has a "different" AcksTo EPR (as defined by R0500) from all the other services listening for callbacks, no RSP 1.0 compliant RMD will piggyback acknowledgements for these sequences, though each RMD (in the case of SOAP/HTTP) will correctly connect to http://b2b.foo.com and POST to /async/AsyncResponseService.

3.10.4 Conflicting Requirements for wsa:Action

Points (2) and (3) of Section 3.3 of the WS-ReliableMessaging state that:

> 2. When an Endpoint generates an Acknowledgement Message that has no element content in the SOAP body, then the value of the wsa:Action IRI MUST be: http://docs.oasis-open.org/ws-rx/wsrm/200702/SequenceAcknowledgement
> 3. When an Endpoint generates an Acknowledgement Request that has no element content in the SOAP body, then the value of the wsa:Action IRI MUST be: http://docs.oasis-open.org/ws-rx/wsrm/200702/AckRequested

However, this text does not take into account the possibility of piggybacking either of the above RM headers on messages with empty SOAP Bodys that contain wsa:Action values necessary to the proper processing of those messages. Such Envelopes could be the result of a WSDL that contains a doc-literal description where the value of the parts attribute of soap:body is an empty string. To clarify the expected behavior of WS-RM nodes under these circumstances, this profile makes the following requirement:

R0530 *In cases where the SequenceAcknowledgement or AckRequested header is piggybacked, then the wsa:Action value of the* **ENVELOPE** *MUST be as defined by Section 3.3 of the WS-ReliableMessaging specification if, and only if, the wsa:Action value has not been agreed upon by some other*

*mechanism (e.g. WSDL).*TESTABLERSP0530aRSP0530bRSP0530cRSP0530d

3.10.5 Use of the mustUnderstand Attribute

Since they are not allowed to interfere with the processing of messages, piggybacked SequenceAcknowledgement and AckRequested SOAP header blocks must not have the mustUnderstand attribute set to a value of true. However, when the SequenceAcknowledgement and AckRequested SOAP header blocks are sent on messages with an empty SOAP body element and a wsa:Action SOAP header block with a corresponding value of http://docs.oasis-open.org/ws-rx/wsrm/200702/SequenceAcknowledgement or http://docs.oasis-open.org/ws-rx/wsrm/200702/AckRequested (i.e. not piggybacked), implementations are advised to set the mustUnderstand attribute on the SequenceAcknowledgement and AckRequested SOAP header blocks to a value of true. This ensures that these headers are not ignored and avoids the resulting unnecessary retransmissions.

R0540 **SENDER***s MUST NOT set the value of mustUnderstand attribute on AckRequested and SequenceAcknowledgement SOAP header blocks to true ("1") when those headers are piggy-backed on outgoing MESSAGEs.*TESTABLERSP0540

4 Secure Conversation

The Profile includes the use of WS-SecureConversation to request and issue security tokens and to broker trust relationships.

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

- WS-SecureConversation 1.4 [WSSecCon1.4]
  Extensibility points:
  - E0011 - SecurityContextToken element and attribute extensions - The SecurityContextToken element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options that apply to the security context identified by the token.

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

All requirements in Section 4 apply only when WS-Security is used to secure a message. All requirements in Sections 4.1 through 4.5 apply only when WS-SecureConversation is used to secure a message.

4.1 WS-SecureConversation Support

4.1.1 Requiring WS-SecureConversation

As noted in Section 2, support for WS-SecureConversation by a specific service is optional. However, a service may require the use of WS-SecureConversation, in which case, for successful interaction with that service, a client will need to support it.

R1002 *If an endpoint requires the use of WS-SecureConversation, any* **ENVELOPE** *sent to that endpoint MUST conform to Section 4 of this Profile.* TESTABLERSP1002aRSP1002b

R1003 *If an endpoint requires or supports the use of WS-SecureConversation, the corresponding* **INSTANCE** *MUST behave in accordance with Section 4 of this Profile.*
NOT_TESTABLE

Note that two RSP compliant web services implementations might both support the use of WS-SecureConversation yet fail to agree on a common set of features necessary to interact with one another. For example, a service might require the use of a particular cipher suite that a client is not equipped to support.

4.2 Optionality of Operations

WS-SecureConversation (WS-SC) describes a set of bindings of WS-Trust for amending, renewing, and canceling security contexts. WS-SC does not define whether support for these bindings is mandatory or optional, for either clients or services. This creates the potential for interoperability problems due to differing expectations about such support. The following requirements clarify the optionality of the SCT Amend, Renew, and Cancel bindings.

4.2.1 Support for Amending Contexts

As of the date of this Profile, there are no known implementations of WS-SC that support the SCT Amend binding. CONSUMERS are advised to avoid its use unless they are certain that the target INSTANCE supports it.

R1004 *A* **CONSUMER** *SHOULD NOT send an ENVELOPE containing a* `wst:RequestSecurityToken` *in the SOAP Body and an action URI of* `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Amend`. TESTABLERSP1004

4.2.2 Support for Renewing Contexts

Support for the SCT Renew binding is elective for both CONSUMERs and INSTANCEs.

