Basic Security Profile Version 1.1

Committee Specification 01

22 October 2014

Specification URIs

This version:

http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/cs01/BasicSecurityProfile-v1.1-cs01.doc (Authoritative)
http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/cs01/BasicSecurityProfile-v1.1-cs01.html
http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/cs01/BasicSecurityProfile-v1.1-cs01.pdf

Previous version:

http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/csprd01/BasicSecurityProfile-v1.1-csprd01.doc (Authoritative)
http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/csprd01/BasicSecurityProfile-v1.1-csprd01.html
http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/csprd01/BasicSecurityProfile-v1.1-csprd01.pdf

Latest version:

http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/BasicSecurityProfile-v1.1.doc (Authoritative)
http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/BasicSecurityProfile-v1.1.html
http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/BasicSecurityProfile-v1.1.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.

Related work:

This specification is related to:

- WS-I Basic Security Profile 1.1 Final Material 2010-01-24. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html.

Abstract:

The Basic Security Profile is an extension profile to the Basic Profile (either v1.1 or v1.0), consisting of a set of clarifications, refinements, interpretations and amplifications to a combination of non-proprietary Web services specifications in order to promote interoperability. It is designed to support the addition of security functionality to SOAP messaging.

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. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-brsp#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/ws-brsp/.

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

Citation format:

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

[BasicSecurityProfile-v1.1]

Basic Security Profile Version 1.1. Edited by Ram Jeyaraman, Tom Rutt, Jacques Durand, and Micah Hainline. 22 October 2014. OASIS Committee Specification 01. http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/cs01/BasicSecurityProfile-v1.1-cs01.html. Latest version: http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/BasicSecurityProfile-v1.1.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 https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 Introduction

This document defines the WS-I Basic Security Profile 1.1 (hereafter, "Profile"), consisting of a set of non-proprietary Web services specifications, along with clarifications to and amplifications of those specifications which promote interoperability.

Section 1, *"Introduction,"* introduces the Basic Security Profile 1.1 and relates the philosophy that it takes with regard to interoperability.

Section 2, *" Conformance,"* explains what it means to be conformant to the Basic Security Profile 1.1.

Section 3, *"Document Conventions,"* describes notational conventions utilized by the Basic Security Profile 1.1.

Each subsequent section addresses a component of the Basic Security Profile 1.1, 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 Guiding Principles

The Profile was developed according to a set of principles that, together, form the philosophy of the Basic Security Profile 1.1, as it relates to bringing about interoperability. This section documents these guidelines.

No guarantee of interoperability
: Although it is impossible to completely guarantee the interoperability of a particular service, the Basic Security Profile 1.1 attempts to increase interoperability by addressing the most common problems that implementation experience has revealed to date.

Focus profiling effort
: The focus of the Basic Security Profile 1.1 is the specifications that are explicitly defined as in-scope for the Basic Security Profile 1.1. Other specifications are profiled to the minimal extent necessary to allow meaningful profiling of the scoped specifications. This allows an in-depth profile of the scoped specifications with reduced constraining of other specifications.

Application semantics
: Although communication of application semantics can be facilitated by the technologies that comprise the Basic Security Profile 1.1, assuring the common understanding of those semantics is not addressed by it.

Testability
: When possible, the Basic Security Profile 1.1 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"). Note: Due to the nature of cryptographic security, non-intrusive testing may not be possible.

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., MAY, 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 (including the Basic Profile 1.0), the Basic Security Profile 1.1 may restrict them, but does not relax them (e.g., cannot change a MUST to a MAY).

Multiple mechanisms

If a referenced specification allows multiple mechanisms to be used interchangeably to achieve the same goal, the Basic Security Profile 1.1 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 Basic Security Profile 1.1 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 Basic Security Profile 1.1 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 Basic Security Profile 1.1, 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 Basic Security Profile 1.1 only addresses those that affect interoperability.

Conformance targets

Where possible, the Basic Security Profile 1.1 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 web-services layer only; it assumes that interoperability of lower-layer protocols (e.g. TCP, HTTP) and technologies (e.g. encryption and signature algorithms) is adequate and well-understood. WS-I does not attempt to assure the interoperability of these protocols and technologies as a whole. This assures that WS-I's expertise in and focus on Web Services standards is used effectively.

Do no harm

Interoperability of security technologies does not in and of itself ensure security, and the act of combining new technologies and protocols is especially susceptible to security threats. The profile takes steps to avoid introducing new security threats.

Best Practices

It is not the intent of the Basic Security Profile 1.1 to define security best practices. However, when multiple options exist, we may use known security weaknesses as a means of reducing choice and thus enhancing interoperability. The Basic Security Profile 1.1 will offer non-normative security considerations where the authors deem appropriate; however, these are by no means exhaustive and should not be perceived as a sanctioning of a security best practice.

Selected Errata Inclusion

The Basic Security Profile 1.1 restates selected requirements from the WS-Security Errata rather than including the entire Errata by reference, preferring interoperability over strict conformance.

1.2 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 in the Basic Security Profile 1.1 (i.e., those impacting conformance, as outlined in Section 2) are called “Requirements” and presented in the following manner:

Rnnnn*Statement text here.*

where "nnnn" is replaced by a number that is unique among the Requirements in the Basic Security Profile 1.1, thereby forming a unique Requirement identifier. Each Requirement uses one RFC2119 keyword.

Requirement identifiers can be considered to be namespace qualified, in such a way as to be compatible with QNames from Namespaces in XML [xml-names]. 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

Extensibility points in underlying specifications (see Section 2.3 "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 Basic Security Profile 1.1. 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.

- **soap** - "http://schemas.xmlsoap.org/soap/envelope/"
- **xsd** - "http://www.w3.org/2001/XMLSchema"
- **wsi** - "http://www.ws-i.org/schemas/conformanceClaim"
- **ds** - "http://www.w3.org/2000/09/xmldsig#"
- **xenc** - "http://www.w3.org/2001/04/xmlenc#"
- **c14n** - "http://www.w3.org/2001/10/xml-exc-c14n#"
- **wsse** - "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
- **wsse11** - "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
- **wsu** - "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- **b10** - "http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.htm"
- **bp11** - "http://www.ws-i.org/Profiles/BasicProfile-1.1.html"
- **saml** - "urn:oasis:names:tc:SAML:1.0:assertion"
- **saml2** - "urn:oasis:names:tc:SAML:2.0:assertion"
- **rel** - "urn:mpeg:mpeg21:2003:01-REL-R-NS"

1.3 Terminology

There is no terminology specific to this specification.

1.4 Profile Identification and Versioning

This document is identified by a name (in this case, Basic Security Profile) and a version number (here, 1.1). Together, they identify a *profile* (here Basic Security Profile 1.1)

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.

1.5 Normative References

[AP1.0] "Attachments Profile Version 1.0 (AP1.0)", WS-I Final Material, 20 April 2006, (ISO/IEC 29362:2008 Information technology -- Web Services Interoperability -- WS-I Attachments Profile Version 1.0), http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html

[BP1.0] K. Ballinger et al, “WS-I Basic Profile 1.0” , April 2004. http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[BP1.0Errata] "Basic Profile Version 1.0 Errata", 25 October 2005, http://www.ws-i.org/Profiles/BasicProfile-1.0-errata.html

[BP1.1] "Basic Profile Version 1.1 (BP 1.1)", WS-I Final Material, 10 April 2006, , (ISO/IEC 29361:2008 Information technology -- Web Services Interoperability -- WS-I Basic Profile Version 1.10), http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[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 199,. http://www.ietf.org/rfc/rfc2119.txt.

[RFC2459] R. Housley et al, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile” , January 1999, http://www.ietf.org/rfc/rfc2459.txt

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006. , http://www.ietf.org/rfc/rfc4514.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt

[SSLV3] Freirer, A,, P. Karlton, and P. Kocher, The SSL Protocol Version 3.0 , Internet Draft , November 18, 1996, http://tools.ietf.org/search/draft-ietf-tls-ssl-version3-00

[SSBP1.0] "Simple SOAP Binding Profile Version 1.0 (SSBP1.0)", WS-I Final Material, 24 August 2004, http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html

[WSS-User1.1] "Web Services Security: UsernameToken Profile 1.1", OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf

[WSS-SOAP] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[WSS-X.509-1.1] "Web Services Security: X.509 Certificate Token Profile 1.1", OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf

[WSS-Rel1.1] “Web Services Security: Rights Expression Language (REL) Token Profile 1.1” OASIS Standard Specification: 1 February 2006, http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf

[WSS-Kerb1.1] "Web Services Security: Kerberos Token Profile 1.1", OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

[WSS-SAML1.1] "Web Services Security: SAML Token Profile 1.1", OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[WSS-SWA1.1] "Web Services Security: SOAP Messages with Attachments (SwA) Profile 1.1", OASIS Standard, 1 February 2006, http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf

[xml-names] "Namespaces in XML 1.0 (Second Edition)", T. Bray, D. Hollander, A. Layman, R. Tobin, Editors, W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/

[xmldsig] "XML Signature Syntax and Processing (Second Edition)" , D. E. Eastlake, J. Reagle, D. Solo, F. Hirsch, T. Roessler, Editors, W3C Recommendation, 10 June 2008, http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/ .

[XPointer] "XPointer Framework" , P. Grosso, E. Maler, J. Marsh, N. Walsh, Editors, W3C Recommendation, 25 March 2003, http://www.w3.org/TR/2003/REC-xptr-framework-20030325/

[xmlenc] "XML Encryption Syntax and Processing" , D. E. Eastlake, J. Reagle, Editors, W3C Recommendation, 10 December 2002, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[X.509-2000TC1] "Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks Technical Corrigendum 1" , ITU-T Rec. X.509 (2000)/Cor.1, October 2001, http://www.itu.int/rec/T-REC-X.509-200110-S!Cor1

1.6 Non-Normative References

There are no non-normative references.

2 Conformance

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

2.1 Requirements Semantics

Requirements (see section 1.4 for the general format of a Requirement, including its identification) 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 Basic Security Profile 1.1 are normative.. When requirements in the Basic Security Profile 1.1 and its referenced specifications contradict each other, the Basic Security Profile 1.1's Requirements take precedence for purposes of Profile conformance.

For example;

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

This Requirement 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 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.

None of the Requirements in the Basic Security Profile 1.1, 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).

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

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

2.2 Conformance Targets

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

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

Requirements' conformance targets are generally processable artifacts with well-defined representation , to simplify testing and avoid ambiguity.

The following conformance targets are used in the Basic Security Profile 1.1:

- **BINARY_SECURITY_TOKEN** - a SECURITY_TOKEN named wsse:BinarySecurityToken.
- **CANONICALIZATION_METHOD** - an element named ds:CanonicalizationMethod, included as a child of a SIGNED_INFO or a wsse:TransformationParameters child of a SIG_TRANSFORM.
- **CREATED** - an element named wsu:Created, included as a child of a TIMESTAMP or USERNAME_TOKEN.
- **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)
- **DIGEST_METHOD** - an element named ds:DigestMethod, included as a child of a SIG_REFERENCE.
- **ED_ENCRYPTION_METHOD** - an element named xenc:EncryptionMethod, included as a child of an ENCRYPTED_DATA.
- **EK_DATA_REFERENCE** - an element named xenc:DataReference, included as a child of an EK_REFERENCE_LIST.
- **EK_ENCRYPTION_METHOD** - an element named xenc:EncryptionMethod, included as a child of an ENCRYPTED_KEY.
- **EK_KEY_REFERENCE** - an element named xenc:KeyReference, included as a child of an EK_REFERENCE_LIST.
- **EK_REFERENCE_LIST** - an element named xenc:ReferenceList, included as a child of an ENCRYPTED_KEY.
- **ENC_DATA_REFERENCE** - an element named xenc:DataReference, included as a child of an ENC_REFERENCE_LIST.
- **ENC_KEY_INFO** - an element named ds:KeyInfo, included as a child of an ENCRYPTED_KEY or ENCRYPTED_DATA.
- **ENC_KEY_REFERENCE** - an element named xenc:KeyReference, included as a child of an ENC_REFERENCE_LIST.
- **ENC_REFERENCE_LIST** - an element named xenc:ReferenceList, included as a child of a SECURITY_HEADER.
- **ENCRYPTED_DATA** - an element named xenc:EncryptedData, referenced by an EK_REFERENCE_LIST or an ENC_REFERENCE_LIST.
- **ENCRYPTED_HEADER** - a HEADER_ELEMENT named wsse11:EncryptedHeader.
- **ENCRYPTED_KEY** - an element named xenc:EncryptedKey, included as a child of a SECURITY_HEADER.
- **ENCRYPTED_KEY_TOKEN** - a SECURITY_TOKEN that is an ENCRYPTED_KEY.
- **EXPIRES** - an element named wsu:Expires, included as a child of a TIMESTAMP.
- **EXTERNAL_SAML_TOKEN** - an EXTERNAL_SECURITY_TOKEN that is a SAML_TOKEN.
- **EXTERNAL_SECURITY_TOKEN** - a security token defined in a security token profile, not included as a descendant of a SOAP_ENVELOPE.

- **EXTERNAL_TOKEN_REFERENCE** - a SECURITY_TOKEN_REFERENCE that refers to an EXTERNAL_SECURITY_TOKEN.
- **HEADER_ELEMENT** - an element included as a child of the SOAP_HEADER.
- **INCLUSIVE_NAMESPACES** - an element named xc14n:InclusiveNamespaces, include as a child of a SIG_TRANSFORM or a CANONICALIZATION_METHOD.
- **INSTANCE** - software that implements a wsdl:port or a uddi:bindingTemplate. (from Basic Profile 1.0)
- **INTERNAL_SAML_TOKEN** - an INTERNAL_SECURITY_TOKEN that is a SAML_TOKEN.
- **INTERNAL_SAML_V2_SECURITY_TOKEN** - An INTERNAL_SECURITY_TOKEN that is a SAML_V2_0_TOKEN.
- **INTERNAL_SECURITY_TOKEN** - a security token defined in a security token profile, included as a child of a SECURITY_HEADER or STR_EMBEDDED.
- **KERBEROS_TOKEN** - a BINARY_SECURITY_TOKEN containing a GSS wrapped Kerberos v5 AP-REQ.
- **MIME_BODY** - the body of a multipart entity, as defined by MIME.
- **MIME_HEADER** - a header field of a multipart entity, as defined by MIME.
- **MIME_PART** - the MIME_BODY and all MIME_HEADERs associated with a single multipart entity, as defined by MIME.
- **NONCE** - an element named wsse:Nonce, included as a child of a USERNAME_TOKEN.
- **PASSWORD** - an element named wsse:Password, included as a child of a USERNAME_TOKEN.
- **PKCS7_TOKEN** - a BINARY_SECURITY_TOKEN containing a PKCS#7 certificate chain.
- **PKIPATH_TOKEN** - a BINARY_SECURITY_TOKEN containing a PkiPath certificate chain.
- **RECEIVER** - software that consumes a message according to the protocol(s) associated with it. A receiver is considered conformant when it is capable of consuming conformant messages containing the artifacts that it supports and its behavior is conformant with all statements related to RECEIVER in the Basic Security Profile 1.1. A conformant receiver need not accept all possible conformant messages. A conformant receiver may choose not to support artifacts that provide unneeded or undesired functionality. When a receiver supports a specific artifact, and the Basic Security Profile 1.1 contains statements related to that artifact, a conformant receiver must accept all required conformant forms of that artifact. (from Basic Profile 1.0)
- **REL_TOKEN** - a SECURITY_TOKEN named rel:license.
- **SAML_AUTHORITY_BINDING** - an element named saml:AuthorityBinding, included as a child of an SECURITY_TOKEN_REFERENCE.
- **SAML_SC_KEY_INFO** - an element named ds:KeyInfo, included as a child of a SAML_SUBJECT_CONFIRMATION.
- **SAML_SUBJECT_CONFIRMATION** - an element named saml:SubjectConfirmation, included in a SAML_TOKEN.
- **SAML_TOKEN** - a SECURITY_TOKEN which is a SAML_V1_1_TOKEN or a SAML_V2_0_TOKEN.
- **SAML_V1_1_TOKEN** - a SECURITY_TOKEN named saml:Assertion which conforms to SAML 1.1 (via the OASIS Web Services Security SAML Token Profile 1.0).
- **SAML_V2_0_TOKEN** - a SECURITY_TOKEN named saml2:Assertion which conforms to SAML 2.0 (via the OASIS Web Services Security SAML Token Profile 1.1).
- **SECURE_ENVELOPE** - a SOAP envelope that contains sub-elements that have been subject to integrity and/or confidentiality protection. A message is considered conformant when all of its contained artifacts are conformant with all statements targeted to those artifacts as appropriate in the Basic Security Profile. Use of artifacts for which there are no statements in the Basic Security Profile does not affect conformance.
- **SECURE_MESSAGE** - protocol elements that have Web Services Security applied to them. Protocol elements include a primary SOAP envelope and optionally associated SwA attachments.
- **SECURITY_HEADER** - a HEADER_ELEMENT named wsse:Security.
- **SECURITY_TOKEN** - an INTERNAL_SECURITY_TOKEN or EXTERNAL_SECURITY_TOKEN (e.g. USERNAME_TOKEN, X509_TOKEN, REL_TOKEN, SAML_TOKEN, SAML_V2_0_TOKEN, KERBEROS_TOKEN, etc.).

