Reliable Secure Profile Version 1.0

Working Group Draft - revision 11

2007-10-25

This version:
http://www.ws-i.org/Profiles/ReliableSecureProfile-1.0-2007-10-25.html
Latest version:
http://www.ws-i.org/Profiles/ReliableSecureProfile-1.0.html
Editors:
Jacques Durand, Fujitsu
Anish Karmarkar, Oracle
Gilbert Pilz, BEA
Administrative contact:
secretary@ws-i.org

Abstract

This document defines the WS-I Reliable Secure Profile 1.0, consisting of a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability

Status of this Document

This document is a Working Group Draft; it has been accepted by the Working Group as reflecting the current state of discussions. It is a work in progress, and should not be considered authoritative or final; other documents may supersede this document.

Notice

The material contained herein is not a license, either expressly or impliedly, to any intellectual property owned or controlled by any of the authors or developers of this material or WS-I. The material contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and WS-I hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR WS-I BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Feedback

The Web Services-Interoperability Organization (WS-I) would like to receive input, suggestions and other feedback ("Feedback") on this work from a wide variety of industry participants to improve its quality over time.

By sending email, or otherwise communicating with WS-I, you (on behalf of yourself if you are an individual, and your company if you are providing Feedback on behalf of the company) will be deemed to have granted to WS-I, the members of WS-I, and other parties that have access to your Feedback, a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free license to use, disclose, copy, license, modify, sublicense or otherwise distribute and exploit in any manner whatsoever the Feedback you provide regarding the work. You acknowledge that you have no expectation of confidentiality with respect to any Feedback you provide. You represent and warrant that you have rights to provide this Feedback, and if you are providing Feedback on behalf of a company, you represent and warrant that you have the rights to provide Feedback on behalf of your company. You also acknowledge that WS-I is not required to review, discuss, use, consider or in any way incorporate your Feedback into future versions of its work. If WS-I does incorporate some or all of your Feedback in a

future version of the work, it may, but is not obligated to include your name (or, if you are identified as acting on behalf of your company, the name of your company) on a list of contributors to the work. If the foregoing is not acceptable to you and any company on whose behalf you are acting, please do not provide any Feedback.

Feedback on this document should be directed to wsi_rsp_comment@lists.ws-i.org.

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, WS-I Basic Profile 2.0, WS-I Basic Security Profile 1.0 and WS-I Basic Security Profile 1.1. 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.

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 Notational Conventions

The keywords "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.

Requirement identifiers can be considered to be namespace qualified, in such a way as to be compatible with QNames from Namespaces in XML. 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 by the conformance URI of the document section it occurs in. If it is qualified, the prefix should be interpreted according to the namespace mappings in effect, as documented below.

Some requirements clarify the referenced specification(s), but do not place additional constraints upon implementations. For convenience, clarifications are annotated in the following manner: C

Some requirements are derived from ongoing standardization work on the referenced specification(s). For convenience, such forward-derived statements are annotated in the following manner: xxxx, where "xxxx" is an identifier for the

specification (e.g., "WSDL20" for WSDL Version 2.0). Note that because such work was not complete when this document was published, the specification that the requirement is derived from may change; this information is included only as a convenience to implementers.

As noted above, some requirements may present compatibility issues (whether forwards or backwards) with previously published versions of the profile. For convenience, such requirements are annotated in the following manner: Compat

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"
- **xsi** - "http://www.w3.org/2001/XMLSchema-instance"
- **xsd** - "http://www.w3.org/2001/XMLSchema"
- **wsdl** - "http://schemas.xmlsoap.org/wsdl/"
- **soapbind** - "http://schemas.xmlsoap.org/wsdl/soap/"
- **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"

1.4 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.

2 Profile 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 Conformance Requirements

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; i.e., although this requirement must be met to maintain conformance in most cases, there are some situations where there may be valid reasons for it not being met (which are explained in the requirement itself, or in its accompanying text).

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.

None of the requirements in the Profile, regardless of their conformance level, should be interpreted as limiting the ability of an otherwise conforming implementation to apply security countermeasures in response to a real or perceived threat (e.g., a denial of service attack).