R1005 *An* **INSTANCE** *that acts as a WS-SC security token service MAY process ENVELOPEs containing a* `wst:RequestSecurityToken` *in the SOAP Body and an action URI of* `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Renew` *as per Section 5 of WS-SecureConversation.* NOT_TESTABLE

R1006 *An* **INSTANCE** *that acts as a WS-SC security token service but does not process ENVELOPEs containing a* `wst:RequestSecurityToken` *in the SOAP Body and an action URI of* `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Renew` *as per Section 5 of WS-SecureConversation MUST generate a fault with a [Subcode] value of "*`wsa:ActionNotSupported`*" (as per Section 6.4.4 of WS-Addressing 1.0 SOAP Binding) upon receiving a MESSAGE containing such an ENVELOPE.* TESTABLERSP1006aRSP1006b

4.2.3 Support for Canceling Contexts

Support for the SCT Cancel binding is mandatory for INSTANCEs and elective for CONSUMERs.

R1007 *An* **INSTANCE** *that acts as a WS-SC security token service MUST process ENVELOPEs containing a* `wst:RequestSecurityToken` *in the SOAP Body and an action URI of* `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Cancel` *as per Section 6 of WS-SecureConversation.* TESTABLERSP1007aRSP1007b

R1008 *When a* **CONSUMER** *concludes its use of a security context it SHOULD transmit an ENVELOPE containing a* `wst:RequestSecurityToken` *in the SOAP Body and an action URI of* `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Cancel` *to the WS-SC security token service that issued the SCT for that context.* TESTABLERSP1008

4.3 Unsupported Context Tokens

4.3.1 Unrecognized Extensions in a Security Context Token

During the establishment of a security context, it is possible for a participant to obtain an SCT that, for some reason, it chooses not to accept. One such possible reason is the presence of unrecognized extensions which, by definition, may indicate unknown and possibly harmful semantics. If the RECEIVER chooses to accept such an SCT, however, it must preserve this unrecognized content or nodes that understand and depend on this content may break.

R1000 *A* **RECEIVER** *MAY not accept an SCT due to unrecognized extensions in exception to R0001.* NOT_TESTEDRSP1000

R1001 *If a* **RECEIVER** *obtains an SCT containing content it does not recognize, the RECEIVER MUST preserve this unrecognized content in all subsequent use of the token.* TESTABLE_SCENARIO_DEPENDENTRSP1001

R1001 goes beyond R0001 (which does not require preservation of unrecognized content) to bring forward requirements from the WS-SecureConversation specification. R1000 has precedence over R1001 since an INSTANCE which faulted due to unrecognized content would not subsequently use the relevant token.

4.4 Demonstrating Proof of Possession

The following requirements describe how, for ENVELOPEs carrying a wst:RequestSecurityToken, the SOAP Body and crucial headers, specified in Section 4.5 and Section 6, must be signed.

4.4.1 Amending Contexts

R1100 *An* **ENVELOPE** *containing a wst:RequestSecurityToken in the SOAP Body and an action URI of http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Amend, MUST also contain a wsse:Security header with a ds:Signature child element that covers the SOAP Body and crucial headers as specified in Sections 4.5 and 6.* TESTABLERSP1100

R1101 *In an* **ENVELOPE**, *the signature referred to in R1100 MUST be created using the key associated with the security context that is being amended.* NOT_TESTABLECOM1101

4.4.2 Renewing Contexts

R1110 *An* **ENVELOPE** *containing a wst:RequestSecurityToken in the SOAP Body and an action URI of http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Renew, MUST also contain a wsse:Security header with a ds:Signature child element that covers the SOAP Body and crucial headers as specified in Sections 4.5 and 6.* TESTABLERSP1110

R1111 *In an* **ENVELOPE**, *the signature referred to in R1110 MUST be created using the key associated with the security context that is being renewed.* NOT_TESTABLE

4.4.3 Cancelling Contexts

R1120 *An* **ENVELOPE** *containing a wst:RequestSecurityToken in the SOAP Body and an action URI of http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Cancel, MUST also contain a wsse:Security header with a ds:Signature child element that covers the SOAP Body and crucial headers as specified in Sections 4.5 and 6.* TESTABLERSP1120

R1121 *In an* **ENVELOPE**, *the signature referred to in R1120 MUST be created using the key associated with the security context that is being canceled.* NOT_TESTABLE

4.5 Claims Re-Authentication

4.5.1 Re-Authenticating Claims

As per section 5 of the WS-SecureConversation specification, the request to renew a security context must include the re-authentication of the context's original claims. It is recommended, but not required, that the claims re-authentication be done in the same manner as the original token issuance request. This creates the potential for some implementations of WS-SecureConversation to attempt claims re-authentication in a manner different than the original token issuance request, to the obvious detriment of both interoperability and security.