- **SECURITY_TOKEN_REFERENCE** - an element named wsse:SecurityTokenReference, included as a descendant of a SECURITY_HEADER or ENCRYPTED_DATA.
- **SENDER** - software that generates a message according to the protocol(s) associated with it. A sender is considered conformant when all of the messages it produces are conformant and its behavior is conformant with all statements related to SENDER in the Basic Security Profile 1.1. (from Basic Profile 1.0)
- **SIG_KEY_INFO** - an element named ds:KeyInfo, included as a child of a SIGNATURE.
- **SIG_REFERENCE** - an element named ds:Reference, included as a child of a SIGNED_INFO.
- **SIG_TRANSFORMS** - an element named ds:Transforms, included as a child of a SIG_REFERENCE.
- **SIG_TRANSFORM** - an element named ds:Transform, included as a child of a SIG_TRANSFORMS.
- **SIGNATURE** - an element named ds:Signature, included as a child of a SECURITY_HEADER.
- **SIGNATURE_METHOD** - an element named ds:SignatureMethod, included as a child of a SIGNED_INFO.
- **SIGNED_INFO** - an element named ds:SignedInfo, included as a child of a SIGNATURE.
- **SOAP_ENVELOPE** - an element named soap:Envelope, which has no parent element.
- **SOAP_HEADER** - an element named soap:Header, included as a child of the SOAP_ENVELOPE.
- **STR_EMBEDDED** - an element named wsse:Embedded, included as a child of a SECURITY_TOKEN_REFERENCE.
- **STR_ISSUER_SERIAL** - an element named ds:X509IssuerSerial, included as a child of a child element named ds:X509Data of a SECURITY_TOKEN_REFERENCE.
- **STR_KEY_NAME** - an element named ds:KeyName, included as a child of a SECURITY_TOKEN_REFERENCE.
- **STR_KEY_IDENTIFIER** - an element named wsse:KeyIdentifier, included as a child of a SECURITY_TOKEN_REFERENCE.
- **STR_REFERENCE** - an element named wsse:Reference, included as a child of a SECURITY_TOKEN_REFERENCE.
- **TIMESTAMP** - an element named wsu:Timestamp, included as a child of a SECURITY_HEADER.
- **USERNAME_TOKEN** - a SECURITY_TOKEN named wsse:UsernameToken.
- **X509_TOKEN** - a BINARY_SECURITY_TOKEN containing an X.509 certificate.

2.3 Conformance Scope

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

The Profile's scope is further limited by extensibility points. Referenced specifications often provide extension mechanisms and unspecified or open-ended configuration parameters; when identified in the Basic Security Profile 1.1 as an extensibility point, such a mechanism or parameter is outside the scope of the Basic Security Profile 1.1, and its use or non-use is not relevant to conformance.

These extensibility points are however listed here for advisory purpose only, to point at possible risks of interoperability loss that are not addressed by the Basic Security Profile 1.1.

Because the use of extensibility points may impair interoperability, their use should be negotiated or documented in some fashion by the parties to a Web service; for example, this could take the form of an out-of-band agreement.

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

The Profile's scope is defined by the referenced specifications in clause 1.5, as limited by the extensibility points in Appendix A.

2.4 Conformance Clauses

This Profile concerns several conformance targets. Conformance targets of Basic Security Profile 1.1 are listed in Section 2.2. Each Requirement identifies its conformance target as described in Section 2.2. Conformance to the Basic Security Profile 1.1 is defined by adherence to the set of *requirements* defined for a specific *target*, within the *scope* of the Profile.

This Profile is an extension of the Basic Profile (1.1 or 1.0) It includes additional Requirements to those of BP. Conformance to this profile should always be mentioned or claimed in conjunction with a mention or claim about which one of the basic profile (V1.1 or V1.0) is used as foundation. However, only a few Requirements make a distinction between BP1.1 and BP1.0. Such Requirements are prefixed with either “bp11” or “bp10”.

The two conformance clauses for BSP1.1 reflect the two options for the underlying Basic Profile.

2.4.1 Conformance based on BP1.0

A conformance target (as previously defined) is said to be conforming to this profile based on BP1.0 if this target fulfills all the mandatory Requirements that identify this target type, and that are either annotated for “bp10” or not annotated for any profile foundation at all..

2.4.2 Conformance based on BP1.1

A conformance target (as previously defined) is said to be conforming to this profile based on BP1.1 if this target fulfills all the mandatory Requirements that identify this target type, and that are either annotated for “bp11” or not annotated for any profile foundation at all..

2.5 Claiming Conformance

Claims of conformance to the Basic Security Profile 1.1 MAY be made using the following mechanisms, as described in Conformance Claim Attachment Mechanisms [claimAttachment], when the applicable Profile Requirements associated with the listed targets have been met:

The conformance claim URI for the Basic Security Profile 1.1 is " http://ws-i.org/profiles/basic-security/1.1/core " , with the following additional conformance claim URIs, which are associated with specific sections:

- Section 4 Transport Layer Mechanisms - "http://ws-i.org/profiles/basic-security/1.1/transport"
- Section 12 Username Token - "http://ws-i.org/profiles/basic-security/1.1/username-token"
- Section 13 X.509 Certificate Token - "http://ws-i.org/profiles/basic-security/1.1/x.509-certificate-token"
- Section 14 REL Token - "http://ws-i.org/profiles/basic-security/1.1/rel-token"
- Section 15 Kerberos Token - "http://ws-i.org/profiles/basic-security/1.1/kerberos-token"
- Section 16 SAML Token - "http://ws-i.org/profiles/basic-security/1.1/saml-token"
- Section 18 Attachment Security - "http://ws-i.org/profiles/basic-security/1.1/swa"

If a claim of conformance is made as described in CCAM to Basic Security Profile 1.1 (" http://ws-i.org/profiles/basic-security/1.1/core "), then the claim MUST also specify which security tokens, be they BSP profile tokens or other mutually agreed upon tokens, are supported.

The conformance URI for transport level security ("http://ws-i.org/profiles/basic-security/1.1/transport ") can be used in isolation or in combination with other conformance URIs.

3 Document Conventions

This document follows conventions common to all WS-I profiles. These are described in the following sections.

3.1 Security Considerations

In addition to interoperability recommendations and Requirements (which are made in Rnnnn statements and intended to improve interoperability), the Basic Security Profile 1.1 makes a number of security recommendations intended to improve security. These Security Considerations are presented as follows:

> Cnnnn *Statement text here.*

where "nnnn" is replaced by a number that is unique among the security recommendations in the Basic Security Profile 1.1, thereby forming a unique security recommendation identifier. Each security recommendation contains a *SHOULD* or a *MAY* to highlight exactly what is being recommended. Security recommendations are expected to be tested by the test tools to highlight possible security problems, but have no impact on conformance.

It should be understood that, while a number of recommendations are made about security, adherence to these security recommendations does not guarantee security.

4 Transport Layer Mechanisms

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference, and defines extensibility points within them:

- RFC 2818: HTTP over TLS [RFC2818]
- RFC 2246: The TLS Protocol Version 1.0 [RFC2246]
  Extensibility points:
  - E0009 - TLS Ciphersuites - TLS allows for the use of arbitrary encryption algorithms. Note that while section 4.2 of the Basic Security Profile 1.1 mandates, recommends, and discourages support for certain ciphersuites, the Basic Security Profile 1.1 does not prohibit use of any specific ciphersuite.
  - E0010 - TLS Extensions - TLS allows for extensions during the handshake phase.
- The SSL Protocol Version 3.0 [SSLV3]
  Extensibility points:
  - E0011 - SSL Ciphersuites - SSL allows for the use of arbitrary encryption algorithms. Note that while section 4.2 of the Basic Security Profile 1.1 mandates, recommends, and discourages support for certain ciphersuites, the Basic Security Profile 1.1 does not prohibit use of any specific ciphersuite.

4.1 TLS and SSL Versions

SSL and TLS are both used as underlying protocols for HTTP/S. The Profile places the following constraints on those protocols:

4.1.1 SSL 2.0 Prohibited

SSL 2.0 has known security issues and all current implementations of HTTP/S support more recent protocols. Therefore the Basic Security Profile 1.1 prohibits use of SSL 2.0.

R2001 *A* **SENDER** *MUST NOT use SSL 2.0 as the underlying protocol for HTTP/S.*

R2002 *A* **RECEIVER** *MUST NOT use SSL 2.0 as the underlying protocol for HTTP/S.*

4.2 TLS and SSL Ciphersuites

In SSL and TLS, choices of algorithms are expressed as ciphersuites. The subsections of this section specify ciphersuites that are required, recommended, discouraged and prohibited, respectively. The use of any other ciphersuite not discussed below is optional.

4.2.1 Mandatory Ciphersuites

The specified algorithm suites are considered to be widely-implemented, secure and interoperable.

R5701 *Any TLS-capable* **INSTANCE** *that is not FIPS compliant MUST support TLS_RSA_WITH_3DES_EDE_CBC_SHA*

R5702 *Any SSL-capable* **INSTANCE** *that is not FIPS compliant MUST support SSL_RSA_WITH_3DES_EDE_CBC_SHA*

R5703 *Any TLS-capable* **INSTANCE** *that is FIPS compliant MUST support TLS_RSA_FIPS_WITH_3DES_EDE_CBC_SHA*

R5704 *Any SSL-capable* **INSTANCE** *that is FIPS compliant MUST support SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA*

4.2.2 Recommended Ciphersuites

As the AES encryption algorithm is intended to supersede the 3DES algorithm, it is recommended that TLS-capable implementations implement TLS_RSA_WITH_AES_128_CBC_SHA or the FIPS equivalent, and SSL-capable implementations implement SSL_RSA_WITH_AES_128_CBC_SHA or the FIPS equivalent.

4.2.3 Discouraged Ciphersuites

The ciphersuites defined in the SSL and TLS specifications that use anonymous Diffie-Hellman (i.e. those that have DH_anon in their symbolic name) are vulnerable to man-in-the-middle attacks. It is also recommended that ciphersuites that include MD5 (i.e. those that have MD5 in their symbolic name) be avoided, due to known security weaknesses of the MD5 algorithm. It is recommended that such ciphersuites be avoided.

The Profile recommends against the use of the following ciphersuites due to their lack of confidentiality services:

- SSL_RSA_WITH_NULL_SHA
- TLS_RSA_WITH_NULL_SHA
- SSL_RSA_WITH_NULL_MD5
- TLS_RSA_WITH_NULL_MD5

It is also recommended that ciphersuites that use 40 or 56 bit keys be avoided, due to their relative ease of compromise through brute-force attack.

4.2.4 Prohibited Ciphersuites

The Profile does not prohibit the use of any transport layer security ciphersuites, but careful thought should be given prior to the use of any ciphersuites discussed under "4.2.3 Discouraged ciphersuites".

5 SOAP Nodes and Messages

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference, and defines extensibility points within them:

- Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification [WSS-SOAP]
  Extensibility points:
  - E0002 - Security Tokens - Security tokens may be specified in additional security token profiles.
- Basic Profile Version 1.0 [BP1.0]
- Basic Profile Version 1.0 Errata [BP1.0Errata]
- Basic Profile Version 1.1 [BP1.1]
- Simple SOAP Binding Profile Version 1.0 [SSBP1.0]

5.1 Security Policy

5.1.1 Out of Band Agreement

Partners in a message exchange must agree which elements must be signed and/or encrypted and which elements may be signed and/or encrypted and which security tokens must be present and may be present.

R3105 *A* **SENDER** *MAY agree in an out of band fashion with a RECEIVER on required and allowed signed and/or encrypted message content and security tokens.*

R3106 *A* **RECEIVER** *MAY agree in an out of band fashion with a SENDER on required and allowed signed and/or encrypted message content and security tokens.*

Some vendor platforms have strict implementation of what message content must be signed and/or encrypted and which security tokens must be present and which may be present. Other vendor platforms are more tolerant to receiving additional signed content in a message. The Profile allows for an out of band agreement between partners on how to address this issue.

5.2 SOAP Envelope

5.2.1 Secure Envelope Validity

The Envelope, Header, or Body elements must not be encrypted. Encrypting these elements would break the SOAP processing model and is therefore prohibited.

R5607 *Any* **SECURE_ENVELOPE** *MUST still be a valid SOAP Envelope after SOAP Message Security, including encryption, is applied.*

5.2.2 wsu:Id Attribute Value Uniqueness

One of the principles of the underlying specifications is that processing of messages should not require schema validation. However, without schema processing it is not possible to determine whether individual attributes are of type ID and must therefore be unique.

R3204 *Any* **SECURE_ENVELOPE** *MUST NOT contain more than one element specifying the same wsu:Id attribute value.*

Since verification of signatures typically requires dereferencing of elements based on ID attribute values, these values are required to be unique within a message.

5.3 Intermediary Processing

5.3.1 Removal of Headers

R3207 *A SOAP intermediary* ***INSTANCE*** *MUST NOT remove or modify any HEADER_ELEMENT unless that SOAP intermediary is acting in the role specified by the S11:actor attribute of that HEADER_ELEMENT.*

5.4 Basic Profile Clarification

The Basic Security Profile is an extension profile to the Basic Profile. This means it is consistent with the Basic Profile but profiles additional functionality - how to add conformant security features to the Basic Profile when needed.

As an extension of the Basic Profile, the Basic Security Profile is designed to support the addition of security functionality to SOAP messaging, in an interoperable manner. One example of such functionality is the confidentiality of selected SOAP header blocks and SOAP body elements and content through the use of OASIS Web Services Security encryption. The intent of such techniques is to change the nature of the SOAP message so that unintended parties cannot read such content. This means that the secured SOAP message is no longer obviously related to the original WSDL description, and is not intelligible without decryption. Other security mechanisms such as signatures may also modify the content of SOAP envelopes.

The Basic Profile includes requirements on the content of SOAP envelopes (or in Basic Profile 1.0 the format of SOAP messages). Testing conformance to these statements by using a "man-in-the-middle" interceptor as outlined in the WS-I Monitor Tool Functional Specification will not be possible if encryption has been applied to portions of the SOAP envelope and have not yet been decrypted. Even if interception is possible, some messages may have a different structure due to security.

Such SOAP messages still conform to the Basic Profile, since conformance to the Basic Profile means conformance once a receiver has reversed security changes introduced by a message sender. This is not obvious in some Basic Profile requirements, so this document further clarifies these requirements in the normative "Basic Profile Clarifications" section below.

It is helpful to visualize a SOAP message in light of a protocol layering model, such as the ISO seven layer protocol model [Tanenbaum]. This model shows how a protocol is in fact composed of different layers, and how to a given layer underlying layers are transparent. The implementation of a given protocol layer at an endpoint may be modeled as that implementation consuming a service of the underlying protocol layer, and providing a service to the layer above it. In this model no protocol layer need be aware of layers above or below it, making the layer implementations independent. This is illustrated in Figure 1.