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)
- **DESCRIPTION** - descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (e.g., WSDL descriptions) (from Basic Profile 1.0)
- **INSTANCE** - software that implements a wsdl:port or a uddi:bindingTemplate (from Basic Profile 1.0)
- **CONSUMER** - software that invokes an INSTANCE (from Basic Profile 1.0)
- **SENDER** - software that generates a message according to the protocol(s) associated with it (from Basic Profile 1.0)
- **RECEIVER** - software that consumes a message according to the protocol(s) associated with it (e.g., SOAP processors) (from Basic Profile 1.0)
- **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 1.0 (from WS-MakeConnection 1.0)
- **MC-RECEIVER** - software that consumes a MakeConnection message as defined by WS-MakeConnection 1.0 (from WS-MakeConnection 1.0)

- **RMS** - RM Source as defined by WS-ReliableMessaging 1.1
- **RMD** - RM Destination as defined by WS-ReliableMessaging 1.1

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 Appendix A, as refined by the extensibility points in Appendix B.

2.4 Claiming Conformance

Claims of conformance to the Profile can be made using the following mechanisms, as described in Conformance Claim Attachment Mechanisms, when the applicable Profile requirements associated with the listed targets have been met:

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

The conformance claim URI for this Profile is "http://ws-i.org/profiles/rsp/1.0".

3. Reliable Messaging

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

- Web Services Reliable Messaging (WS-ReliableMessaging) 1.1
- Internationalized Resource Identifiers (IRIs)
- Web Services Addressing 1.0 - SOAP Binding

3.1 Use of Extensibility Points

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

3.1.1 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 ignore any extension elements and/or attributes that it does not recognize. Any exceptions to this rule are clearly identified in requirements below or the specifications underlying the profile*

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.

3.2 Retransmission of Messages

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

3.2.1 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.*

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

3.2.2 Retransmission of Sequence Lifecycle Messages

WS-ReliableMessaging 1.1 Section 2.1 defines the messages that affect the created/closing/closed/terminating state of a Sequence as "Sequence Lifecycle Messages". WS-RM 1.1 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.*

3.2.3 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.*

3.3 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:

3.3.1 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.*

3.3.2 Synchronizing Sequence Status

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 also to messages missing at the end of a sequence the RMD must be aware of the highest message number sent.

R0210 *When sending a wsrm:CloseSequence or a wsrm:TerminateSequence, an* **RMS** *MUST always include a LastMsgNumber element.*

3.4 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.

3.4.1 Transmission of Sequence Faults

In Section 4, "Faults" WS-ReliableMessaging 1.1 states that a receiver that generates a fault related to a known sequence SHOULD transmit that fault. However, the WS-I Basic Profile 1.2 states, in requirement R1029, that, under certain circumstances, the receiver must transmit the fault. Mapping the specifics of the BP 1.2 requirement onto the details of the WS-RM 1.1 specification results in the following requirement:

R0400 *If a fault is generated while processing a wsrm:CreateSequence, wsrm:CloseSequence, or wsrm:TerminateSequence message, or a message containing a wsrm:AckRequested header, the* **RECEIVER** *MUST transmit the fault.*

3.5 Piggybacked Acknowledgements

WS-ReliableMessaging 1.1 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 SequenceAcknowledgment headers.

3.5.1 Endpoint Comparison

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, at a minimum, perform a simple string comparison algorithm, as indicated in the RFC 3987 section 5.3.1, of the respective wsa:Address IRIs before piggybacking a SequenceAcknowledgement Header onto another message.*

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 1.1 specification.*

These requirements establish a minimum baseline for an RMD to correctly piggyback SequenceAcknowledgement headers. Individual RMD implementations may choose to consider and/or compare additional elements of the EndpointReference (e.g. the value of any ReferenceParameters elements).

3.5.2 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, 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.

R0502 *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.*

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 Sequence for which AcksTo/wsa:Address is not anonymous.

3.5.3 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.

> R0503 *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.*

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 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.

3.6 Sequence Assignment

WS-ReliableMessaging 1.1 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.