R1200 *When a* **SENDER** *makes a request to renew a security context, it MUST re-authenticate the original claims in the same way as in the original token issuance request.* TESTABLERSP1200

4.6 Referencing Security Context Tokens

4.6.1 Associating a Security Context

Section 8 of WS-SecureConversation states that references to an SCT from within a `wsse:Security` header, a `wst:RequestSecurityToken` element, or a `wst:RequestSecurityTokenReponse` element may be either message dependent or message independent. However, references to SCTs from outside a `wsse:Security` header (or an RST, or an RSTR) must be message independent. In order to improve interoperability, this profile includes the following requirement:

R1300 *A* **RECEIVER** *MUST support both message dependent and message independent references to a wssc:SecurityContextToken from within a wsse:Security header, a wst:RequestSecurityToken element, or a wst:RequestSecurityTokenReponse element.* TESTABLERSP1300

4.6.2 Derived Token References to Security Contexts

Section 7 of the WS-SecureConversation specification describes a mechanism for using keys derived from a shared secret for signing and encrypting the messages associated with a security context. The `wssc:DerivedKeyToken` element is used to express these derived keys. WS-SC states that the `/wssc:DerivedKeyToken/wsse:SecurityTokenReference` element SHOULD be used to reference the `wssc:SecurityContextToken` of the security context who's shared secret was used to

derive the key. This creates an interoperability issue because it leaves open the possibility for a derived key to either lack any relationship between the shared secret or for this relationship to be expressed by some mechanism other than a wsse:SecurityTokenReference.

R1310 *When an* **ENVELOPE** *contains a wssc:DerivedKeyToken, the wsse:SecurityTokenReference element MUST be used to reference the wssc:SecurityContextToken of the security context from which they key is derived.* TESTABLERSP1310

To properly and interoperably process derived keys it is necessary to relate the key to the shared secret from which it is derived. There are no alternatives to using wsse:SecurityTokenReference's that are consistent with WS-Security.

4.7 Addressing Headers

4.7.1 Protecting Addressing Headers

Since the semantics of the WS-SecureConversation protocol are dependent upon the value of various WS-Addressing headers, ensuring the proper functioning of WS-SecureConversation requires protecting the integrity of these headers. These requirements are not specific to the use of WS-SecureConversation. They also apply whenever WS-Security is being used in conjunction with WS-Addressing.

R1400 *When present in an* **ENVELOPE** *in which the SOAP Body in that ENVELOPE is signed, each of the following SOAP header blocks MUST be included in a signature:* `wsa:To`, `wsa:From`, `wsa:ReplyTo`, `wsa:Action`, `wsa:FaultTo`, `wsa:MessageId`, `wsa:RelatesTo`. TESTABLERSP1400

R1401 *In an* **ENVELOPE**, *the signature(s) referred to in R1400 MUST be coupled cryptographically (e.g. share a common signature) with the message body.* TESTABLERSP1401

R1402 *When present in an* **ENVELOPE** *in which the SOAP Body in that ENVELOPE is signed, SOAP Header blocks with the wsa:isReferenceParameter attribute MUST be included in a signature for their designated SOAP role.* TESTABLERSP1402

R1403 *In an* **ENVELOPE**, *the signature(s) referred to in R1402 MUST be coupled cryptographically (e.g. share a common signature) with the message body.* TESTABLERSP1403

5 Make Connection

The Profile includes the use of WS-MakeConnection to transfer messages using a transport-specific back-channel.

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

- Web Services Make Connection 1.1 [WSMC1.1]
  Extensibility points:
  - E0012 - MakeConnection element and attribute extensions - The MakeConnection element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options that apply to the request.
  - E0013 - Attribute extensions to the MakeConnection Address - The /wsmc:MakeConnection/wsmc:Address element may be extended via additional attributes to indicate the use of supplemental semantics or options that apply to this instance of the MakeConnection Anonymous URI
  - E0014 - MessagePending element and attribute extensions - The MessagePending header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the indication of pending messages.
- SOAP Version 1.2 Part 2: Adjunct (Second Edition) [SOAP1.2-2], Section 6.2 SOAP Request-Response Message Exchange Pattern
- Web Services Description Language (WSDL) 1.1 [WSDL1.1], Section 2.4 Port Types

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

The requirements and supporting text in this section make use of the following terms:

- **non-addressable client** - a client deployed on a host that cannot accept incoming connections from the services to which it transmits requests. Examples include clients that are deployed on hosts behind a firewall or network address translation (NAT).
- **addressable client** - a client deployed on a host that is capable of accepting incoming connections from the services to which it transmits requests.

5.1 WS-MakeConnection Support

5.1.1 Requiring WS-MakeConnection

As noted in Section 2, support for WS-MakeConnection by a specific service is optional. However, a service may require the use of WS-MakeConnection, in which case, for successful interaction with that service, a client will need to support it.