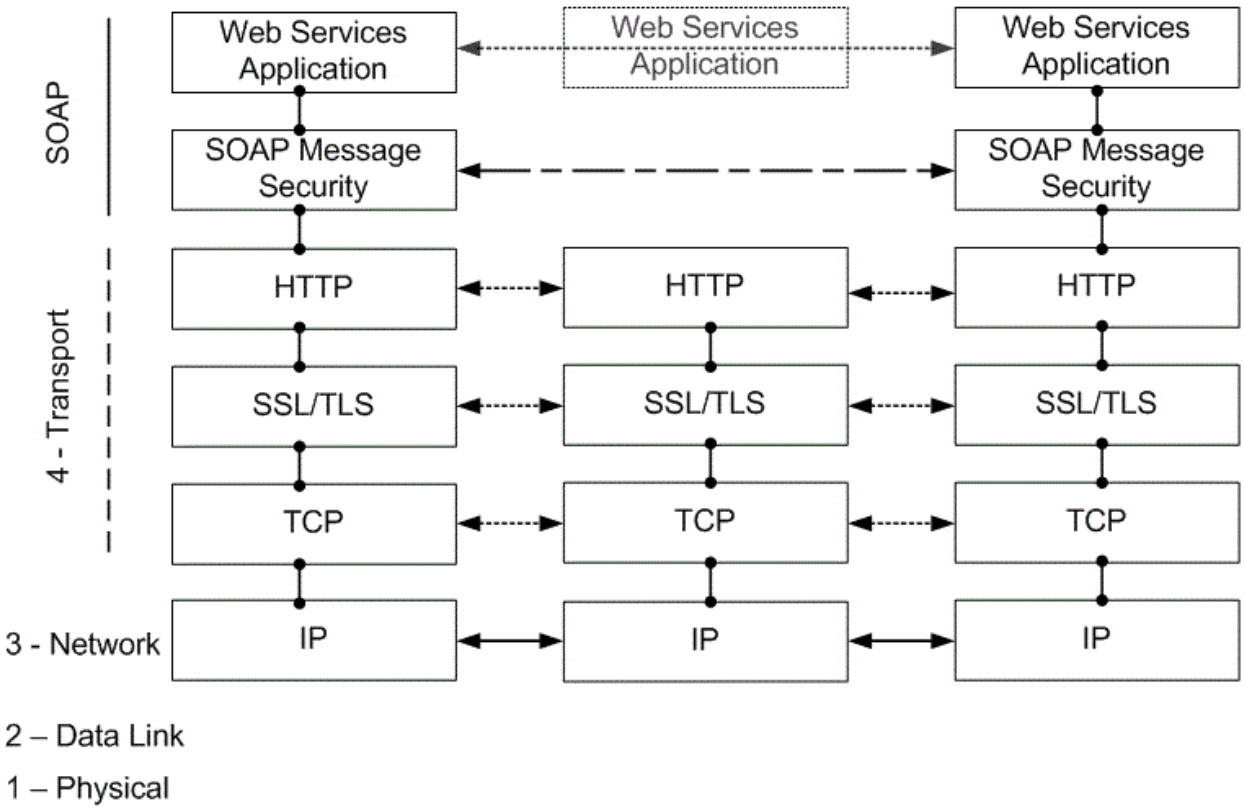

Figure 1: Protocol Stack with SOAP Message Security

Traditionally, protocol layers have been distinguished by the use of protocol enveloping, where the message at one layer is conveyed as the body in the next lower layer. The sender passes a message to the lower level protocol implementation that packages it in a protocol envelope and sends it to the corresponding layer in the receiver. The sender and receiver at this lower layer perform whatever processing is necessary for delivery according to the specification of that layer, and finally the receiver passes the message up to the peer of the sender.

SOAP Security may be viewed as a lower layer with respect to the more general SOAP web services application layer. Thus a SOAP sender may pass a SOAP message to a lower layer SOAP security implementation that applies encryption (for example), and sends the message to the destination SOAP Security layer, which removes the encryption before passing the message up to the peer SOAP web services application layer.

Thus a Basic Profile interceptor and compliance monitoring activity should logically occur at a receiver at the interface between the SOAP security implementation and SOAP web services application layer.

This section clarifies the BP1.0 (including Errata), BP1.1, SSBP1.0, and AP1.0 statements that might be unclear when SOAP Message Security is applied in compliance with the Basic Security Profile.

This section lists each possibly confusing BP1.0, BP1.1, SSBP1.0, and AP1.0 requirement and an associated statement to clarify that requirement in the context of the basic security profile.

When these clarifying statements include the phrase "reverse SOAP Message Security" it means to remove various impacts of applying SOAP Message Security that may have been applied since the MESSAGE (BP1.0) or ENVELOPE (BP 1.1) was originally created for that recipient according to the BP. This may mean decrypting relevant portions of the XML or removing XML Signature elements or making

other reverse transformations as appropriate to the aspects of SOAP Message Security that were applied in the specific circumstance.

Not all security must be reversed, only that for the intended recipient, as applied to the BP compliant envelope before sent to that recipient.

5.4.1 BP Requirement R1029

This clarifies the Basic Profile's R1029 to reflect the fact that transmission of security related faults may increase the vulnerability to certain attacks and in some cases faults should not be transmitted.

R5814 *Where the normal outcome of processing a SECURE_ENVELOPE would have resulted in the transmission of a SOAP Response, but rather a fault is generated instead, a* **RECEIVER** *MAY transmit a fault or silently discard the message.*

5.4.2 BP Requirement R2301

bp10:R2301 states "The order of the elements in the soap:body of a MESSAGE MUST be the same as that of the wsdl:parts in the wsdl:message that describes it."

bp11:R2301 states "The order of the elements in the soap:body of an ENVELOPE MUST be the same as that of the wsdl:parts in the wsdl:message that describes it."

R5800 *bp10:R2301 MUST be true after any SOAP Message Security has been reversed for the* **MESSAGE**.

R5801 *bp11:R2301 MUST be true after any SOAP Message Security has been reversed for the* **ENVELOPE**.

5.4.3 BP Requirement R2710

bp10:R2710 states "The operations in a wsdl:binding in a DESCRIPTION MUST result in operation signatures that are different from one another."

bp11:R2710 states "The operations in a wsdl:binding in a DESCRIPTION MUST result in operation signatures that are different from one another."

R5802 *bp10:R2710 MUST be true after SOAP Message Security processing has been reversed for the* **MESSAGE**

R5803 *bp11:R2710 MUST be true after SOAP Message Security processing has been reversed for the* **ENVELOPE**

5.4.4 BP Requirement R2712

bp10:R2712 states "A document-literal binding MUST be serialized as a MESSAGE with a soap:Body whose child element is an instance of the global element declaration referenced by the corresponding wsdl:message part."

bp11:R2712 states "A document-literal binding MUST be serialized as an ENVELOPE with a soap:Body whose child element is an instance of the global element declaration referenced by the corresponding wsdl:message part."

R5804 *bp10:R2712 MUST be true after any SOAP Message Security has been reversed for the* **MESSAGE**

R5805 *bp11:R2712 MUST be true after any SOAP Message Security has been reversed for the* **ENVELOPE**

5.4.5 BP Requirement R2724

bp10:R2724 states "If an INSTANCE receives a message that is inconsistent with its WSDL description, it SHOULD generate a soap:Fault with a faultcode of 'Client', unless a 'MustUnderstand' or 'VersionMismatch' fault is generated."

bp11:R2724 states "If an INSTANCE receives an envelope that is inconsistent with its WSDL description, it SHOULD generate a soap:Fault with a faultcode of 'Client', unless a 'MustUnderstand' or 'VersionMismatch' fault is generated."

R5806 *For bp10:R2724 "Inconsistent" MUST be taken to mean "Inconsistent after SOAP Message security has been reversed", for the* **MESSAGE**

R5807 *For bp11:R2724 "Inconsistent" MUST be taken to mean "Inconsistent after SOAP Message security has been reversed", for the* **ENVELOPE**

5.4.6 BP Requirement R2725

bp10:R2725 states "If an INSTANCE receives a message that is inconsistent with its WSDL description, it MUST check for "VersionMismatch", "MustUnderstand" and "Client" fault conditions in that order."

bp11:R2725 states "If an INSTANCE receives an envelope that is inconsistent with its WSDL description, it MUST check for "VersionMismatch", "MustUnderstand" and "Client" fault conditions in that order."

R5808 *With respect to bp10:R2725 the* **INSTANCE** *must check for consistency of the MESSAGE per BP 1.0 after reversing SOAP Message Security.*

R5809 *With respect to bp11:R2725 the* **INSTANCE** *must check for consistency of the ENVELOPE per BP 1.1 after reversing SOAP Message Security.*

5.4.7 BP Requirement R2729

bp10:R2729 states "A MESSAGE described with an rpc-literal binding that is a response message MUST have a wrapper element whose name is the corresponding wsdl:operation name suffixed with the string 'Response'."

bp11:R2729 states "An ENVELOPE described with an rpc-literal binding that is a response MUST have a wrapper element whose name is the corresponding wsdl:operation name suffixed with the string 'Response'."

R5810 *With respect to bp10:R2729 the verification of the wrapper element name of the* **MESSAGE** *must be performed after reversing SOAP Message Security.*

R5811 *With respect to bp11:R2729 the verification of the wrapper element name of the* **ENVELOPE** *must be performed after reversing SOAP Message Security.*

5.4.8 BP Requirement R2738

bp10:R2738 states "A MESSAGE MUST include all soapbind:headers specified on a wsdl:input or wsdl:output of a wsdl:operation of a wsdl:binding that describes it.

bp11:R2738 states "An ENVELOPE MUST include all soapbind:headers specified on a wsdl:input or wsdl:output of a wsdl:operation of a wsdl:binding that describes it."

R5812 *With respect to bp10:R2738 verification of a* **MESSAGE** *must occur after SOAP Message Security has been reversed.*

R5813 *With respect to bp11:R2738 verification of an* **ENVELOPE** *must occur after SOAP Message Security has been reversed.*

6 SecurityHeaders

6.1 Processing Order

Web Services Security: SOAP Message Security defines the order for processing elements within wsse:Security headers. The Profile provides the following guidance:

6.1.1 In Order of Appearance

Messages may be signed and encrypted, potentially by multiple entities signing and encrypting overlapping elements. A signature applied before encryption has different security properties than encryption applied before a signature. Determining which security properties should be used requires an out-of-band agreement.

With signature before encryption, the signer is known to have created or vouched for the plaintext data. It is not known to the receiver whether the signer performed the encryption. The potential exists for the identity of the signer to remain confidential except to the receiver by encryption of the signature and signer's security token.

With encryption before signature, the signer is known to have created or vouched for the ciphertext data, but it is not known whether the signer was aware of the plaintext. It is known that the signer was aware that the data was encrypted and intended to be delivered to the receiver.

R3212 *Any SIGNATURE, ENCRYPTED_KEY, and ENC_REFERENCE_LIST elements MUST be ordered within a* **SECURITY_HEADER** *so a receiver will get the correct result by processing the elements in the order they appear.*

As signature and encryption elements are added to a security header they must be ordered in a way that ensures that if a receiver of the message processing the elements in the order they appear they will achieve the correct result.

6.2 SOAP Actor Attribute

SOAP defines an actor attribute for use in SOAP headers. The Profile places the following constraints on its use with Security headers:

6.2.1 Avoid Target Ambiguity

The actor attribute allows a security header to be targeted to a specific processing component or node.

R3206 *Any* **SOAP_HEADER** *MUST NOT contain more than one SECURITY_HEADER with the actor attribute omitted.*

R3210 *Any* **SOAP_HEADER** *MUST NOT contain more than one SECURITY_HEADER with the same actor attribute value.*

Correct security header processing is order dependent. Eliminating potential ambiguity caused by ordering dependencies between headers targeted to the same actor eliminates complexity.

7 Timestamps

Web Services Security: SOAP Message Security defines a Timestamp element for use in SOAP messages. The Profile places the following constraints on its use:

7.1 Placement

7.1.1 Not More Than One per Security Header

R3227 *A* **SECURITY_HEADER** *MUST NOT contain more than one TIMESTAMP.*

7.2 Content

7.2.1 Exactly One Created per Timestamp

The wsu:Created element represents the creation time of the security semantics.

R3203 *A* **TIMESTAMP** *MUST contain exactly one CREATED.*

This element can only be specified once in a Timestamp element. Within the SOAP processing model, creation is the instant that the Infoset is serialized for transmission.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsu:Timestamp element is
missing a wsu:Created child element -->
          <wsu:Timestamp wsu:Id="timestamp">
          <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires>
          </wsu:Timestamp>
```

CORRECT:

```
          <wsu:Timestamp wsu:Id="timestamp">
          <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
          <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires>
          </wsu:Timestamp>
```

7.2.2 Not More Than One Expires per Timestamp

R3224 *Any* **TIMESTAMP** *MUST NOT contain more than one EXPIRES.*

7.2.3 Created Precedes Expires in Timestamp

A timestamp may optionally contain an expires element.

R3221 *Any* **TIMESTAMP** *containing an EXPIRES MUST contain a CREATED that precedes its sibling EXPIRES.*

Preventing multiple expires elements and enforcing the order of elements reduces complexity.

7.2.4 Timestamp Contains Nothing Other Than Create and Expires

R3222 *Any* **TIMESTAMP** *MUST NOT contain anything other than CREATED or EXPIRES elements.*

7.3 Constraints on Created and Expires

7.3.1 Value Precision to Milliseconds

The underlying specifications do not limit the resolution of timestamp values.

R3220 *Any* **CREATED** *SHOULD NOT contain a seconds value with more than three digits to the right of the decimal (milliseconds).*

R3229 *Any* **EXPIRES** *SHOULD NOT contain a seconds value with more than three digits to the right of the decimal (milliseconds).*

Since implementations have practical limits on resolution of time values the Profile requires a reasonable processing capability.

7.3.2 Leap Second Values Prohibited

Leap seconds are allowed by the underlying specifications.

R3213 *Any* **CREATED** *containing second values MUST specify seconds values less than 60.*

R3215 *Any* **EXPIRES** *containing second values MUST specify seconds values less than 60.*

Leap second processing is complex and error prone. The Profile disallows specification of leap seconds.

7.3.3 ValueType Attribute Prohibited

The underlying specifications allow for the specification of a timestamp ValueType.

R3225 *Any* **CREATED** *MUST NOT include a ValueType attribute.*

R3226 *Any* **EXPIRES** *MUST NOT include a ValueType attribute.*

There is no specified set of values for the ValueType attribute so the Basic Security Profile 1.1 disallows its use.

7.3.4 UTC Format Mandatory

The underlying specifications allow for a variety of timestamp formats.

R3217 *Any* **CREATED** *MUST contain time values in UTC format as specified by the XML Schema type (dateTime).*

R3223 *Any* **EXPIRES** *MUST contain time values in UTC format as specified by the XML Schema type (dateTime).*

Limiting timestamp values to UTC time eliminates complexity.

8 Security Token References

Web Services Security: SOAP Message Security defines a wsse:SecurityTokenReference element for use in SOAP messages. The Profile places the following constraints on its use:

8.1 Content

8.1.1 Exactly One SecurityTokenReference Child Element

Web Services Security: SOAP Message Security allows for a single SecurityTokenReference to include multiple reference mechanisms to the same security token. The Profile requires that only one be used.

R3061 *A* **SECURITY_TOKEN_REFERENCE** *MUST provide exactly one token reference.*

Restricting the number of reference mechanisms reduces complexity.

8.2 TokenType Attribute

8.2.1 Value of TokenType Attribute

R3074 *Any wsse:11:TokenType Attribute in a* **SECURITY_TOKEN_REFERENCE** *MUST specify a value that a TokenType specified by a security token profile for the referenced SECURITY_TOKEN.*

Restricting the number of reference mechanisms reduces complexity.

8.3 Direct References

8.3.1 Direct Reference to Security Token Reference Prohibited

The only proper way to refer to an INTERNAL_SECURITY_TOKEN by Direct Reference (even one inside a STR_EMBEDDED) is to refer directly to the INTERNAL_SECURITY_TOKEN.

R3057 *Any* **STR_REFERENCE** *MUST NOT reference a SECURITY_TOKEN_REFERENCE.*

R3064 *Any* **STR_REFERENCE** *MUST NOT reference an STR_EMBEDDED.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the second
wsse:SecurityTokenReference element refers to the
          wsse:SecurityTokenReference with an wsu:Id of TheFirstSTR  -->
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
```

```
          </wsse:BinarySecurityToken>
          <wsse:SecurityTokenReference wsu:Id="TheFirstSTR">
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#TheFirstSTR'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
```

CORRECT:

```
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeCert">
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
```

8.3.2 Reference/@ValueType Attribute Mandatory

The ValueType attribute in a security token reference is optional and has no accepted default value. This creates ambiguity between implementations when it is missing. Furthermore, security tokens similarly have ValueType attributes, which creates the possibility of contradiction between the reference and the token. There is no accepted processing model to resolve this.

R3059 *Any* **STR_REFERENCE** *MUST specify a ValueType attribute with the exception of STR_REFERENCE pointing to a SAML_V2_0_TOKEN or a KERBEROS_TOKEN or an ENCRYPTED_KEY_TOKEN.*

R3058 *Any* **STR_REFERENCE** *ValueType attribute MUST contain a value for the referenced SECURITY_TOKEN specified by the corresponding security token profile.*

Requiring that Security Token References carry a ValueType attribute makes it clear what type of security token is being referenced enabling security token specific reference mechanisms and aiding in error detection.

8.3.3 Reference/@URI Attribute Mandatory

Web Services Security: SOAP Message Security treats the URI attribute as optional allowing for extensibility in the reference mechanism. However, the only fully specified mechanism which uses the Reference element requires a URI value.

R3062 *Any* **STR_REFERENCE** *MUST specify a URI attribute.*

Eliminating underspecified functionality removes complexity.

8.4 Key Name References

8.4.1 Key Name References Prohibited

Key Name References may be ambiguous.

R3027 *Any* **SECURITY_TOKEN_REFERENCE** *MUST NOT contain an STR_KEY_NAME.*

In any case where a security token would be referred to by Key Name, it would also be possible to refer to it by a more efficient and/or less ambiguous mechanism (e.g. Direct, Key Identifier and/or Issuer and Serial Number). Thus, the Basic Security Profile 1.1 disallows the use of Key Name References.