3.6.1 Sequence Assignment for 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 common understanding of the Sequence assignment mechanism for reliable response messages.

> R0600 *An* **RMS** *SHOULD use the same Sequence for all reliable response messages (replies and faults) corresponding to all reliable request messages that shared the same Sequence.*

Note that the RMS referred to above is a "server-side RMS" (i.e. the RMS responsible for transmitting response messages from the producer to the consumer in a reliable fashion).

3.7 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.

3.7.1 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.*

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.

4. Secure Conversation

The Profile includes the use of WS-SecureConversation 1.3 to request and issue security tokens and to broker trust relationship.

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

- WS-SecureConversation 1.3

4.1 Fault Codes for Unsupported Context Tokens

4.1.1 Unsupported Key Sizes

During the establishment of a security context, it is possible for a participant to obtain an SCT that, for some reason (key sizes, etc.), it cannot support. To promote interoperability, the parties involved in the establishment of a security context should share a common understanding of such a situation. WS-SC's references to other "more specific fault codes" opens the possibility for one participant to use fault codes that are not recognized by the other participants.

R1000 *If a* **RECEIVER** *obtains an SCT that it cannot, for whatever reason, support, the RECEIVER MUST generate a fault using the wsc:UnsupportedContextToken fault code.*

This requirement is a specific exception to the general guidelines outlined by R0001.

4.2 Demonstrating Proof of Possession

When attempting to amend, renew or cancel a security context, it is necessary for the requester to prove that they possess the key associated with the security context. WS-SecureConversation recommends, but does not require, that this be done in a specific fashion. This creates the potential for implementations to demonstrate proof of possession in ways that are not mutually understood, to the obvious detriment of both interoperability and security.

4.2.1 Amending Contexts

When attempting to amend an existing security context, the use of a single mechanism to demonstrate proof of possession of the key associated with the security context improves interoperability.

R1100 *When a* **SENDER** *makes a request to amend the claims associated with a security context, it MUST demonstrate proof of possession of the key associated with the security context by creating a signature over the message body and crucial headers using that key.*

4.2.2 Renewing Contexts

When attempting to renew a security context, the use of a single mechanism to demonstrate proof of possession of the key associated with the security context improves interoperability.

R1110 *When a* **SENDER** *makes a request to renew a security context, it MUST demonstrate proof of possession of the key associated with the*

security context by creating the original claims signature over the signature that signs the message body and crucial headers.

4.2.3 Cancelling Contexts

When attempting to cancel a security context, the use of a single mechanism to demonstrate proof of possession of the key associated with the security context improves interoperability.

R1120 *When a* **SENDER** *makes a request to cancel a security context, it MUST demonstrate proof of possession of the key associated with the security context by creating a signature over the message body and crucial headers using that key.*

4.3 Claims Re-Authentication

4.3.1 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.*

4.4 Referencing Security Context Tokens

4.4.1 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. Since message independent references provide a superset of the functionality of message dependent references, and it is simpler to support one mechanism for referencing SCTs than two, this profile includes the following requirement:

R1300 *In an* **ENVELOPE** *that contains either a wsse:Security header, a wst:RequestSecurityToken element, or a*

wst:RequestSecurityTokenReponse element in which there are references to wsc:SecurityContextToken elements, such references MUST be message independent (i.e. MUST use a wsse:Reference to the wsc:Identifier element).

4.4.2 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 `wsc:DerivedKeyToken` element is used to express these derived keys. WS-SC states that the `/wsc:DerivedKeyToken/wsse:SecurityTokenReference` element SHOULD be used to reference the `wsc: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 a* **SENDER** *uses a wsc:DerivedKeyToken, the wsse:SecurityTokenReference element MUST be used to reference the wsc:SecurityContextToken of the security context from which they key is derived.*

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.

4.5 Addressing Headers

4.5.1 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.

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

This requirement is not specific to the use of WS-SecureConversation. It applies whenever WS-Security is being used in conjunction with WS-Addressing.

5. MakeConnection

The Profile includes the use of WS-MakeConnection 1.0 to transfer messages using a transport-specific back-channel.