R2011 *If an endpoint requires the use of WS-MakeConnection, any response EPRs in an* **ENVELOPE** *transmitted to this endpoint MUST use either an instance of the MakeConnection Anonymous URI, the WS-Addressing anonymous URI (`http://www.w3.org/2005/08/addressing/anonymous`), or the WS-Addressing none URI (`http://www.w3.org/2005/08/addressing/none`) in their `wsa:Address` element.* TESTABLERSP2011aRSP2011b

5.1.2 Honoring EPRs with the MakeConnection Anonymous URI

R2012 *If an endpoint supports the use of WS-MakeConnection, the* **INSTANCE** *corresponding to that endpoint MUST NOT generate a fault due to the use of the* `wsmc:MakeConnection` *message.* TESTABLERSP2012

R2013 *If an endpoint supports the use of WS-MakeConnection, the* **INSTANCE** *corresponding to that endpoint MUST NOT generate a fault due to the use of a MakeConnection Anonymous URI in the* `wsa:Address` *element on any response EPRs in a request message.* TESTABLERSP2013

R2014 *If an endpoint requires the use of WS-MakeConnection, any* **MESSAGE** *transmitted from this endpoint MUST be transmitted over a connection that is associated with either an instance of the WS-MakeConnection Anonymous URI or the WS-Addressing anonymous URI (*`http://www.w3.org/2005/08/addressing/anonymous`*).* TESTABLERSP2014aRSP2014b

The association referred to in R2014 can be established by either a request message that carries response EPRs that use instances of the MakeConnection Anonymous URI, a request message that carries response EPRs that use instances of the WS-Addressing Anonymous URI, or a `wsmc:MakeConnection` message.

5.2 Guidance On the Use of MakeConnection

This section describes how, when wsmc:MakeConnection is used, WSDL input and output messages correspond to SOAP envelopes containing a request or a response sent over HTTP.

5.2.1 Action Values

The WS-MakeConnection specification, while not formally requiring the use of WS-Addressing headers, neglects to mention what the wsa:Action and soapAction URIs should be - when needed.

R2030 *If an* **ENVELOPE** *contains a wsmc:MakeConnection element as the child of the SOAP Body, the* `wsa:Action` *header, if present, MUST contain the value "http://docs.oasis-open.org/ws-rx/wsmc/200702/MakeConnection".* TESTABLERSP2030

R2031 *If a* **MESSAGE** *contains a SOAP 1.1 envelope with the wsmc:MakeConnection element as the child of the Body, the HTTP* `SOAPAction` *header, if present and not equal to the value of "" (empty string), MUST contain the value "http://docs.oasis-open.org/ws-rx/wsmc/200702/MakeConnection".* TESTABLERSP2031

R2032 *If a* **MESSAGE** *contains a SOAP 1.2 envelope with the wsmc:MakeConnection element as the child of the Body, the* `action` *parameter of the HTTP* `Content-Type` *header, if present, MUST contain the value "http://docs.oasis-*

open.org/ws-rx/wsmc/200702/MakeConnection". TESTABLERSP2032

5.2.2 Binding to HTTP

Consider the case of a non-addressable client *NA* invoking an addressable service *B* that supports a WSDL request-response operation. A possible request-response exchange involving wsmc:MakeConnection might take the following form:

1. Through some mechanism, *NA* is provided the EPR for *B*.
2. *NA* sends a SOAP request message (contained in the entity body of an HTTP request) to *B*. The SOAP request message corresponds to and is described by the input message in the WSDL request-response operation supported by *B*.
3. If *B* chooses not to send the application response on the current back channel, then *B* sends an HTTP response (via the HTTP back channel) with a status code of "202 Accepted". No SOAP envelope is included as part of this HTTP response.
4. *NA* sends a MakeConnection request message (in the entity body of an HTTP request) to *B* using the same MakeConnection anonymous URI identifying *NA* as in the wsa:ReplyTo addressing property contained in the SOAP envelope of the original request message.
5. *B* sends a SOAP response message (contained in the entity body of an HTTP response) via the HTTP back channel. The SOAP response message corresponds to and is described by the output message in the WSDL request-response operation supported by *B*. If the HTTP response in step 5 does not contain a SOAP envelope, and if there is no failure, then the HTTP response must not contain an entity body and the status code must be 202.

Note: If the HTTP response from step 5 above does not contain a response message corresponding to the output message in the WSDL request-response operation, *NA* repeats step 3 above until a response message corresponding to the output message in the WSDL request-response operation is retrieved from *B*, as described in step 4.

Now consider the case of an addressable client *A* invoking a non-addressable service *NB*. Another possible message exchange involving wsmc:MakeConnection might take the following form:

1. Through some mechanism, *A* is provided an EPR for *NB* - this EPR uses an instance of a MakeConnection anonymous URI that identifies *NB*, and *NB* is provided the EPR for *A*.
2. *NB* sends a MakeConnection request message (contained in the entity body of an HTTP request) to *A* using the same MakeConnection anonymous URI identifying *NB* as in the EPR for *NB*.
3. *A* sends a SOAP request message (contained in the entity body of the HTTP response) via the back channel. This SOAP request message corresponds to and is described by the input message in the WSDL operation supported by *NB*.
4. In the case of a request-response operation, *NB* sends a SOAP response message (contained in the entity body of a new HTTP request) to *A*. This SOAP response message corresponds to and is described by the output message in the WSDL request-response operation supported by *NB*.
5. *A* sends an HTTP response via the back channel with a status code of 202 Accepted. No SOAP envelope is included as part of the HTTP response.