For example,

INCORRECT:

```
          <!-- This example is incorrect because it uses a ds:KeyName element
to refer to an X.509 certificate  -->
          <wsse:SecurityTokenReference>
          <ds:KeyName>CN=Security WG, OU=BSP, O=WS-I, C=US</ds:KeyName>
          </wsse:SecurityTokenReference>
```

8.5 Key Identifier References

8.5.1 KeyIdentifier/@ValueType Attribute Mandatory

Having an explicit ValueType removes ambiguity about the format of the KeyIdentifier. The Profile restricts the value to that specified in the security token profile that is associated with the security token. The ValueType attribute in a KeyIdentifier is optional. This can cause ambiguity when it is not explicitly stated. Furthermore, interoperability is discouraged if a ValueType is specified but does not correspond to the value associated with that token as stated in its security token profile.

R3054 *Any* **STR_KEY_IDENTIFIER** *MUST specify a ValueType attribute.*

R3063 *Any* **STR_KEY_IDENTIFIER** *ValueType attribute MUST contain a value specified within the security token profile associated with the referenced SECURITY_TOKEN.*

Having an explicit ValueType removes ambiguity about the format of the KeyIdentifier and enhances processing efficiency. The Profile restricts the value to that specified in the security token profile that is associated with the security token.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:KeyIdentifier
element is missing a ValueType attribute -->
          <wsse:SecurityTokenReference>
```

```
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">
          MIGfMa0GCSq
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

CORRECT:

```
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509SubjectKeyIdentifier" >
          MIGfMa0GCSq
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

8.5.2 KeyIdentifier/@EncodingType Attribute Mandatory

Base64Binary is the only encoding type specified by Web Services Security: SOAP Message Security. Explicit specification of attribute values simplifies XML processing requirements and as a general principle the Basic Security Profile 1.1 requires that attributes be explicitly specified rather than relying on default values.

R3070 *Any* **STR_KEY_IDENTIFIER** *that refers to a SECURITY_TOKEN other than a SAML_TOKEN MUST specify an EncodingType attribute.*

R3071 *Any* **STR_KEY_IDENTIFIER** *EncodingType attribute MUST have a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary".*

A wsse:KeyIdentifier may specify its encoding type. The Profile restricts the encoding type to Base64Binary and requires its explicit specification.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:KeyIdentifier
element is missing an EncodingType attribute -->
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509SubjectKeyIdentifier" >
          MIGfMa0GCSq
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

CORRECT:

```
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509SubjectKeyIdentifier" >
          MIGfMa0GCSq
```

```
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

8.6 Embedded References

8.6.1 Embedded Content

Embedded elements may potentially contain multiple elements, creating ambiguity about which token should be processed. Furthermore, elements may be of a type that is not defined within a security token profile. This can cause problems with interoperability.

R3060 *Any* **STR_EMBEDDED** *MUST contain only a single child element which is an INTERNAL_SECURITY_TOKEN.*

In order to reduce ambiguity surrounding which token to process, the Basic Security Profile 1.1 restricts embedded security tokens to contain exactly one security token element. It also restricts tokens to those defined in a token profile; this establishes a defined scope of profiles and thus allows for interoperability between implementations.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:Embedded element
has multiple element children -->
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeCerts">
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <wsse:BinarySecurityToken wsu:Id='SomeOtherCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
```

CORRECT:

```
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeCert">
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
```

```
lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
```

8.6.2 Embedded Token Format

Using a single consistent format for security tokens, regardless of reference mechanism, ensures consistent processing.

R3025 *Any* **INTERNAL_SECURITY_TOKEN** *contained in an STR_EMBEDDED MUST be in the same format as if it were a child of a SECURITY_HEADER.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:Embedded element
carries the data for the X.509 certificate
          directly rather than as a wsse:BinarySecurityToken element  -->
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="SomeCert">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
```

CORRECT:

```
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeCert">
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
```

8.6.3 Security Token Reference in Embedded Prohibited

Embedded elements can contain multiple binary security token elements, which creates ambiguity about which token should be processed. Furthermore, the security token may be a type that is not defined within a security token profile. This can cause problems with interoperability. Using a single consistent format for security tokens, regardless of reference mechanism, ensures consistent processing. Embedded security tokens can potentially chain to other security tokens, which adds complexity to processing and discourages interoperability.

R3056 *Any* **STR_EMBEDDED** *MUST NOT contain a wsse:SecurityTokenReference child element.*

In order to reduce ambiguity surrounding which token to process, the Basic Security Profile 1.1 restricts embedded security tokens to contain exactly one security token element. It also restricts tokens to those defined in a token profile; this establishes a defined scope of profiles and thus allows for interoperability between implementations. Eliminating redirection from within embedded elements reduces required complexity in handling embedded security tokens.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:Embedded element
contains a wsse:SecurityTokenReference element -->
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeSTR">
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
```

8.7 Internal References

8.7.1 Direct or Embedded References Where Possible

Web Services Security: SOAP Message Security provides a list of reference mechanisms in preferred order (i.e., most specific to least specific). This adds ambiguity and complexity, which discourages interoperability.

R3022 *Any* **SECURITY_TOKEN_REFERENCE** *that references an INTERNAL_SECURITY_TOKEN which has a wsu:Id attribute MUST contain an STR_REFERENCE or STR_EMBEDDED.*

The recommendation does not allow the use of Key Identifier and Key Name references due to possible ambiguities. Direct References and Embedded References are to be used instead of these. This reduces complexity and improves interoperability.

For example,

INCORRECT:

```
          <!-- This example is incorrect because it refers to a
wsse:BinarySecurityToken element which specifies a wsu:id
          attribute using a wsse:KeyIdentifier element rather than a
wsse:Reference or wsse:Embedded element -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
```

```
lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509SubjectKeyIdentifier">
          MIGfMa0GCSq
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
          </wsse:Security>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </wsse:Security>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeCert">
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Embedded>
```

```
          </wsse:SecurityTokenReference>
          </wsse:Security>
```

8.7.2 Direct Preferred to Embedded References

Since multiple security elements may reference a single security token and processing of those elements may result in the removal of the element, consistent use of direct rather than embedded references simplifies processing. Direct references are encouraged, embedded references are discouraged.

R3023 *Any* **SECURITY_TOKEN_REFERENCE** *that references an INTERNAL_SECURITY_TOKEN that is referenced several times SHOULD contain an STR_REFERENCE rather than an STR_EMBEDDED.*

The Profile encourages the consistent use of Direct Reference to security tokens.

For example,

INCORRECT:

```
          <!-- This example is incorrect because it uses a wsse:Embedded
element for the wsse:BinarySecurityToken
          with the wsu:Id of SomeCert. It is assumed that this token is
referred to from several places elsewhere
          in the SOAP envelope (not shown)  -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:SecurityTokenReference>
          <wsse:Embedded wsu:Id="TheEmbeddedElementAroundSomeCert">
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Embedded>
          </wsse:SecurityTokenReference>
          </wsse:Security>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
```

```
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </wsse:Security>
```

8.7.3 Shorthand XPointers Mandatory for Direct References

Constraining the number of referencing mechanisms reduces complexity and thus improves interoperability. The wsse:BinarySecurityToken has a wsu:Id attribute allowing references to this token to use the relatively efficient and unambiguous Shorthand XPointer.

R5204 *Any* **STR_REFERENCE** *to an INTERNAL_SECURITY_TOKEN having an ID attribute MUST contain a URI attribute with a Shorthand XPointer value.*

The Profile requires the use of Shorthand XPointer Reference to ensure that the URI efficiently references the correct token.

For example,

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
```

```
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedKey>
          </wsse:Security>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <rel:license xmlns:rel='urn:mpeg:mpeg21:2003:01-REL-R-NS'
          wsu:Id='SomeLic'
          licenseId='uuid:3D680C71-177B-40cc-84C1-123B02503524' >
          . . .
          </rel:license>
          <ds:Signature>
          . . .
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeLic'
          ValueType="http://docs.oasisopen.org/wss/oasis-wss-rel-token-
profile-1.0.pdf#license" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
```

8.7.4 Security Tokens Precede Their References

Security token references are intended to provide access to security tokens residing anywhere in a document. However, token placement can have a significant affect on processing efficiency when the document is processed in a stream-oriented fashion. For example, resolving a forward reference to a token may require significant subsequent document parsing that could otherwise be eliminated. This need to satisfy random access to security tokens adds complexity to implementations that works against interoperability.

R5205 *Any* **INTERNAL_SECURITY_TOKEN** *that is not contained in an STR_EMBEDDED MUST precede all SECURITY_TOKEN_REFERENCE elements that reference it in the SOAP_ENVELOPE.*

Ensuring that a security token element appears before it is referenced, when processing in document order, means that implementations have access to the token content referenced from a wsse:SecurityTokenReference element when it is needed to verify a signature or perform decryption.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:BinarySecurityToken
with the wsu:Id of SomeCert appears after it is
          referenced from within the xenc:EncryptedKey element -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
```

```
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedKey>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          </wsse:Security>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"/>
```

```
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedKey>
          </wsse:Security>
```

8.7.5 References Between Security Headers Prohibited

The potential exists for the same security token to be referenced from multiple security headers.

R3066 *Any* **STR_REFERENCE** *that is a descendant of a SECURITY_HEADER MUST NOT use a Shorthand XPointer to refer to an INTERNAL_SECURITY_TOKEN located in a SECURITY_HEADER other than the SECURITY_HEADER that contains the STR_REFERENCE.*

R3067 *Any* **STR_REFERENCE** *that is a descendant of an ENCRYPTED_DATA MUST NOT use a Shorthand XPointer to refer to an INTERNAL_SECURITY_TOKEN located in a SECURITY_HEADER other than the SECURITY_HEADER containing a reference (EK_REFERENCE_LIST or an ENC_REFERENCE_LIST) to the ENCRYPTED_DATA.*

Since processing of security header elements can result in the removal of those elements, references to elements in another header may not correctly resolve. If an internal security token is referenced from multiple security headers it should be copied into each referencing header.

8.8 External References

8.8.1 Direct References Where Possible

Since multiple security elements may reference a single external security token, consistent use of direct references simplifies processing. Direct references are encouraged.

R3024 *Any* **EXTERNAL_TOKEN_REFERENCE** *that can use an STR_REFERENCE MUST contain an STR_REFERENCE.*

The Profile encourages the use of Direct Reference in order to minimize ambiguity.

For example,

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='http://www.ws-
i.org/CertStore/Examples/BSP.PEM'
```

```
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </wsse:Security>
```

8.9 SecurityTokenReference With EncryptedData

8.9.1 Reference to KeyInfo Prohibited

Security token references are intended to refer directly to security tokens. This requirement prohibits cases where a reference is made to a ds:KeyInfo which in turn contains another reference.

R3211 *Any* **SECURITY_TOKEN_REFERENCE** *MUST NOT reference a ds:KeyInfo element.*

9 XML-Signature

Web Services Security: SOAP Message Security builds on XML Signature, defining usage of various elements from XML Signature and a processing model. The Profile places the constraints defined in this section on the use of XML Signature with Web Services Security: SOAP Message Security. The Profile places no constraints on other use of XML Signature.

In some areas the Basic Security Profile allows limited flexibility and extensibility in the application of security to messages. Some agreement between the SENDER and RECEIVER over which mechanisms and choices should be used for message exchanges is necessary. Since no security policy description language or negotiation mechanism is in scope for the Basic Security Profile, some out of band agreement must be reached for which elements should be signed and which signature algorithms should be used.

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- XML Signature Syntax and Processing [xmldsig]

The following extensibility point is defined:

- Extensibility points:
  - E0014 – Signature Digest algorithm – version of SHA algorithm in use (SHA-1, or one of the SHA-2 algorithms).

9.1 Types of Signature

9.1.1 Enveloping Signatures Prohibited

Due to the nature of the SOAP processing model, which is based on recognizing the elements that are children of soap:Header and/or soap:Body, use of enveloping signatures, where the signed XML is encapsulated in a ds:Signature element, is inappropriate.

R3102 *A* **SIGNATURE** *MUST NOT be an Enveloping Signature as defined by the XML Signature specification.*

Enveloping signatures are not allowed.

For example,

INCORRECT:

```
          <!-- This example is incorrect because it contains an enveloping
signature around the SomeSecurityToken element -->
          <ds:Signature Id='TheSig'
xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:Reference URI='#SigPropBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
```

```
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>i3qi5GjhHnfoBn/jOjQp2mq0Na4=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>oxNwoqGbzqg1YBliz+PProgcjw8=</ds:SignatureValue>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <ds:Object>
          <ds:SignatureProperties>
          <ds:SignatureProperty Id='SigPropBody' Target='#TheSig'>
          <SomeSecurityToken/>
          </ds:SignatureProperty>
          </ds:SignatureProperties>
          </ds:Object>
          </ds:Signature>
```

9.1.2 Enveloped Signatures Discouraged

Enveloped signatures, where the ds:Signature element is a descendant of the signed element, limit the ability of intermediaries to process messages and should be avoided unless said limitation is the desired effect.

R3104 *A* **SIGNATURE** *SHOULD NOT be an Enveloped Signature as defined by the XML Signature specification.*

Enveloped signatures are discouraged.

9.1.3 Detached Signatures Preferred

R3103 *A* **SIGNATURE** *SHOULD be a Detached Signature as defined by the XML Signature specification.*

Detached signatures are encouraged.

For example,

CORRECT:

```
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>i3qi5GjhHnfoBn/jOjQp2mq0Na4=</ds:DigestValue>
```

```
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>PipXJ2Sfc+LTDnq4pM5JcIYt9gg=</ds:SignatureValue>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
```

9.2 Signed Element References

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- XPointer Framework [XPointer]

Element references are used to specify which portions of a SECURE_ENVELOPE are integrity protected. The Basic Security Profile 1.1 places the following constraints on the use of element references:

9.2.1 Shorthand XPointer Where Referent has wsu:Id Attribute

Shorthand XPointer is relatively efficient and interoperable. However, in cases where the referent element is optional in the message and other remedies are unacceptable, it may be necessary to use an absolute path XPath Expression which allows signature verification to detect movement of the signed element within the message (see Section 18.4 for details).

R3001 *Any* **SIG_REFERENCE** *SHOULD contain a URI attribute containing a Shorthand XPointer.*

9.2.2 Shorthand XPointer Where Referent is defined by XML Signature

R3003 *Any* **SIG_REFERENCE** *to a SIGNATURE or descendant of a SIGNATURE MUST contain a URI attribute with a reference value that is a Shorthand XPointer to Local ID attribute defined by XML Signature.*

9.2.3 Shorthand XPointer Where Referent is defined by XML Encryption

R3004 *Any* **SIG_REFERENCE** *to an element defined in XML Encryption MUST contain a URI attribute with a reference value that is a Shorthand XPointer to Local ID attribute defined by XML Encryption.*

9.2.4 Shorthand XPointer to wsu:Id Attribute Where Possible

Processing of Shorthand XPointers requires knowledge of which attributes are IDs. Since the underlying specifications strive to allow message processing without schema processing, some non-schema aware method for identifying ID attributes must be used.

R3005 *Any* **SIG_REFERENCE** *to an element that is not defined in XML Encryption, a SIGNATURE, or a descendant of a SIGNATURE*

SHOULD contain a URI attribute with a reference value that is a Shorthand XPointer to a wsu:Id attribute.

The underlying specifications define well known ID attributes. Limiting references to those well known attributes reduces complexity and the reliance on schema processing.

9.2.5 XPath References Where Necessary

Elements that do not have an attribute of type ID cannot be referred to by Shorthand XPointer so a different referencing mechanism is needed.

R3002 *Any* **SIG_REFERENCE** *to an element that does not have an ID attribute MUST contain a TRANSFORM with an Algorithm attribute value of "http://www.w3.org/2002/06/xmldsig-filter2".*

The XPath Filter 2.0 transform is more efficient than the original XPath transform from XML Digital Signature Syntax and Processing.

For example,

INCORRECT:

```
          <!-- This example is incorrect because it uses the
http://www.w3.org/TR/1999/REC-xpath-19991116 transform instead of the
http://www.w3.org/2002/06/xmldsig-filter2 transform -->
          <soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope' >
          <soap:Header>
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI=''>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/TR/1999/REC-xpath-
19991116'>
          <ds:XPath>ancestor-or-
self::soap:Body[parent::node()=/soap:Envelope]</ds:XPath>
          </ds:Transform>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>VEPKwzfPGOxh2OUpoK0bc158jtU=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
```

```
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
          </soap:Header>
          <soap:Body xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd'
          wsu:Id='TheBody'>
          <m:SomeElement xmlns:m='http://example.org/ws' />
          </soap:Body>
          </soap:Envelope>
```

CORRECT:

```
          <soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope' >
          <soap:Header>
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI=''>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2002/06/xmldsig-filter2'
          xmlns:dsxp='http://www.w3.org/2002/06/xmldsig-filter2'>
          <dsxp:XPath Filter='intersect'>ancestor-or-
self::soap:Body[parent::node()=/soap:Envelope]</dsxp:XPath>
          </ds:Transform>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'>
          <xc14n:InclusiveNamespaces
xmlns:xc14n='http://www.w3.org/2001/10/xml-exc-c14n#' />
          </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>VEPKwzfPGOxh2OUpoK0bc158jtU=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
```

```
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
          </soap:Header>
          <soap:Body xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd'
          wsu:Id='TheBody'>
          <m:SomeElement xmlns:m='http://example.org/ws' />
          </soap:Body>
          </soap:Envelope>
```

9.3 Signature Transforms

9.3.1 Transforms Element Mandatory

At a minimum an XML Canonicalization Algorithm needs to be specified for each Reference, necessitating a ds:Transforms element.