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

- Web Services Make Connection 1.0

5.1 Using MakeConnection

The use of MakeConnection is subject to the following requirements:

5.1.1 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.

R2000 *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.*

R2001 *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.*

5.1.2 MakeConnection Anonymous URI

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.

R2010 *When an* **MC-SENDER** *wishes to use the MakeConnection protocol to retrieve a message targeted to an EPR, the wsmc:MakeConnection Anonymous URI MUST be used within [address] property of that EPR.*

R2011 *Once the MakeConnection protocol is established through the exchange of an EPR that contains the wsmc:MakeConnection Anonymous URI as its [address] property, the MakeConnection message MUST be used to transfer messages targeted to that EPR from the* **MC-RECEIVER** *to the MC-SENDER.*

5.1.3 Use of MessagePending

The MakeConnection protocol defines the MessagePending header so that the MC Receiver can signal whether or not there are additional messages waiting to be delivered. The MC Sender can then use this information to determine the appropriate delay (if any) before sending another MakeConnection message.

R2020 *The* **MC-RECEIVER** *MUST include a MessagePending header on any message returned in response to a MakeConnection message, when additional messages are waiting to be transferred to the EPR that contains the wsmc:MakeConnection Anonymous URI.*

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 (WS-ReliableMessaging) 1.1
- Web Services Make Connection 1.0
- WS-SecureConversation 1.3
- WS-SecurityPolicy 1.2

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.

R3000 *During the wsrm:CreateSequence exchange, the* **RMS** *MUST explicitly identify the security context that will be used to protect the Sequence.*

This requirement only applies to those scenarios in which WS-SC is being used to protect a WS-RM Sequence.

6.1.2 Security Token References

In order to transmit a wsrm:CreateSequence that has been extended to include a wsse:SecurityTokenReference, an RMS must ensure that the RMD both understands and will conform with the requirements listed above.

R3010 *If an* **ENVELOPE** *contains a wsrm:CreateSequence element as a child of the soap:Body and that wsrm:CreateSequence element has been extended with a wsse:SecurityTokenReference element, the ENVELOPE MUST also include the UsesSequenceSTR element as a SOAP header block.*

6.2 Signature Coverage

When using WS-SecureConversation to secure a WS-ReliableMessaging 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***, the wsrm:Sequence header block and the SOAP body MUST be signed with a common signature that uses the key(s) associated with security context, if any, that protects the applicable sequence.*

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***, that body must be signed using the key(s) associated with security context, if any, that protects the applicable sequence.*

R3111 *If a wsrm:AckRequested, or wsrm:SequenceAcknowledgement element appears in the header of an* **ENVELOPE***, that element must be signed using the key(s) associated with security context, if any, that protects the applicable sequence.*

R3112 *When using SOAP 1.2, if a soap12:Fault element appears as the body of an* **ENVELOPE** *and the fault relates to a known sequence, the soap12:Body must be signed using the key(s) associated with security context, if any, that protects the applicable sequence.*

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***, the soap11:Body and the wsrm:SequenceFault header MUST be signed with a common signature that uses the key(s) associated with security context, if any, that protects the applicable sequence.*

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.

R3200 *An* **MC-RECEIVER** *MUST scope its searching of messages to those that were processed under the same security context as the message carrying the EPR that used the wsmc:MakeConnection Anonymous URI.*

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** *that contains a wsrm:Sequence header MUST contain a wsu:Timestamp as a sub-element of the wsse:Security header.*

R3301 *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 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 wsa:Timestamp elements are NOT equal.*

Appendix A: Referenced Specifications

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

- Web Services Reliable Messaging (WS-ReliableMessaging) 1.1
- Internationalized Resource Identifiers (IRIs)
- Web Services Addressing 1.0 - SOAP Binding
- WS-SecureConversation 1.3
- Web Services Make Connection 1.0
- WS-SecurityPolicy 1.2

Appendix B: 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; their use may affect interoperability, and may require private agreement between the parties to a Web service.

There are no extensibility points defined for this profile.

Appendix C: Acknowledgements

This document is the work of the WS-I Reliable Secure Profile Working Group, whose members have included:

Robert Freund (Hitachi Ltd.).