Note: In step 3 above, if *NB* encounters an infrastructure level fault resulting from the processing the SOAP request message (that corresponds to and is described by the input message in the WSDL operation supported by *NB*), *NB* will send the fault to *A* via a separate HTTP request. Notice, this fault message replaces the output message in the WSDL request-response operation, if any, supported by *NB*.

A non-addressable endpoint may use wsmc:MakeConnection in a SOAP envelope to obtain any pending messages from an endpoint.

The MakeConnection specification does not mandate how long an MCReceiver needs to wait for an outgoing message to be generated - this is left as an implementation choice. For example, in some environments if there is no message ready to be sent back to the MCSender then returning an HTTP 202

immediately might be appropriate, while in some other cases waiting a certain period of time might improve performance with respect to network traffic. Either case could occur.

When the SOAP request-response MEP is in use and the client is non-addressable the general rules for binding SOAP envelopes to HTTP requests messages (as described by the Basic Profile) apply. SOAP envelopes, that are described by the input message of the WSDL operations supported by a service, are bound to HTTP request messages. SOAP envelopes, that are described by the output message of the WSDL operations supported by a service, are bound to HTTP response messages. For non-addressable services the situation is reversed; the SOAP envelopes, that are described by the input message of the WSDL operations supported by the service, are bound to HTTP response messages and SOAP envelopes, that are described by the output message of the WSDL operations supported by the service, are bound to HTTP request messages.

The following requirements extend the requirements defined in Basic Profile:

R2004 *When the wsa:ReplyTo addressing property of a request message (SOAP envelope included in the HTTP entity body of the HTTP request) described by the input message of a WSDL request-response operation is set to a MakeConnection anonymous URI, the corresponding response* **MESSAGE** *(SOAP envelope included in the HTTP entity body of the HTTP response) described by the WSDL output message of the same WSDL request-response operation MUST be sent as an HTTP response to either the HTTP request that carried the WSDL input message, or to the HTTP request that carried a wsmc:MakeConnection message with the correct MakeConnection anonymous URI.* TESTABLERSP2004aRSP2004b

R2005 *Any* **MESSAGE** *resulting from the processing of a SOAP envelope included in the HTTP entity body of the HTTP response, if transmitted, MUST be sent via a new HTTP request.* TESTABLERSP2005

5.2.3 Transmission of MakeConnection Faults

Section 3.2 of the WS-MakeConnection specification states that there is no reply to the MakeConnection message and therefore section 3.4 ("Formulating a Reply Message") of the WS-Addressing specification is not used. This requires some clarification with respect to Fault processing. The MakeConnection message is, by definition, a one-way message. Therefore, if during the processing of the MakeConnection message a Fault is generated that is related to the processing of MakeConnection message itself, that Fault message is to be treated like any other Fault related to a one-way message; that is, if the Fault message is transmitted then it will follow the rules defined by section 3.4 of the WS-Addressing specification. Note that this is different from the use of the MakeConnection protocol to transmit a Fault message - those messages are not replies to the MakeConnection message and section 3.4 would not apply.

R2050 *If, when processing a MakeConnection message, an* **MC-RECEIVER** *generates a fault related to the MakeConnection message (e.g. a wsmc:UnsupportedSelection or a wsmc:MissingSelection Fault) the transmission of that Fault MUST adhere to the rules as defined by section 3.4 of the WS-Addressing specification.* TESTABLERSP2050

5.3 MakeConnection Addressing

In section 3.1 of the WS-MakeConnection specification the WS-MC Anonymous URI is defined to uniquely identity anonymous endpoints and to signal the intention to use the MakeConnection protocol to transfer messages between the endpoints. The WS-MakeConnection protocol uses the receipt of the MakeConnection message at an endpoint as the mechanism by which the back-channel of that connection can be uniquely identified. Once identified, the MC Receiver is then free to use that back-channel to send any pending message targeted to the URI specified within the MakeConnection message.

5.3.1 Addressing Variants

The WS-MakeConnection specification defines two distinct ways for the MC-Sender to indicate its messages of interest. One of these mechanisms uses the wsmc:MakeConnection Anonymous URI, the other uses a WS-RM Sequence ID. However, the WS-MakeConnection specification doesn't define any way of advertising or agreeing upon which variant of the MakeConnection protocol is supported or required by an endpoint. This creates the potential for different, incompatible implementations of WS-MakeConnection. To promote interoperability this Profile refines the WS-MakeConnection specification with additional requirements to mandate the use of a single, consistent addressing variant. Since the URI variant of WS-MakeConnection is a superset of the functionality of the Sequence-ID variant, use of the URI variant is mandated by this Profile.

R2100 *If an* **ENVELOPE** *contains a wsmc:MakeConnection element as a child of the SOAP Body, the wsmc:MakeConnection element MUST contain a wsmc:Address child element.* TESTABLERSP2100

R2101 *If an* **ENVELOPE** *contains a wsmc:MakeConnection element as a child of the SOAP Body, the wsmc:MakeConnection element MUST NOT contain a wsrm:Identifier child element.* TESTABLERSP2101

R2102 *If a wsmc:MakeConnection request does not contain a wsmc:Address child element (in violation of R2100), the* **MC-RECEIVER** *MUST generate a wsmc:MissingSelection fault.* TESTABLERSP2102

R2103 *If a wsmc:MakeConnection request contains a wsrm:Identifier element (in violation of R2101) the* **MC-RECEIVER** *MUST generate a wsmc:UnsupportedSelection fault.* TESTABLERSP2103

5.3.2 MakeConnection Anonymous URI

The following requirements describe how the MakeConnection anonymous URI is used in the various addressing properties and within RM protocol elements transmitted on SOAP messages.