R5416 *Any* **SIG_REFERENCE** *MUST contain a SIG_TRANSFORMS child element.*

9.3.2 Transform Element Mandatory

R5411 *Any* **SIG_TRANSFORMS** *MUST contain at least one SIG_TRANSFORM child element.*

9.3.3 Transform Algorithms

These algorithms are chosen for their cryptographic strength, utility or because they address some security concern.

R5423 *Any* **SIG_TRANSFORM** *Algorithm attribute MUST have a value of*

"http://www.w3.org/2001/10/xml-exc-c14n#" or

"http://www.w3.org/2002/06/xmldsig-filter2" or

"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform" or

"http://www.w3.org/2000/09/xmldsig#enveloped-signature" or

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Signature-Transform" or

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete-Signature-Transform"

For example,

CORRECT:

```
          <soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope' >
          <soap:Header>
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedKey xmlns:xenc='http://www.w3.org/2001/04/xmlenc#' >
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedKey>
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#hmac-sha1' />
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>+VTJraRYFT3pl7Z4uAWhmr5+bf4=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
          </ds:Signature>
          </wsse:Security>
          </soap:Header>
          <soap:Body xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd'
          wsu:Id='TheBody'>
          <m:SomeElement xmlns:m='http://example.org/ws' />
          </soap:Body>
```



```
          </soap:Envelope>
```

CORRECT:

```
          <soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope' >
          <soap:Header>
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>+VTJraRYFT3pl7Z4uAWhmr5+bf4=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
          </soap:Header>
          <soap:Body xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd'
          wsu:Id='TheBody'>
          <m:SomeElement xmlns:m='http://example.org/ws' />
          </soap:Body>
          </soap:Envelope>
```

9.3.4 Last Transform Algorithm

Canonicalization is critical to ensuring signatures are processed correctly, thus each ds:Reference will need at least one ds:Transform to specify the Exclusive C14N transform or a transform which itself incorporates Exclusive C14N.

R5412 *Any* **SIG_TRANSFORMS** *MUST contain as its last child a SIG_TRANSFORM with an Algorithm attribute with a value of*

"http://www.w3.org/2001/10/xml-exc-c14n#" or

"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform" or

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Signature-Transform" or

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete-Signature-Transform".

For example,

INCORRECT:

```
          <!-- This example is incorrect because the ds:Reference element
does not have a ds:Transforms child element -->
          <ds:Reference URI='#TheBody'>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>VEPKwzfPGOxh2OUpoK0bcl58jtU=</ds:DigestValue>
          </ds:Reference>
```

CORRECT:

```
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'/>
          </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>+VTJraRYFT3pl7Z4uAWhmr5+bf4=</ds:DigestValue>
          </ds:Reference>
```

9.3.5 Inclusive Namespaces with Exclusive-C14N Transform

R5407 *Any* **SIG_TRANSFORM** *with an Algorithm attribute with a value of "http://www.w3.org/2001/10/xml-exc-c14n#" MUST contain an INCLUSIVE_NAMESPACES with an PrefixList attribute unless the PrefixList is empty.*

9.3.6 Inclusive Namespaces with STR Transform

R5413 *Any* **SIG_TRANSFORM** *with an Algorithm attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform" MUST have an*

INCLUSIVE_NAMESPACES with an PrefixList attribute unless the PrefixList is empty.

9.3.7 TransformationParameters and CanonicalizationMethod with STR Transform

The Security Token Dereferencing Transform allows for the optional specification of a canonicalization algorithm.

R3065 *Any* **SIG_TRANSFORM** *with an Algorithm attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform" MUST contain a child wsse:TransformationParameters element containing a child ds:CanonicalizationMethod element.*

Consistent processing of data to be signed, including security token content, reduces complexity.

9.4 Canonicalization Methods

9.4.1 Exclusive C14N Mandatory

R5404 *Any* **CANONICALIZATION_METHOD** *Algorithm attribute MUST have a value of "http://www.w3.org/2001/10/xml-exc-c14n#" indicating that it uses Exclusive C14N without comments for canonicalization.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because it uses the
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
          canonicalization algorithm -->
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/TR/2001/REC-xml-c14n-20010315' />
```

9.4.2 Inclusive Namespaces with Exclusive-C14N

R5406 *Any* **CANONICALIZATION_METHOD** *MUST contain an INCLUSIVE_NAMESPACES with a PrefixList attribute unless the PrefixList is empty.*

9.5 Inclusive Namespaces

9.5.1 Order of PrefixList

R5414 *A* **RECEIVER** *MUST be capable of accepting and processing an INCLUSIVE_NAMESPACES PrefixList attribute containing prefixes in any order within the string.*

9.5.2 Whitespace in PrefixList

R5415 *A* **RECEIVER** *MUST be capable of accepting and processing an INCLUSIVE_NAMESPACES PrefixList attribute containing*

arbitrary whitespace before, after and between the prefixes within the string.

9.5.3 PrefixList Contents

Unless proper canonicalization is performed, verification of signatures may not work due to changes to the elements in the containing scope.

R5405 *Any* **INCLUSIVE_NAMESPACES** *MUST contain the prefix of all namespaces that are in-scope and desired to be protected, but not visibly utilized, for the element being signed and its descendants, per Exclusive XML Canonicalization Version 1.0.* C

R5408 *Any* **INCLUSIVE_NAMESPACES** *MUST contain the string "#default" if a default namespace is in-scope and desired to be protected, but not visibly utilized, for the element being signed and its descendants, per Exclusive XML Canonicalization Version 1.0.*

The use of Exclusive Canonicalization with xc14n:InclusiveNamespaces/@Prefix addresses problems with both Inclusive Canonicalization and Exclusive Canonicalization without xc14n:InclusiveNamespaces/@Prefix.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the PrefixList of the
xc14n:InclusiveNamespaces element does not contain the
          correct list of prefixes. It should contain the bar prefix. -->
          <soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope'>
          <soap:Header>
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'>
          <xc14n:InclusiveNamespaces
xmlns:xc14n='http://www.w3.org/2001/10/xml-exc-c14n#' />
          </ds:Transform>
          </ds:Transforms>
```

```
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>VEPKwzfPGOxh2OUpoK0bcl58jtU=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
          </soap:Header>
          <soap:Body xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd'
          wsu:Id='TheBody'
          xmlns:bar="http://bar.com">
          <m:SomeElement xmlns:m='http://example.org/ws' foo='bar:none'/>
          </soap:Body>
          </soap:Envelope>
```

CORRECT:

```
          <soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope'>
          <soap:Header>
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'>
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'>
          <xc14n:InclusiveNamespaces
xmlns:xc14n='http://www.w3.org/2001/10/xml-exc-c14n#' PrefixList='bar' />
          </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>VEPKwzfPGOxh2OUpoK0bcl58jtU=</ds:DigestValue>
          </ds:Reference>
```

```
          </ds:SignedInfo>
          <ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
          </soap:Header>
          <soap:Body xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd'
          wsu:Id='TheBody'
          xmlns:bar="http://bar.com">
          <m:SomeElement xmlns:m='http://example.org/ws' foo='bar:none'/>
          </soap:Body>
          </soap:Envelope>
```

9.6 Digest Methods

9.6.1 Use of SHA-1 Preferred

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines the proper URI to be used in the DIGEST_METHOD attribute. The following requirement applies when SHA-1 is used:

R5420 *If SHA-1 is used, the* **DIGEST_METHOD** *Algorithm attribute MUST have the value "http://www.w3.org/2000/09/xmldsig#sha1".*

9.7 Signature Methods

9.7.1 Algorithms

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines the proper URI to be used in the SIGNATURE_METHOD attribute. The following requirement applies when SHA-1 is used:

R5421 *If SHA-1 is used, the* **SIGNATURE_METHOD** *Algorithm attribute SHOULD have a value of "http://www.w3.org/2000/09/xmldsig#hmac-sha1" or "http://www.w3.org/2000/09/xmldsig#rsa-sha1".*

9.7.2 HMACOutputLength Prohibited

The ds:HMACOutputLength provides an input parameter to the HMAC-SHA1 algorithm specifying how many bits of the output to use. Disallowing use of this element results in ALL the bits of the output being used.

R5401 *Any* **SIGNATURE_METHOD** *MUST NOT contain a ds:HMACOutputLength child element.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the ds:SignatureMethod
element has a ds:HMACOutputLength child element -->
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#hmac-sha1'>
          <ds:HMACOutputLength>128</ds:HMACOutputLength>
          </ds:SignatureMethod>
```

CORRECT:

```
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#hmac-sha1' />
```

9.8 KeyInfo

9.8.1 Exactly One KeyInfo Child Element

R5402 *Any* **SIG_KEY_INFO** *MUST contain exactly one child element.*

9.8.2 SecurityTokenReference Mandatory

The ds:KeyInfo element allows for many different child elements. The Profile mandates a single element, wsse:SecurityTokenReference, which is needed to reference security tokens.

R5417 *Any* **SIG_KEY_INFO** *MUST contain a SECURITY_TOKEN_REFERENCE child element.*

For example,

CORRECT:

```
          <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
```

9.9 Manifest

9.9.1 Manifest Prohibited

The ds:Manifest element is designed for specific application level use cases that do not apply to the use of XML Signature in SOAP Message Security.

R5403 *A* **SIGNATURE** *MUST NOT contain a ds:Manifest descendant element.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the ds:Signature element has
a ds:Manifest grandchild element -->
          <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
```

```
          <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
          <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI='#TheManifest'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>OVuYKGY6KCGB0l0XHS3krj8vjek=</ds:DigestValue>
          </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>L7X0Zw23/zYQnX4+Z+p0gCygKQ0=</ds:SignatureValue>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <ds:Object>
          <ds:Manifest Id='TheManifest'>
          <ds:Reference URI='#TheBody'>
          <ds:Transforms>
          <ds:Transform Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'
/>
          <ds:DigestValue>+VTJraRYFT3pl7Z4uAWhmr5+bf4=</ds:DigestValue>
          </ds:Reference>
          </ds:Manifest>
          </ds:Object>
          </ds:Signature>
```

9.10 Signature Encryption

9.10.1 Encrypt Only Entire Signature

If the value of a ds:DigestValue element in a SIGNATURE needs to be encrypted the entire parent ds:Signature element MUST be encrypted.

R5440 *A* **SIGNATURE** *MUST NOT have any xenc:EncryptedData elements amongst its descendants.*

9.11 Signature Confirmation

9.11.1 Signature Confirmation Format

R5441 *A* **SIGNATURE_CONFIRMATION** *MUST contain a wsu:Id attribute.*

10 XML Encryption

Web Services Security: SOAP Message Security builds on XML Encryption, defining usage of various elements from XML Encryption and a processing model. The Basic Security Profile 1.1 places the constraints defined in this section on the use of XML Encryption with Web Services Security: SOAP Message Security. The Basic Security Profile 1.1 places no constraints on other use of XML Encryption.

In some areas the Basic Security Profile allows limited flexibility and extensibility in the application of security to messages. Some agreement between the SENDER and RECEIVER over which mechanisms and choices should be used for message exchanges is necessary. Since no security policy description language or negotiation mechanism is in scope for the Basic Security Profile, some out of band agreement must be reached for which elements should be encrypted and which data encryption, key transport and/or key wrap algorithms should be used.

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- XML Encryption Syntax and Processing [xmlenc]

10.1 EncryptedHeader

10.1.1 EncryptedHeader Format

WSS 1.1 introduced a new ENCRYPTED_HEADER mechanism to encrypt headers. When it is required that an entire SOAP header block including the top-level element and its attributes be encrypted, the original header block is replaced with an ENCRYPTED_HEADER. Where an ENCRYPTED_HEADER element exists, it contains a child ENCRYPTED_DATA element that is the result of encrypting the header block.

R3228 *A soap:Header element in a* **SECURE_ENVELOPE** *MUST NOT contain any child ENCRYPTED_DATA.*

R3299 *A soap:Header element in a* **SECURE_ENVELOPE** *MAY contain ENCRYPTED_HEADER children.*

R3230 *An ENCRYPTED_HEADER MUST NOT contain any children other than a single required ENCRYPTED_DATA.*

R3232 *In cases where a wsu:Id does exist on the ENCRYPTED_HEADER, the child* **ENCRYPTED_DATA** *MAY contain an Id attribute.*

10.2 Encryption ReferenceList

10.2.1 Single Key

R3205 *Any* **ENC_REFERENCE_LIST** *produced as part of an encryption step MUST use a single key.*

10.2.2 Encryption DataReference for EncryptedData

R3231 *Any* **ENC_REFERENCE_LIST** *MUST contain an xenc:DataReference element for each ENCRYPTED_DATA produced in the associated encryption step.*

10.3 EncryptedKey ReferenceList

10.3.1 EncryptedKey DataReference for EncryptedData

Some encryption steps might not produce an xenc:ReferenceList. For those that do produce an xenc:ReferenceList, there must be a separate xenc:ReferenceList for each such encryption step. When there is a xenc:ReferenceList either as a child of wsse:Security or as a child of xenc:EncryptedKey it must list all the corresponding xenc:EncryptedData elements by using xenc:DataReference elements.

R3214 *Any* **EK_REFERENCE_LIST** *MUST contain a xenc:DataReference for each ENCRYPTED_DATA produced in the associated encryption step.*

10.4 EncryptedKey

10.4.1 EncryptedKey Precedes EncryptedData

To facilitate ease of processing, keys are required to appear inside wsse:Security headers and to appear before they are required for decryption of elements inside a wsse:Security header.

R3208 *Any* **ENCRYPTED_KEY** *MUST precede any ENCRYPTED_DATA in the same SECURITY_HEADER referenced by the associated EK_REFERENCE_LIST.*

For example,

INCORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedData Id='Enc1'>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc' />
          <xenc:CipherData>
          <xenc:CipherValue>

9jFtYcLSlDZQBMjKfT7ctg6Jy+6sC8YORhiPeTvOjug7ozY2SRHGuLt8G/vf2f/f4IdF0ewiDOpq.
..
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedData>
```

```
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI='#Enc1' />
          </xenc:ReferenceList>
          </xenc:EncryptedKey>
          </wsse:Security>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI='#Enc1' />
          </xenc:ReferenceList>
          </xenc:EncryptedKey>
          <xenc:EncryptedData Id='Enc1'>
```

```
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc' />
          <xenc:CipherData>
          <xenc:CipherValue>

9jFtYcLSlDZQBMjKfT7ctg6Jy+6sC8YORhiPeTvOjug7ozY2SRHGuLt8G/vf2f/f4IdF0ewiDOpq.
..
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedData>
          </wsse:Security>
```

10.4.2 EncryptedKey/@Type Attribute Prohibited

R3209 *Any* **ENCRYPTED_KEY** *MUST NOT specify a Type attribute.*

10.4.3 EncryptedKey/@MimeType Attribute Prohibited

R5622 *Any* **ENCRYPTED_KEY** *MUST NOT specify a MimeType attribute.*

10.4.4 EncryptedKey/@Encoding Attribute Prohibited

These prohibited attributes are not needed for xenc:EncryptedKey elements used to secure SOAP messages.

R5623 *Any* **ENCRYPTED_KEY** *MUST NOT specify a Encoding attribute.*

10.4.5 EncryptedKey/@Recipient Attribute Prohibited

This attribute is prohibited because the soap:actor attribute conveys the same information.

R5602 *Any* **ENCRYPTED_KEY** *MUST NOT contain a Recipient attribute.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the xenc:EncryptedKey
element has a Recipient attribute  -->
          <xenc:EncryptedKey Recipient='Bert'>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
```

```
          <xenc:DataReference URI='#Enc1' />
          </xenc:ReferenceList>
          </xenc:EncryptedKey>
```

10.4.6 EncryptionMethod Mandatory

Specifying the encryption algorithm used to perform the encryption makes messages more self-describing and aids interoperability.