R2110 *When present in a SOAP* **ENVELOPE***, the /wsmc:MakeConnection/wsmc:Address element MUST be set to a MakeConnection anonymous URI that identifies the MC-SENDER.* TESTABLERSP2110

R2111 *Once the MakeConnection protocol is established through the exchange of an EPR that contains the wsmc:MakeConnection Anonymous URI as its [address] property, the* **MC-RECEIVER** *MUST make use of the MakeConnection response channel to transfer messages targeted to that EPR.* TESTABLERSP2111

R2112 *A* **MESSAGE** *sent to a non-addressable endpoint MUST have the wsa:To addressing property set to an instance of the MakeConnection anonymous URI that identifies that endpoint, except in the following situation where this is [permitted but] not required (a) the message (that is not a WS-RM lifecycle message) is sent non-reliably over the back-channel of an underlying protocol connection initiated by the non-addressable endpoint.* TESTABLERSP2112

R2113 *When referring to a non-addressable endpoint, and if present in a SOAP* **ENVELOPE***, the /wsrm:CreateSequence/wsrm:Offer/wsrm:Endpoint element MUST be set to an instance of the WS-MakeConnection anonymous URI.* TESTABLERSP2113

5.4 MakeConnection Fault Behavior

Section 4 of WS-MakeConnection describes how to map the properties of the faults generated by WS-MakeConnection implementations to SOAP 1.1 and 1.2 fault messages. Although the description of the binding to SOAP 1.2 binds the [Detail] property of a fault to the `/soap12:Fault/soap12:Detail` element, there is no description of how to bind the [Detail] property to any element of a SOAP 1.1 fault message.

5.4.1 [Detail] Property Mapping

The following requirement describes how to bind the [Detail] property of a fault to a SOAP 1.1 fault message.

R2200 *An* **MC-RECEIVER** *that generates a SOAP 1.1 fault MUST include the value of the [Detail] property, if such a value exists, as the first child of the /soap11:Fault/detail element.* TESTABLERSP2200

6 Secure Reliable Messaging

This section of the Profile contains requirements that address the composition of reliable and secure messaging.

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

- Web Services Reliable Messaging 1.2 [WSRM1.2]
- Web Services Make Connection 1.1 [WSMC1.1]
- WS-SecureConversation 1.4 [WSSecCon1.4]
- WS-SecurityPolicy 1.3 [WSS-Policy1.3]

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

6.1 Initiating a Secure Sequence

6.1.1 Secure Context Identification

Section 5.2.2.1 of the WS-ReliableMessaging specification states that "During the CreateSequence exchange, the RM Source SHOULD explicitly identify the security context that will be used to protect the Sequence". This leaves open the possibility for RMS implementations that, for some reason, attempt to use WS-SC to secure their Sequences in some manner that does not explicitly identify the security context that will be used to protect the Sequence (e.g. by some out of band understanding of an inferred security context). This possibility creates an obvious operational and interoperability issues since (a) point-to-point, out-of-band configuration creates unscalable operational overhead and (b) not all WS-RM implementations may be capable of supporting such understandings.

Within Section 6, the phrase "secure Sequence" is defined as "a Sequence beginning with an exchange in which the wsrm:CreateSequence element has been extended with a wsse:SecurityTokenReference element." This profile does not cover the out-of-band understandings mentioned just above.

6.1.2 Security Token References

When initiating a secure Sequence, an RMS must ensure that the RMD both understands and will conform to the requirements listed above.

R3010 *If an* **ENVELOPE** *contains a wsrm:CreateSequence element as a child of the SOAP Body, and the proposed Sequence is to be secured, the ENVELOPE MUST also include the wsrm:UsesSequenceSTR element as a SOAP header block.* TESTABLE RSP3010

6.2 Signature Coverage

In a secure Sequence there exists both security and interoperability issues around the inclusion of SOAP message elements within signatures.

6.2.1 Single Signature for Sequence Header and SOAP Body

As discussed in Section 5.1.1 of WS-ReliableMessaging, any mechanism which allows an attacker to alter the information in a Sequence Traffic Message or break the linkage between a wsrm:Sequence header block and its assigned message, represents a threat to the WS-RM protocol.

R3100 *When present in an* **ENVELOPE** *in a secure Sequence, the wsrm:Sequence header block MUST be included in a signature.* TESTABLE RSP3100a RSP3100b

R3101 *In an* **ENVELOPE**, *the signature referred to in R3100 MUST be coupled cryptographically (e.g. share a common signature) with the message body.* TESTABLE RSP3101a RSP3101b

R3102 *In an* **ENVELOPE**, *the signature referred to in R3100 MUST be created using the key(s) associated with the security context that protects the applicable Sequence.* NOT_TESTABLE

6.2.2 Signed Elements

As discussed in Section 5.1.1 of WS-ReliableMessaging, any mechanism which allows an attacker to alter the information in a Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or Sequence-related fault represents a threat to the WS-RM protocol.

R3110 *If a wsrm:CreateSequence, wsrm:CreateSequenceResponse, wsrm:CloseSequence, wsrm:CloseSequenceResponse, wsrm:TerminateSequence, or wsrm:TerminateSequenceResponse element appears in the body of an* **ENVELOPE** *in a secure Sequence, that body MUST be included in a signature.* TESTABLE RSP3110a RSP3110b

R3111 *In an* **ENVELOPE**, *the signature referred to in R3110 MUST be created using the key(s) associated with the security context, that protects the applicable Sequence.* NOT_TESTABLE

R3114 *If a wsrm:AckRequested, or wsrm:SequenceAcknowledgement element appears in the header of an* **ENVELOPE** *and that element refers to a secure Sequence, that element MUST be included in a signature.* TESTABLE RSP3114a RSP3114b

R3115 *In an* **ENVELOPE**, *the signature referred to in R3114 MUST be created using the key(s) associated with the security context that protects the applicable Sequence.* NOT_TESTABLE

R3117 *When using SOAP 1.2, if a soap12:Fault element appears as the body of an* **ENVELOPE** *and the fault relates to a known secure Sequence, the soap12:Body MUST be included in a signature.* TESTABLE RSP3117a RSP3117b

R3118 *In an* **ENVELOPE**, *the signature referred to in R3117 MUST be created using the key(s) associated with the security context that protects the applicable Sequence.* NOT_TESTABLE

6.2.3 Single Signature for SOAP 1.1 Fault and SequenceFault Header

As described in Section 4.1 of WS-ReliableMessaging, the wsrm:SequenceFault element is used to carry the specific details any SOAP 1.1 faults generated during the WS-RM-specific processing of a message. As with SOAP 1.2, the integrity of fault information needs to be protected. In addition to this, it is necessary to ensure that the linkage between a wsrm:SequenceFault header and the soap11:Fault body is preserved.

R3120 *When using SOAP 1.1, if a wsrm:SequenceFault appears in the header of an* **ENVELOPE** *and the fault relates to a known secure Sequence, the wsrm:SequenceFault header MUST be included in a signature.* TESTABLE RSP3120a RSP3120b RSP3120c

R3121 *In an* **ENVELOPE***, the signature referred to in R3120 MUST be coupled cryptographically (e.g. share a common signature) with the message body.* TESTABLE RSP3121a RSP3121b

R3122 *In an* **ENVELOPE***, the signature referred to in R3120 MUST be created using the key(s) associated with the security context that protects the applicable Sequence.* NOT_TESTABLE

6.3 Secure Use of MakeConnection

This Profile places additional requirements on the composition of MakeConnection, WS-SecureConversation, and WS-ReliableMessaging.

6.3.1 Security Context for MakeConnection

From a security standpoint, it will be commonly desired that the security context of the message sent on the backchannel established by a MakeConnection and that of the MakeConnection message itself be the same. However, it is important to keep in mind that the WS-MakeConnection protocol is independent of the application protocol(s) flowing over it, thus there will be cases in which the MC-SENDER has no knowledge of the security context (if any) of the backchannel messages. For example, the WS-MakeConnection specification details a scenario in which MakeConnection is used to deliver Notifications from an Event Source. The Event Source may have a variety of different security contexts that it uses depending on the type of Notification being delivered. In this case the MC-SENDER has no way of knowing which security context, if any, should to be used. In such situations, the MC-RECEIVER needs to simply ensure that the MC-SENDER is authenticated. It would still be the MC-SENDER's responsibility to ensure that any message sent on the backchannel has the correct security context - just as would any endpoint receiving a message over a new connection.

6.3.2 Signing the MessagePending header

Since the value of the wsmc:MessagePending header effects the operation of the MakeConnection protocol, it must be protected to ensure the proper functioning of that protocol.

R3201 *If a wsmc:MessagePending element appears as a header block in an* **ENVELOPE***, that element MUST be signed using the key(s) associated with a security context, if any, that protects the SOAP Body of the ENVELOPE.* TESTABLE RSP3201

6.4 Replay Detection

As mentioned in Section 5 of WS-ReliableMessaging, there is a potential tension between certain aspects of security and reliable messaging; a security implementation may attempt to detect and prevent message replay attacks, but one of the invariants of the WS-RM protocol is to resend messages until they are acknowledged. Implementations must have the information necessary to distinguish between a valid retransmission of an unacknowledged message and a replayed message.