R5603 *Any* **ENCRYPTED_KEY** *MUST contain an xenc:EncryptionMethod child element.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the xenc:EncryptedKey
element is missing an xenc:EncryptionMethod child element -->
          <xenc:EncryptedKey>
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI='#Enc1' />
          </xenc:ReferenceList>
          </xenc:EncryptedKey>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
```

```
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI='#Enc1' />
          </xenc:ReferenceList>
          </xenc:EncryptedKey>
          <xenc:EncryptedData Id='Enc1'>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc' />
          <xenc:CipherData>
          <xenc:CipherValue>

9jFtYcLSlDZQBMjKfT7ctg6Jy+6sC8YORhiPeTvOjug7ozY2SRHGuLt8G/vf2f/f4IdF0ewiDOpq.
..
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedData>
          </wsse:Security>
```

10.5 EncryptedData

10.5.1 EncryptedData and KeyInfo

The ds:KeyInfo element is useful for determining the security token with which the relevant key material is associated.

R5629 *An* **ENCRYPTED_DATA** *which is not referenced from an ENCRYPTED_KEY MUST contain a ds:KeyInfo.*

10.5.2 EncryptedData/@Id or EncryptedHeader/@wsu:Id Attribute Mandatory

xenc:EncryptedData and wsse11:EncryptedHeader elements are referred to from an xenc:Reference list. Such references use shorthand XPointers.

R5624 *In cases where a wsu:Id does not exist on the ENCRYPTED_HEADER, the child* **ENCRYPTED_DATA** *MUST contain an ID attribute.*

R5627 *In cases where an ID does not exist on the ENCRYPTED_DATA, the parent* **ENCRYPTED_HEADER** *MUST contain a wsu:Id attribute.*

10.5.3 EncryptedData EncryptionMethod Mandatory

R5601 *Any* **ENCRYPTED_DATA** *MUST contain an xenc:EncryptionMethod child element.*

For example,

INCORRECT:

```
          <!-- This example is incorrect because the xenc:EncryptedData
element is missing an xenc:EncryptionMethod child element -->
          <xenc:EncryptedData Id='Enc1'>
          <xenc:CipherData>
          <xenc:CipherValue>

9jFtYcLSlDZQBMjKfT7ctg6Jy+6sC8YORhiPeTvOjug7ozY2SRHGuLt8G/vf2f/f4IdF0ewiDOpq.
..
          </xenc:CipherValue>
          </xenc:CipherData>
          </xenc:EncryptedData>
```

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
          <xenc:EncryptedKey>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#rsa-1_5' />
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>
          XZEEVABD3L9G+VNTCDiDTE7WB1a4kILtz5f9FT747eE=
          </xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI='#Enc1' />
          </xenc:ReferenceList>
          </xenc:EncryptedKey>
          <xenc:EncryptedData Id='Enc1'>
          <xenc:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc' />
          <xenc:CipherData>
          <xenc:CipherValue>

9jFtYcLSlDZQBMjKfT7ctg6Jy+6sC8YORhiPeTvOjug7ozY2SRHGuLt8G/vf2f/f4IdF0ewiDOpq.
..
```

```
        </xenc:CipherValue>
        </xenc:CipherData>
        </xenc:EncryptedData>
        </wsse:Security>
```

10.6 Encryption KeyInfo

10.6.1 Exactly One Encryption KeyInfo Child Element

R5424 *Any* **ENC_KEY_INFO** *MUST have exactly one child element.*

10.6.2 KeyInfo SecurityTokenReference Mandatory

The ds:KeyInfo element allows for many different child elements. The Profile mandates a single element, wsse:SecurityTokenReference, which is needed to reference security tokens.

R5426 *Any* **ENC_KEY_INFO** *MUST contain a child SECURITY_TOKEN_REFERENCE.*

10.7 Encryption DataReference

10.7.1 DataReference/@URI with Shorthand XPointer to EncryptedData or EncryptedHeader

R5608 *Any* **ENC_DATA_REFERENCE** *MUST contain a URI attribute containing a Shorthand XPointer reference value based on either the Id attribute of the referenced ENCRYPTED_DATA or the wsu:Id attribute of the referenced ENCRYPTED_HEADER.*

10.8 EncryptedKey DataReference

10.8.1 EncryptedKey DataReference/@URI with Shorthand XPointer to EncryptedData

R3006 *Any* **EK_DATA_REFERENCE** *MUST contain a URI attribute containing a Shorthand XPointer reference value based on either the Id attribute of the referenced ENCRYPTED_DATA or the wsu:Id attribute of the referenced ENCRYPTED_HEADER.*

10.9 Encryption KeyReference

10.9.1 KeyReference/@URI with Shorthand XPointer to EncryptedKey

R5613 *Any* **ENC_KEY_REFERENCE** *MUST contain a URI attribute containing a Shorthand XPointer reference value based on the Id attribute of the referred to ENCRYPTED_KEY.*

10.10 EncryptedKey KeyReference

10.10.1 EncryptedKey KeyReference/@URI with Shorthand XPointer to EncryptedKey

The above requirements ensure that Shorthand XPointer References are used where appropriate and that they refer to the correct attribute values.

R3007 *Any* **EK_KEY_REFERENCE** *MUST contain a URI attribute containing a Shorthand XPointer reference value based on the Id attribute of the referred to ENCRYPTED_KEY.*

10.11 EncryptedData EncryptionMethod

10.11.1 Data Encryption Algorithms

Data encryption algorithms are used for encrypting elements and element content. Industries, organizations, and application domains are currently choosing from a variety of data encryption algorithms based on reasons including performance, security characteristics, and regulatory compliance. A set of the most commonly chosen and widely deployed data encryption algorithms are supported by the Basic Security Profile in order to avoid disenfranchising existing applications. At some point in the future, if and when consensus is reached for a single data encryption algorithm the Basic Security Profile 1.1 may be revised to constrain instances to use only that algorithm.

R5620 *Any* **ED_ENCRYPTION_METHOD** *Algorithm attribute MUST have a value of*

"http://www.w3.org/2001/04/xmlenc#tripledes-cbc", or

"http://www.w3.org/2001/04/xmlenc#aes128-cbc" or

"http://www.w3.org/2001/04/xmlenc#aes256-cbc"

The 3DES algorithm ("http://www.w3.org/2001/04/xmlenc#tripledes-cbc") is widely implemented and deployed in existing practice. The AES algorithm is relatively new and becoming widely implemented and deployed. The 128-bit variation of AES ("http://www.w3.org/2001/04/xmlenc#aes128-cbc") is relatively faster but weaker than the 256-bit variation ("http://www.w3.org/2001/04/xmlenc#aes256-cbc").

10.12 EncryptedKey EncryptionMethod

10.12.1 Key Transport Algorithms

Key transport algorithms are used for encrypting symmetric encryption keys, such as data encryption keys, with asymmetric encryption keys. This technique allows for encryption of relatively large amount of data with efficient symmetric encryption and securely transmitting the associated relatively small symmetric encryption key. Industries, organizations, and application domains are currently choosing from a variety of key transport algorithms based on reasons including performance, security characteristics, and regulatory compliance. A set of the most commonly chosen and widely deployed key transport algorithms are supported by the Basic Security Profile in order to avoid disenfranchising existing applications. At some point in the future, if and when consensus is reached for a single key transport algorithm the Basic Security Profile 1.1 may be revised to constrain instances to use only that algorithm.

R5621 *When used for Key Transport, any* **EK_ENCRYPTION_METHOD** *Algorithm attribute MUST have a value of*

"http://www.w3.org/2001/04/xmlenc#rsa-1_5" or

"http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"

The RSA (PKCS#1.5) algorithm ("http://www.w3.org/2001/04/xmlenc#rsa-1_5") is widely implemented and deployed in existing practice. The RSA-OAEP algorithm ("http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p") is relatively new and becoming widely implemented and deployed.

10.12.2 Key Wrap Algorithms

Key wrap algorithms are used for encrypting symmetric encryption keys, such as data encryption keys, with symmetric encryption keys. Industries, organizations, and application domains are currently choosing from a variety of key wrap algorithms based on reasons including performance, security characteristics, and regulatory compliance. A set of the most commonly chosen and widely deployed key wrap algorithms are supported by the Basic Security Profile in order to avoid disenfranchising existing applications. At some point in the future, if and when consensus is reached for a single key wrap algorithm the Basic Security Profile 1.1 may be revised to constrain instances to use only that algorithm.

R5625 *When used for Key Wrap, any* **EK_ENCRYPTION_METHOD** *Algorithm attribute MUST have a value of*

"http://www.w3.org/2001/04/xmlenc#kw-tripledes", or

"http://www.w3.org/2001/04/xmlenc#kw-aes128", or

"http://www.w3.org/2001/04/xmlenc#kw-aes256".

The 3DES algorithm ("http://www.w3.org/2001/04/xmlenc#kw-tripledes") is widely implemented and deployed in existing practice. The AES algorithm is relatively new and becoming widely implemented and deployed. The 128-bit variation of AES ("http://www.w3.org/2001/04/xmlenc#kw-aes128") is relatively faster but weaker than the 256-bit variation ("http://www.w3.org/2001/04/xmlenc#kw-aes256").

10.12.3 Key Encryption Algorithms

R5626 *Any* **EK_ENCRYPTION_METHOD** *Algorithm attribute MUST have a value of*

"http://www.w3.org/2001/04/xmlenc#rsa-1_5" or

"http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" or

"http://www.w3.org/2001/04/xmlenc#kw-tripledes" or

"http://www.w3.org/2001/04/xmlenc#kw-aes128" or

"http://www.w3.org/2001/04/xmlenc#kw-aes256".

10.13 Encrypted Headers

10.13.1 Encrypted Headers

In order for the data to be protected from inspection, it must be replaced with the corresponding encrypted content.

R5614 *A* **HEADER** *encrypted as a result of an encryption step MUST be replaced by a corresponding ENCRYPTED_HEADER.* C

R5606 *Any encrypted element or element content within a* **SECURE_ENVELOPE***, encrypted as a result of an encryption step, MUST be replaced by a corresponding*

ENCRYPTED_DATA, unless the element is a HEADER_ELEMENT. C

11 Binary Security Tokens

11.1 Binary Security Tokens

11.1.1 BinarySecurityToken/@EncodingType Attribute Mandatory

Base64Binary is the only encoding type specified by Web Services Security: SOAP Message Security. Explicit specification of attribute values simplifies XML processing requirements and as a general principle the Basic Security Profile 1.1 requires that attributes be explicitly specified rather than relying on default values.

R3029 *Any* **BINARY_SECURITY_TOKEN** *MUST specify an EncodingType attribute.*

R3030 *Any* **BINARY_SECURITY_TOKEN** *EncodingType attribute MUST have a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary".*

A BinarySecurityToken may specify its encoding type. The Profile restricts the encoding type to Base64Binary and requires its explicit specification.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:BinarySecurityToken
element is missing an EncodingType attribute -->
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0##X509v3">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
```

CORRECT:

```
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
```

11.1.2 BinarySecurityToken/@ValueType Attribute Mandatory

There is no appropriate default for ValueType so the Basic Security Profile 1.1 mandates that an explicit value be provided.

R3031 *Any* **BINARY_SECURITY_TOKEN** *MUST specify an ValueType attribute.*

R3032 *Any* **BINARY_SECURITY_TOKEN** *ValueType attribute MUST have a value specified by the related security token profile.*

Web Services Security: SOAP Message Security allows for a BinarySecurityToken to optionally specify its ValueType. The Profile restricts the ValueType to one of those specified by a security token profile and requires its specification.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:BinarySecurityToken
element is missing a ValueType attribute -->
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
```

INCORRECT:

```
          <!-- This example is incorrect because the ValueType attribute on
the wsse:BinarySecurityToken element has an incorrect value. -->
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://www.mta.org/NYC#SubwayToken"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
```

CORRECT:

```
          <wsse:BinarySecurityToken wsu:Id='SomeCert'
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">

lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uxxVBovT7JV1A2RnWSWkXm9jAEdsm/.
..
          </wsse:BinarySecurityToken>
```

12 Username Token

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- Web Services Security: UsernameToken Profile 1.1 OASIS Standard Specification [WSS-User1.1]

12.1 Password

12.1.1 Not More Than One Password

R4222 *Any* **USERNAME_TOKEN** *MUST NOT have more than one PASSWORD.*

12.1.2 Password/@Type Attribute Mandatory

Passwords may be present in a variety of formats. The Type attribute specifies the format of the Password value.

R4201 *Any* **PASSWORD** *MUST specify a Type attribute.*

To avoid ambiguity, the Type attribute must be specified on the wsse:Password element of a wsse:UsernameToken.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the wsse:Password element is
missing a Type attribute with a value of
          http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
token-profile-1.0#PasswordText -->
          <wsse:UsernameToken
          xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd' >
          <wsse:Username>Bert</wsse:Username>
          <wsse:Password>Ernie</wsse:Password>
          </wsse:UsernameToken>
```

INCORRECT:

```
          <
          <!-- This example is incorrect because the wsse:Password element is
missing a Type attribute with a value of
          http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
token-profile-1.0#PasswordDigest -->
          <wsse:UsernameToken
          xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd' >
          <wsse:Username>Bert</wsse:Username>
          <wsse:Password>B5twk47KwSrjeg==</wsse:Password>
          </wsse:UsernameToken>
```

CORRECT:

```
          <wsse:UsernameToken
```

```
          xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd' >
          <wsse:Username>Bert</wsse:Username>
          <wsse:Password
          Type='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText'>
          Ernie
          </wsse:Password>
          </wsse:UsernameToken>
```

CORRECT:

```
          <wsse:UsernameToken xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' >
          <wsse:Username>Bert</wsse:Username>
          <wsse:Password
          Type='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest'>
          B5twk47KwSrjeg==
          </wsse:Password>
          </wsse:UsernameToken>
```

12.1.3 Digest Value

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines how it is used when calculating the Password_Digest . The following requirement applies when SHA-1 is used: The underlying specifications specify a Type value for password digests but there is ambiguity in the algorithm to be used to calculate the digest.

R4212 *If SHA-1 is used, a* **PASSWORD** *with a Type attribute value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest" MUST have its value computed using the following formula, where "+" indicates concatenation: Password_Digest = Base64 (SHA-1 (nonce + created + password)). That is, concatenate the text forms of the nonce, creation time, and the password (or shared secret or password equivalent), digest the combination using the SHA-1 hash algorithm, then include the Base64 encoding of that result as the password (digest). Any elements that are not present are simply omitted from the concatenation.*

The Profile describes the digest calculation details to eliminate ambiguity.

12.1.4 Key Derivation

R4216 *When a* **SECURITY_TOKEN**_*REFERENCE, within a SIGNATURE or ENCRYPTED_KEY, refers to a SECURITY_TOKEN named wsse:UsernameToken to derive a key, the key MUST be derived using the algorithm specified in Section 4 of Web Services Security: UsernameToken Profile 1.1.*

R4217 *When a* **SECURITY_TOKEN**_*REFERENCE, within a SIGNATURE or ENCRYPTED_KEY, refers to a SECURITY_TOKEN named wsse:UsernameToken to derive a key, the SECURITY_TOKEN MUST contain a wsse11:Salt child element.*

R4218 *When a* **SECURITY_TOKEN**_*REFERENCE, within a SIGNATURE or ENCRYPTED_KEY, refers to a SECURITY_TOKEN named wsse:UsernameToken to derive a key, the SECURITY_TOKEN MUST contain a wsse11:Iteration child element with a value greater than or equal to 1000.*

12.2 Created

12.2.1 Not More Than One Created

R4223 *Any* **USERNAME_TOKEN** *MUST NOT have more than one CREATED.*

12.3 Nonce

12.3.1 Not More Than One Nonce

R4225 *Any* **USERNAME_TOKEN** *MUST NOT have more than one NONCE.*

12.3.2 Nonce/@EncodingType Attribute Mandatory

Base64Binary is the only encoding type specified by Web Services Security: SOAP Message Security. Explicit specification of attribute values simplifies XML processing requirements and as a general principle the Basic Security Profile 1.1 requires that attributes be explicitly specified rather than relying on default values.

R4220 *Any* **NONCE** *MUST specify an EncodingType attribute.*

R4221 *Any* **NONCE** *EncodingType attribute MUST have a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary".*

A UsernameToken may specify its encoding type. The Profile restricts the encoding type to Base64Binary and requires its explicit specification.

12.4 SecurityTokenReference

12.4.1 UsernameToken Reference/@ValueType Attribute Value

The underlying specifications do not fully describe the proper ValueType for UsernameToken.

R4214 *Any* **STR_REFERENCE** *to a USERNAME_TOKEN MUST have a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken".*

The Profile requires a specific value for the ValueType.

12.4.2 UsernameToken KeyIdentifier Prohibited

The underlying specifications do not describe a mechanism for generating a KeyIdentifier for a UsernameToken.

R4215 *Any* **SECURITY_TOKEN_REFERENCE** *to a USERNAME_TOKEN MUST NOT contain an STR_KEY_IDENTIFIER.*

The Profile disallows the use of unspecified mechanisms for generation of KeyIdentifier values.

13 X.509 Certificate Token

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference, and defines extensibility points within them:

- Web Services Security: X.509 Certificate Token Profile 1.1 OASIS Standard Specification [WSS-X.509-1.1]
- RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile [RFC2459]
  Extensibility points:
  - E0012 - Certificate Authority - The choice of the Certificate Authority is a private agreement between parties.
  - E0013 - Certificate Extensions - X.509 allows for arbitrary certificate extensions.
- Public-key and attribute certificate frameworks Technical Corrigendum 1 [X.509-2000TC1]
- RFC4514 . Lightweight Directory Access Protocol : String Representation of Distinguished Names [RFC4514]

13.1 X.509 Token Types

In some areas the Basic Security Profile allows limited flexibility and extensibility in the application of security to messages. Some agreement between the SENDER and RECEIVER over which mechanisms and choices should be used for message exchanges is necessary. Since no security policy description language or negotiation mechanism is in scope for the Basic Security Profile, some out of band agreement must be reached for which certificate extensions and issuers should be used.

Web Services Security: X.509 Token Profile defines 3 token types: X509v3; x509PKIPathv1; and PKCS7. The Profile places the following constraints on their use:

13.1.1 X.509 Token Format

R3033 *Any* **X509_TOKEN** *MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3".*

13.1.2 Certificate Path Token Types

The underlying specifications allow certificate path information to be provided via either X509PKIPathv1 or PKCS7 formats.

R5201 *Any* **BINARY_SECURITY_TOKEN** *containing an X.509 Certificate Path MUST be either a PKCS7_TOKEN or a PKIPATH_TOKEN.*

R5202 *Any* **BINARY_SECURITY_TOKEN** *containing an X.509 Certificate Path SHOULD be a PKIPATH_TOKEN.*

Interoperability issues may arise if different forms of certificate path information are used when not expected. X509PKIPathv1 is preferred because it allows more efficient certificate path processing. PKCS7 is a more mature and widely implemented standard so it is also allowed. Section 3.1 of X.509 Token Profile incorrectly defines #X509PKIPathv1 as a PKIPath. Section 3.1 should define #X509PKIPathv1 as a PkiPath (note case), which X.509 defect report 279 defines as an ordered collection of certificates beginning with the most significant.

13.1.3 PKCS7 Token Format

R5211 *Any* **PKCS7_TOKEN** *MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7".*

13.2 SecurityTokenReference

13.2.1 SecurityTokenReference to X.509 Token

R5218 *Any* **STR_REFERENCE** *to a X509_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3".*

13.2.2 SecurityTokenReference to PKCS7 Token

R5212 *Any* **SECURITY_TOKEN_REFERENCE** *to a PKCS7_TOKEN MUST contain a wsse11:TokenType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7".*

R5213 *Any* **STR_REFERENCE** *to a PKCS7_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7".*

13.2.3 PkiPath Token Format

R5214 *Any PKIPATH_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1".*

13.2.4 SecurityTokenReference to PkiPath Token

R5215 *Any* **SECURITY_TOKEN_REFERENCE** *to a PKIPATH_TOKEN MUST contain a wsse11:TokenType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1".*

R5216 *Any* **STR_REFERENCE** *to a PKIPATH_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1".*

13.2.5 KeyIdentifier or X509IssuerSerial for External References

Web Services Security: SOAP Message Security provides a list of reference mechanisms in preferred order (i.e., most specific to least specific). Direct References are preferred, but when they cannot be used Key Identifier or Issuer and Serial Number is required.

R5209 *When a* **SECURITY_TOKEN_REFERENCE** *references an EXTERNAL_SECURITY_TOKEN that cannot be referred to*

using an STR_REFERENCE but can be referred to using an STR_KEY_IDENTIFIER or STR_ISSUER_SERIAL, an STR_KEY_IDENTIFIER or STR_ISSUER_SERIAL MUST be used.

In the event that Direct References are not possible, the Basic Security Profile 1.1 encourages the usage of mechanisms that are most likely to be unique.

For example,

CORRECT:

```
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509SubjectKeyIdentifier" >
          MIGfMa0GCSq
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

CORRECT:

```
          <wsse:SecurityTokenReference>
          <ds:X509Data>
          <ds:X509IssuerSerial>
          <ds:X509IssuerName>CN=Security WG, OU=BSP, O=WS-I,
C=US</ds:X509IssuerName>
          <ds:X509SerialNumber>54A4E9</ds:X509SerialNumber>
          </ds:X509IssuerSerial>
          </ds:X509Data>
          </wsse:SecurityTokenReference>
```

13.2.6 KeyIdentifier/@ValueType Attribute Value

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines how it is used in the URI for STR_KEY_IDENTIFIER.

R5206 *Any* **STR_KEY_IDENTIFIER** *that references an X509_TOKEN MUST have a ValueType attribute with the value of either*

"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier" or only if SHA-1 is used ,"http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1".

13.2.7 KeyIdentifier Value

The underlying specifications do not fully describe the proper ValueType for X.509 SubjectKeyIdentifier.

R5208 *Any* **STR_KEY_IDENTIFIER** *that references an X509_TOKEN and has a ValueType attribute with the value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier" MUST contain the value of the token's SubjectKeyIdentifier extension.*

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines how it is used in the URI for the ValueType attribute.

R5210 *If SHA-1 is used, STR_KEY_IDENTIFIER that references an X509_TOKEN which does not contain a SubjectKeyIdentifier extension MUST have a ValueType attribute with the value of "http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1" and MUST contain the value of the SHA1 of the raw octets of the X509_TOKEN that is referenced.*

The Profile requires a specific value for the ValueType.

13.2.8 X509IssuerSerial Value

R5409 *Any* **STR_ISSUER_SERIAL** *MUST contain a value following the encoding rules specified in the XML Signature specification for DNames.*

Per XML Signature, DNames are encoded as follows:

- To encode a distinguished name (X509IssuerSerial, X509SubjectName, and KeyName if appropriate), the encoding rules in section 2 of RFC 4514 SHOULD be applied, except that the character escaping rules in section 2.4 of RFC 4514 MAY be augmented as follows:
  - Escape all occurrences of ASCII control characters (Unicode range \x00 - \x1f) by replacing them with "\" followed by a two digit hex number showing its Unicode number.
  - Escape any trailing space characters (Unicode \x20) by replacing them with "\20", instead of using the escape sequence "\ ".
- Since a XML document logically consists of characters, not octets, the resulting Unicode string is finally encoded according to the character encoding used for producing the physical representation of the XML document.

14 REL Token

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- Web Services Security: Rights Expression Language (REL) Token Profile 1.1 OASIS Standard [WSS-Rel1.1]

14.1 SecurityTokenReferences

Web Services Security: Rights Expression Language (REL) Token Profile Section 3.4 defines several mechanisms for referencing REL tokens. This Profile places the following constraints on their use:

14.1.1 SecurityTokenReference to REL Token

R6304 *Any* **STR_REFERENCE** *to a REL_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf#license".*

14.1.2 Reference by licenseId Prohibited When wsu:Id Present

Direct references by shorthand XPointer are easier to resolve than references by licenseId. Using such references is consistent with constraints on other token types.

R6301 *Any* **STR_REFERENCE** *to a INTERNAL_SECURITY_TOKEN that is an REL_TOKEN containing a wsu:Id attribute, MUST NOT use a licenseId reference.*

For example,

CORRECT:

```
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <rel:license xmlns:rel='urn:mpeg:mpeg21:2003:01-REL-R-NS'
          wsu:Id='SomeLic'
          licenseId='uuid:3D680C71-177B-40cc-84C1-123B02503524' >
          . . .
          </rel:license>
          <ds:Signature>
          . . .
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='#SomeLic'
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-rel-token-
profile-1.0.pdf#license" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
```

INCORRECT:

```
          <!-- This example is incorrect because it refers to REL License
using the
          value of the licenseId attribute rather than the value of the
wsu:Id attribute -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <rel:license xmlns:rel='urn:mpeg:mpeg21:2003:01-REL-R-NS'
          wsu:Id='SomeLic'
          licenseId='uuid:3D680C71-177B-40cc-84C1-123B02503524' >
          . . .
          </rel:license>
          <ds:Signature>
          . . .
          <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:Reference
          URI='uuid:3D680C71-177B-40cc-84C1-123B02503524'
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-rel-token-
profile-1.0.pdf#license" />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </ds:Signature>
          </wsse:Security>
```

14.1.3 Issuer Signature on REL Token Precedes First Reference

This requirement ensures that the integrity and provenance of the license has been determined prior to use.

R6302 *Any* **SECURITY_HEADER** *child elements MUST be ordered so that any SIGNATURE necessary to verify the issuance of an REL_TOKEN precedes the first SECURITY_TOKEN_REFERENCE that refers to that REL_TOKEN.*

15 Kerberos Token

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- Web Services Security: Kerberos Token Profile 1.1 OASIS Standard Specification [WSS-Kerb1.1]

15.1 Content

Web Services Security: Kerberos Token Profile contains various statements containing the RFC2119 'MUST' keyword. This Profile restates some of those statements:

15.1.1 Kerberos Token Format

There is no appropriate default for ValueType in a wsse:BinarySecurityToken, so Basic Security Profile mandates that an explicit value be provided. Web Services Security: Kerberos Token Profile defines six token types. Although ValueType URIs are defined for each, the option can discourage interoperability.

R6902 *Any* **KERBEROS_TOKEN** *MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ"*

Web Services Security: SOAP Message Security allows for a BinarySecurityToken to optionally specify its ValueType. The Profile restricts the ValueType to GSS wrapped tickets to promote interoperability.

For example,

INCORRECT:

```
          <!-- This example is incorrect because it contains no ValueType
attribute in the SECURITY_TOKEN -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id="myKerberosToken"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          YIIEZzCCA9CgAwIBAgIQEmtJZc0...
          </wsse:BinarySecurityToken>
          ...
          </wsse:Security>
```

INCORRECT:

```
          <!-- This example is incorrect because it includes a ValueType
attribute indicating that it is a non-wrapped
          Kerberos v5 AP-REQ -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
```

```
          <wsse:BinarySecurityToken wsu:Id="myKerberosToken"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#Kerberosv5_AP_REQ"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          boIEZzCCA9CgAwIBAgIQEmtJZc0...
          </wsse:BinarySecurityToken>
          ...
          </wsse:Security>


CORRECT:
          <!-- This example is correct because it includes a ValueType
attribute with a value specified in the profile. In this case, it
          indicates that it is a GSS wrapped Kerberos v5 AP-REQ -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id="myKerberosToken"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#GSS_Kerberosv5_AP_REQ"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          YIIEZzCCA9CgAwIBAgIQEmtJZc0...
          </wsse:BinarySecurityToken>
          ...
          </wsse:Security>
```

15.1.2 Internal Token in First Message

R6903 *Any* **KERBEROS_TOKEN** *MUST be an INTERNAL_SECURITY_TOKEN in the initial SECURE_ENVELOPE of an authenticated message exchange between a SENDER and RECEIVER.*

15.1.3 External Token in Subsequent Messages

R6904 *Any* **KERBEROS_TOKEN** *MUST be an EXTERNAL_SECURITY_TOKEN in each SECURE_ENVELOPE after the initial SECURE_ENVELOPE of an authenticated message exchange between a SENDER and RECEIVER.*

15.2 SecurityTokenReference

15.2.1 SecurityTokenReference to Kerberos Token

R6907 *Any* **SECURITY_TOKEN_REFERENCE** *to a KERBEROS_TOKEN MUST contain a wsse11:TokenType*

attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ".

15.2.2 KeyIdentifier ValueType for Kerberos

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines how it is used in the URI for the ValueType attribute

R6906 *If SHA-1 is used, an* **STR_KEY_IDENTIFIER** *to a KERBEROS_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-kerberos-tokenprofile-1.1#Kerberosv5APREQSHA1".*

15.2.3 KeyIdentifier for External Token

Kerberos provides a security session mechanism where the first secure envelope in an authenticated message exchange contains the kerberos token, and subsequent secure envelopes can make reference to this token. Thus, the same kerberos token may be referred to as an internal or external token, depending on its order in an authenticated message exchange. This ambiguity could cause interoperability problems.

R6905 *Any* **SECURITY_TOKEN_REFERENCE** *to an EXTERNAL_SECURITY_TOKEN which is a KERBEROS_TOKEN MUST contain an STR_KEY_IDENTIFIER.*

This clarifies the referencing mechanism, depending on whether this is the first exchanged using a token, or is a subsequent message that makes reference to an earlier exchanged token.

For example,

INCORRECT:

```
          <!-- This example is incorrect because the external reference is
direct instead of using a wsse:KeyIdentifier.  -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >

          <wsse:SecurityTokenReference>
          <wsse:Reference URI="http://www.ws-
i.org/Kerberos/Examples/kerberosToken.b64"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#GSS_Kerberosv5_AP_REQ"/>
          </wsse:SecurityTokenReference>
          ...
          </wsse:Security>
```

CORRECT:

```
          <!-- This example is correct for the initial SECURE_ENVELOPE of an
authenticated message exchange. -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
```

```
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:BinarySecurityToken wsu:Id="myKerberosToken"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#GSS_Kerberosv5_AP_REQ"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          YIIEZzCCA9CgAwIBAgIQEmtJZc0...
          </wsse:BinarySecurityToken>
          <wsse:SecurityTokenReference>
          <wsse:Reference URI="#myKerberosToken"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#GSS_Kerberosv5_AP_REQ"/>
          </wsse:SecurityTokenReference>
          ...
          </wsse:Security>
```

CORRECT:

```
          <!-- This example is correct for any SECURE_ENVELOPE after the
initial SECURE_ENVELOPE of an authenticated message exchange.  -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-
tokenprofile-1.1#Kerberosv5APREQSHA1"/>
          EZzCCA9CgAwIB...
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
          ...
          </wsse:Security>
```

16 SAML Token

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- Web Services Security: SAML Token Profile 1.1 OASIS Standard [WSS-SAML1.1]

16.1 KeyInfo

16.1.1 References to SAML Tokens Prohibited

This requirement rules out the possibility of a SAML assertion referring to itself, an undesirable occurrence as it essentially makes the assertion self certifying. In addition a reference to another SAML assertion is also ruled out, this is undesirable as SAML does not have a transitive trust model.

R6601 *Any* **SAML_SC_KEY_INFO** *MUST NOT contain a reference to a SAML_TOKEN.* C

For example,

INCORRECT:

```
          <!-- This example is incorrect because the ds:KeyInfo in the SAML
assertion contains a reference to another such assertion thus conflicting
with R6601 -->
          <wsse:Security xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
          xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd'
          xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
          xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <saml:Assertion xmlns:saml='urn:oasis:names:tc:SAML:1.0:assertion'
          MajorVersion='1' MinorVersion='1'
          AssertionID='uuid:006ab385-35e0-41b1-b0f5-ccef5090c1b0'
          Issuer='http://example.org/issuer' IssueInstant='2004-11-
04T21:01:50Z' >
          . . .
          <saml:AuthenticationStatement
AuthenticationMethod='urn:oasis:names:tc:SAML:1.0:am:password'
AuthenticationInstant='2004-11-04T21:01:50Z' >
          <saml:Subject>
          . . .
          <saml:SubjectConfirmation>
          . . .
          <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
          <wsse:SecurityTokenReference>
          <wsse:Reference URI='uuid:a9afffbe-a0fb-4789-8b54-299782c3c0ac'
          ValueType='http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.0#SAMLAssertionID' />
          </wsse:SecurityTokenReference>
          </ds:KeyInfo>
          </saml:SubjectConfirmation>
          </saml:Subject>
          </saml:AuthenticationStatement>
          . . .
          </saml:Assertion>
```

```
          </wsse:Security>
```

16.2 SecurityTokenReference

16.2.1 SecurityTokenReference to SAML V1.1 Token

R6611 *Any* **SECURITY_TOKEN_REFERENCE** *to a SAML_V1_1_TOKEN MUST contain a wsse11:TokenType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1".*

16.2.2 SecurityTokenReference to SAML V2.0 Token

R6617 *Any* **SECURITY_TOKEN_REFERENCE** *to a SAML_V2_0_TOKEN MUST contain a wsse11:TokenType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0".*

R6614 *Any* **SECURITY_TOKEN_REFERENCE** *that references an INTERNAL_SAML_V2_0_TOKEN using a STR_REFERENCE MUST NOT contain a ValueType attribute.*

For example,

CORRECT:

```
          <wsse:SecurityTokenReference
          xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd'
          xmlns:wsse11='http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
secext-1.1.xsd'
          wsse11:TokenType='http://docs.oasis-open.org/wss/oasis-wss-saml-
tokenprofile-1.1#SAMLV2.0'>
          <wsse:KeyIdentifier wsu:Id='Token01'
          ValueType='http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLID'>
          _a75adf55-01d7-40cc-929f-dbd8372ebdfc
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

16.2.3 KeyIdentifier/@ValueType Attribute

R6602 *Any* **STR_KEY_IDENTIFIER** *that references a INTERNAL_SAML_TOKEN MUST include a ValueType attribute.* C

R6609 *Any* **STR_KEY_IDENTIFIER** *that references a EXTERNAL_SAML_TOKEN MUST include a ValueType attribute.* C

R6603 *Any* **STR_KEY_IDENTIFIER** *ValueType attribute that references a SAML_V1_1_TOKEN MUST have a value of*

"http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID" C

R6616 *Any* **STR_KEY_IDENTIFIER** *ValueType attribute that references a SAML_V2_0_TOKEN MUST have a value of "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID"* C

16.2.4 KeyIdentifier/@EncodingType Attribute

These requirements restate various statements from the base specification related to references to SAML assertions that use wsse:KeyIdentifiers.

R6604 *Any* **STR_KEY_IDENTIFIER** *that references a SAML_TOKEN MUST NOT include an EncodingType attribute.* C

R6605 *Any* **STR_KEY_IDENTIFIER** *that references a SAML_TOKEN MUST have a value encoded as an xs:string.* C

For example,

CORRECT:

```
          <wsse:SecurityTokenReference xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'>
          <wsse:KeyIdentifier ValueType='http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID' >
          uuid:006ab385-35e0-41b1-b0f5-ccef5090c1b0
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

INCORRECT:

```
          <!-- This example is incorrect because the wsse:KeyIdentifier
element is missing a ValueType attribute thus conflicting with R6602 -->
          <wsse:SecurityTokenReference xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'>
          <wsse:KeyIdentifier>uuid:006ab385-35e0-41b1-b0f5-
ccef5090c1b0</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

INCORRECT:

```
          <!-- This example is incorrect because the wsse:KeyIdentifier
element has an incorrect value for the ValueType attribute thus conflicting
with R6603 -->
          <wsse:SecurityTokenReference xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'>
          <wsse:KeyIdentifier ValueType='http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.0#SAML'>
          uuid:006ab385-35e0-41b1-b0f5-ccef5090c1b0
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

INCORRECT:

```
          <!-- This example is incorrect because the wsse:KeyIdentifier has
an EncodingType attribute thus conflicting with R6604 -->
```

```
          <wsse:SecurityTokenReference xmlns:wsse='http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'>
          <wsse:KeyIdentifier ValueType='http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID'
          EncodingType='xs:string' >
          uuid:006ab385-35e0-41b1-b0f5-ccef5090c1b0
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
```

16.2.5 References to Internal SAML Assertions

Note that the Web Services Security: SAML Token Profile explicitly allows use of a key identifier reference when a direct reference is possible. This WS-I SAML Token Profile requires use of a direct or embedded reference to increase interoperability. TokenType and ValueType should be consistent and disallowing use of ValueType removes the potential for error.