6.4.1 Unique Timestamp Values

R3300 *In the absence of WS-SecurityPolicy assertions that indicate otherwise, an* **ENVELOPE** *in a secure Sequence that contains a wsrm:Sequence header MUST contain a wsu:Timestamp as a sub-element of the wsse:Security header.* TESTABLERSP3300

R3301 *For any two* **ENVELOPE***s in a particular secure Sequence that contain WS-RM Sequence headers in which the value of their wsrm:MessageNumber elements are equal, it MUST be true that neither of the envelopes contains a wsu:Timestamp as a child element of wsse:Security header, OR that both messages contain a wsu:Timestamp as child elements of their wsse:Security headers and the value of these wsu:Timestamp elements are NOT equal.* TESTABLERSP3301

Appendix A. Extensibility Points

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

These mechanisms 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 Web Services Reliable Messaging 1.2 [WSRM1.2]:

- E0001 - **CreateSequence element and attribute extensions** - Extending CreateSequence, via additional elements or attributes, is the primary mechanism for negotiating supplemental semantics to be applied to the requested and/or offered Sequence. Note this extensiblity point does not cover the pre-defined use of the /wsrm:CreateSequence/wsse:SecurityTokenReference element.
- E0002 - **CreateSequenceResponse element and attribute extensions** - Extending CreateSequenceResponse, via additional elements or attributes, may be used to signal the acceptance of the supplemental semantics requested by the use of E0001 or it may be used in its own right to request or signal additional semantics to be applied to either requested and/or offered Sequence.
- E0003 - **CloseSequence element and attribute extensions** - The CloseSequence element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the closure of the Sequence.
- E0004 - **CloseSequenceResponse element and attribute extensions** - The CloseSequenceResponse element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the closure of the Sequence.
- E0005 - **TerminateSequence element and attribute extensions** - The TerminateSequence element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the termination of the Sequence.
- E0006 - **TerminateSequenceResponse element and attribute extensions** - The TerminateSequenceResponse element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options in the termination of the Sequence.
- E0007 - **Sequence element and attribute extensions** - The Sequence header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the Sequence identified by the header.
- E0008 - **AckRequested element and attribute extensions** - The AckRequest header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the request.
- E0009 - **SequenceAcknowledgment element and attribute extensions** - The SequenceAcknowledgment header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the acknowledgment.
- E0010 - **SequenceFault element and attribute extensions** - The SequenceFault element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the fault.

In WS-SecureConversation 1.4 [WSSecCon1.4]:

- E0011 - **SecurityContextToken element and attribute extensions** - The SecurityContextToken element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options that apply to the security context identified by the token.

In Web Services Make Connection 1.1 [WSMC1.1]:

- E0012 - **MakeConnection element and attribute extensions** - The MakeConnection element may be extended via additional elements or attributes to indicate the use of supplemental semantics or options that apply to the request.
- E0013 - **Attribute extensions to the MakeConnection Address** - The /wsmc:MakeConnection/wsmc:Address element may be extended via additional attributes to indicate the use of supplemental semantics or options that apply to this instance of the MakeConnection Anonymous URI
- E0014 - **MessagePending element and attribute extensions** - The MessagePending header element may be extended via additional elements or attributes to convey supplemental semantics or options that apply to the indication of pending messages.

Appendix B. Schemas

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

```
<xs:schema targetNamespace='http://ws-i.org/profiles/rsp/1.0/'
           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 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-realted 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) ***[markup: testAssertion/@id]***	A unique ID for the current test assertion.
Description (optional) ***[markup: testAssertion/description]***	A plain text description of the current test assertion. At minimum expressing the TA predicate.
Comments (optional) ***[markup: testAssertion/comments]***	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)	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

[markup: testAssertion/target]	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: • passed • failed • warning • notApplicable • notRelevant • missingInput • undetermined See the "reporting" item for the meaning of these results.
Cotarget (optional) ***[markup: testAssertion/cotarget]***	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 '$target'. For example, the target can be a SOAP message and the cotarget the WSDL file that describes this SOAP message. A cotarget must have a @name attribute that identifies it. The value of this attribute can be used as a variable (when prepending '$' to it) by subsequently defined cotargets, prerequisite and predicate.
Prerequisite (optional) ***[markup: testAssertion/@preReq]*** (optional) ***[markup: testAssertion/prerequisite]*** (optional)	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") 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. (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). 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]).
Predicate (required) ***[markup: testAssertion/predicate]***	A logical expression that evaluates whether this target is fulfilling the profile requirement addressed by this test Assertion. By default: - 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). However, in some cases and for testability reasons, the predicate may be designed as a partial indicator e.g. only indicates some cases of fulfillment, or some cases of violation. As a result, when "true" indicates fulfillment it may be that "false" is unconclusive, or conversely "false" will indicate violation, but "true" is unconclusive. In such cases, the "Reporting" element specifies the meaning of the predicate result w/r to the profile requirement. The predicate expression implicitly refers to the target (whic 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]).
Prescription (required) ***[markup: testAssertion/prescription/@level]***	Conveys the level of prescription associated with the profile requirement. At least three values may be used: • **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.
Reporting (optional) ***[markup: testAssertion/reporting]***	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: • **@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') The reported outcomes have the following meaning: • **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 deserved further investigation. • **undetermined**: the test result is inconclusive for this predicate value. 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'. Other test results that are automatically generated and not 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. 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 Reliable and Secure Profile Working Group whose work provided the foundation for this document, and in particular the former editorial team:

Gilbert Pilz, Oracle
Jacques Durand, Fujitsu

Appendix E. 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.