R6610 *Any* **SECURITY_TOKEN_REFERENCE** *that references an INTERNAL_SAML_TOKEN that has an ID attribute, the reference MUST contain an STR_REFERENCE or an STR_EMBEDDED.* C

R6612 *Any* **SIG_REFERENCE** *to a SECURITY_TOKEN_REFERENCE which contains an STR_EMBEDDED which contains an INTERNAL_SAML_V2_0_TOKEN MUST NOT include a SIG_TRANSFORM with an Algorithm attribute value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform".* C

16.2.6 References to External SAML Assertions

These requirements restate various statements from the base specification related to references to SAML assertions that are outside a SECURE_ENVELOPE.

R6606 *Any* **SECURITY_TOKEN_REFERENCE** *that references an EXTERNAL_SAML_TOKEN MUST contain a SAML_AUTHORITY_BINDING.* C

R6607 *Any AuthorityKind attribute of a* **SAML_AUTHORITY_BINDING** *MUST have a value of saml:AssertionIDReference.* C

R6608 *Any* **SECURITY_TOKEN_REFERENCE** *that references an INTERNAL_SAML_TOKEN MUST NOT contain a SAML_AUTHORITY_BINDING.* C

R6613 *Any* **SECURITY_TOKEN_REFERENCE** *to an EXTERNAL_SAML_V2_0_TOKEN MUST contain an STR_REFERENCE.* C

Both SAML Token Profile 1.0 and 1.1 specified an incorrect QName to reference external SAML 1.1 assertions for the AssertionIDReference. This element is defined in Assertions and Protocols for SAML (SAML assertion schema). SAML Protocol uses the AssertionIDReference. The namespace and the element name were incorrectly specified - samlp:AssertionIdReference rather than saml:AssertionIDReference in the SAML Token Profiles. This profile references the correct QName. The SAML 1.1 assertion schema is the normative reference to this element referenced in this profile.

To encourage backward compatibility and acknowledge the errors, this profile suggests interoperable implementations accept either case - AssertionIdReference or AssertionIDreference.

17 EncryptedKey Token

17.1 SecurityTokenReference

17.1.1 SecurityTokenReference to EncryptedKey Token

R3069 *Any* **SECURITY_TOKEN_REFERENCE** *to a ENCRYPTED_KEY_TOKEN MUST contain a wsse11:TokenType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey".*

The choice of signature digest algorithm is governed by the extensibility point E0014. For interoperability reason, this profile determines how it is used in the URI for the ValueType attribute:

R3072 *If SHA-1 is used, an STR_KEY_IDENTIFIER element in a* **SECURITY_TOKEN_REFERENCE** *that refers to an ENCRYPTED_KEY_TOKEN MUST contain a ValueType attribute with a value of "http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKeySHA1".*

18 Attachment Security

This section of the Basic Security Profile 1.1 incorporates the following specifications by reference:

- Attachments Profile Version 1.0 [AP1.0]
- Web Services Security: SOAP Messages with Attachments (SwA) Profile 1.1 [WSS-SWA1.1]

The section provides guidance for protecting attachments when they are used with SOAP Messages. As is explained in Section 3 Conformance all features described in the Basic Security Profile 1.0, including support for attachments and security for attachments in any form by any instance is not required.

SSL/TLS may be used to provide authentication, integrity and confidentiality protection, on a hop-by-hop basis, for an entire HTTP Message. This includes HTTP Headers, the SOAP Envelope, and all MIME_PARTs.

SSL/TLS does not provide protection, except between adjacent HTTP Nodes, for HTTP Messages when the SOAP Message Path contains SOAP Intermediaries. An instance should not use SSL/TLS without Web Services Security with Message Exchange Patterns (MEPs) that may contain SOAP intermediaries or when these security functions are required to be performed independently of the connection.

Web Services Security may be used to provide authentication, integrity and confidentiality protection for a subset of the SOAP Message and associated attachments. Web Services Security provides protection for SOAP Messages and attachments when the SOAP Message Path contains SOAP Intermediaries. An instance should use Web Services Security with MEPs that may contain SOAP Intermediaries or when these security functions are required to be performed independently of the transport layer connection.

An instance may use SSL/TLS in conjunction with Web Services Security if warranted by application security requirements. This combination provides integrity and confidentiality protection for the entire HTTP Message (on a hop-by-hop basis) including HTTP Headers, SOAP Envelope, and all MIME_PARTs.

Application level security mechanisms, including XML Signature, XML Encryption, PKCS#7, S/MIME, etc. for attachment data may also be used by a instance where appropriate, but statements regarding the interoperability of such mechanisms are out of scope for the Basic Security Profile 1.0.

The Basic Security Profile 1.0 describes one attachment security mechanism and URI.

18.1 SOAP with Attachments

18.1.1 Conformance

The Basic Security Profile is an extension profile to the Basic Profile, and thus this requirement makes it explicit that when using the Web Services Security SwA profile in conformance to the Basic Security Profile 1.1 SwA profile section that the messages containing attachments must conform to the WS-I Attachments Profile 1.0. This is consistent with the Web Services Security SwA profile and good practice.

R6001 *Any* **SECURE_MESSAGE** *MUST conform to WS-I Attachments Profile 1.0.*

18.1.2 Relationship between Parts

The Web Services Security SwA profile outlines how attachments may be secured when conveyed in conjunction with a primary SOAP envelope. The SOAP envelope is contained in a distinct MIME part. Attachment parts can contain arbitrary content as indicated by the MIME-Type, and an attachment could contain content with nested MIME parts. In order to enable interoperable processing at the SOAP messaging layer it is important to only secure the top-level MIME attachments that are in the same Multipart structure as the MIME part conveying the primary SOAP envelope.

R6002 *A signed and/or encrypted* **MIME_PART** *MUST be at the same MIME level as the root MIME_PART containing the SECURE_ENVELOPE.*

18.1.3 Encryption and Root Part

It is essential that SOAP processors (intermediaries and ultimate SOAP receiver) be able to process the primary SOAP envelope. For this reason, the MIME part containing the primary SOAP envelope must not be secured using the Web Services Security SwA profile mechanisms. This includes encryption, since an encrypted SOAP structure cannot be processed, and signing, since this would preclude intermediary processing of the SOAP message, since this often involves adding, or removing headers. Note that Web Services Security may be applied to portions of the primary SOAP envelope in conformance with the Basic Security Profile, but not by using attachment security mechanisms.

R6003 *A root* **MIME_PART** *MUST NOT be referenced by a SIG_REFERENCE, ENC_REFERENCE_LIST, or EK_REFERENCE_LIST.*

18.2 Signed Attachments

18.2.1 Reference to Signed Attachments

R6100 *A* **SIG_REFERENCE** *to a MIME_PART MUST use a URI attribute of the form "cid:partToBeSigned".*

18.2.2 Attachment Transforms

The Web Services Security SwA profile requires that a ds:Reference to an attachment part that is signed must use a transform specifying either the Attachment-Complete-Signature-Transform or the Attachment-Content-Signature-Transform URIs. This statement reiterates that requirement for testability.

R6101 *A* **SIG_REFERENCE** *to a MIME_PART MUST use a TRANSFORM specified by the Algorithm*

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Signature-Transform" or

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete-Signature-Transform".

18.2.3 Canonicalization

Signature digest calculations require that the exact same input be provided to the digest algorithm both when the signature is created and when it is verified. Canonicalization is required to ensure the same literal representation despite changes due to message transformation during transport. XML exclusive canonicalization is required as part of the Attachment-Complete-Signature-Transform or Attachment-Content-Signature-Transform processing. A separate transform for canonicalization should not be provided since it is already included as part of the processing associated with these transforms.

R6103 *At any* **RECEIVER** *the output of the Attachment-Content-Signature-Transform or the XML octet stream portion of the output of the Attachment-Complete-Signature-Transform must be consistent with Exclusive XML Canonicalization without comments having been performed when creating that output.*

18.2.4 Digest Values

Signature digest calculations require that the exact same input be provided to the digest algorithm both when the signature is created and when it is verified. Canonicalization is required to ensure the same literal representation despite changes due to message transformation during transport, for example line ending changes. MIME canonicalization algorithms are required for interoperability by Web Services Security SwA profile. This statement reiterates that requirement for testability.

R6104 *Any* **SIG_REFERENCE** *to a MIME_PART not containing XML MUST include a ds:DigestValue calculated using MIME canonicalization according to the requirements of the MIME Type.*

R6108 *Any* **SIG_REFERENCE** *to a MIME_PART containing the "application/text" content type MUST include a ds:DigestValue which is calculated after the text is canonicalized according to MIME canonicalization algorithm for text.*

R6109 *Any* **SIG_REFERENCE** *to a MIME_PART containing an XML content type MUST include a ds:DigestValue which is calculated after the XML is canonicalized according to Exclusive XML Canonicalization without comments, as specified by the URI "http://www.w3.org/2001/10/xml-exc-c14n#".*

18.2.5 Content-Type

Signature digest calculations require that the exact same input be provided to the digest algorithm both when the signature is created and when it is verified. Canonicalization is required to ensure the same literal representation despite changes due to message transformation during transport. It is essential to determine the correct type of canonicalization to perform, based on the MIME_PART content type. For this reason it is required that the Content-Type be explicitly stated with a Content-Type MIME header.

R6106 *A* **MIME_PART** *referenced by a SIG_REFERENCE which specifies a Transform algorithm of "http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete-Signature-Transform" MUST have a Content-Type MIME-header.*

R6107 *A* **MIME_PART** *referenced by a SIG_REFERENCE which specifies a Transform algorithm of "http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Signature-Transform" MUST have a Content-Type MIME-header.*

18.3 Encrypted Attachments

18.3.1 References to Encrypted Attachments

The Web Services Security SwA profile requires that an EncryptedData element in the primary SOAP envelope be used to reference the encrypted MIME part. This statement reiterates that requirement for testability.

R6200 *An encrypted* **MIME_PART** *MUST be referenced using an ENCRYPTED_DATA.*

18.3.2 Type attribute

The Web Services Security SwA profile requires that an EncryptedData element in the primary SOAP envelope that references an encrypted MIME part specify the Type as either Attachment-Content-Only or Attachment-Complete so that it can be processed correctly upon decryption. This statement reiterates that requirement for testability.

R6201 *An* **ENCRYPTED_DATA** *that references a MIME_PART MUST include a Type attribute with the value*

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Only" or

"http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete".

18.3.3 Reference URIs

The Web Services Security SwA profile requires that the EncryptedData element in the primary SOAP envelope that references an encrypted MIME part must reference the Cipherdata using the same cid: as the original attachment cid:, simplifying processing. This statement reiterates that requirement for testability.

R6202 *An* **ENCRYPTED_DATA** *that references a MIME_PART MUST contain a xenc:CipherData/xenc:CipherReference element with a URI attribute having the same URI as the original MIME_PART.*

18.3.4 Content

The Web Services Security SwA profile specifies that when an attachment is encrypted, the resulting EncryptedData element be placed in the wsse:Security header in the primary SOAP envelope and that the cipherdata be placed in the attachment part. This statement reiterates the requirement of replacing the content of the attachment with the cipherdata, for testability.

R6203 *The content of a* **MIME_PART** *encrypted according to the Web Services Security SwA profile MUST be replaced by the cipher value that results from encrypting the MIME_PART and is referenced from the CipherReference in the EncryptedData element.*

19 Security Considerations

This section lists a number of security considerations that should be taken into account when using one or more of the technologies discussed in the Basic Security Profile 1.1.

19.1 SOAPAction Header

The use of the SOAPAction header in situations where the message content is being integrity or confidentiality protected can result in security risks when the transport layer does not provide the same protection to the SOAPAction header. The most obvious risk is that the SOAPAction header can potentially expose sensitive information about a SOAP message such as the URI of the service, or the context of the transaction that is taking place. Another, more subtle risk occurs in a situation where message routing is done based on the value of the SOAPAction header. By modifying the value, an attacker could cause the message to be directed to a different receiver. This could potentially defeat a replay detection mechanism that was based on the assumption that the message would always be routed to the same place. Yet another risk occurs when intermediates are present. An intermediate might decide on a set of processing steps based on the value of SOAPAction or application/soap+xml, which is subject to tampering. A subsequent receiver might base its processing on the actual message content, which could be secured through XML signatures. Suppose that the intermediate was a security gateway. It could be tricked into allowing a payment operation through that had only query security.

19.1.1 SOAPAction header

C2010 *In a* **DESCRIPTION***, the soapAction attibute of a soapbind:operation element SHOULD be either omitted, or have as its value an empty string.*

19.2 Clock Synchronization

The specifications covered by the Basic Security Profile 1.1 use time-based mechanisms to prevent replay attacks. These mechanisms will be ineffective unless the system clocks of the various network nodes are synchronized. Since the technology to perform distributed clock synchronization are well known and widely available and are not among the technologies being profiled here, this document does not specify how clock synchronization should be done. However, the recommendation of the use of time-based security mechanisms implies that synchronization is being done.

19.3 Security Token Substitution

19.3.1 Security Token Substitution

If a ds:SignedInfo contains one or more ds:Reference children whose URI attribute contains a shorthand XPointer reference to a wsse:SecurityTokenReference that uses a potentially ambiguous mechanism to refer to the Security Token (e.g. KeyIdentifier), then in order to protect against post-signature substitution of the Security Token with one that binds the same key to different claims, it is recommended that the content of the Security Token be signed either directly or using the Security Token Dereferencing Transform.

C5443 *When the signer's SECURITY_TOKEN is an INTERNAL_SECURITY_TOKEN, the* **SIGNED_INFO** *MAY include a SIG_REFERENCE that refers to the signer's SECURITY_TOKEN in order to prevent substitution with another SECURITY_TOKEN that uses the same key.* C

C5441 *When the signer's SECURITY_TOKEN is an EXTERNAL_SECURITY_TOKEN, the* **SIGNED_INFO** *MAY include a SIG_REFERENCE that refers to the SECURITY_TOKEN_REFERENCE that refers to the signer's SECURITY_TOKEN using the Security Token Dereferencing Transform in order to prevent substitution of another SECURITY_TOKEN that uses the same key.* C

If a message is signed using a Security Token that binds a public verification key with other claims, and specific processing is performed based on those claims, then in order to protect against post-signature substitution of the Security Token with one that binds the same key to different claims, Security Token itself should be part of the signature computation. This can be achieved by putting a child ds:Reference element whose URI attribute contains a shorthand XPointer reference to the wsse:SecurityTokenReference that specifies the Security Token into the ds:SignedInfo element of a signature.

19.3.2 Security Token Reference in Subsequent Messages

When a key is provided in band within a Security Token or otherwise for the purpose of specifying a key to be used by another node for encrypting information to be sent in a future message, it is recommended that the sender of the key cryptographically bind the key to the message in which it is transmitted. This can be done either by using the key to perform a Signature or HMAC over critical elements of the message body or by including the key under a signature covering critical elements of the message body which uses some other key.

C5442 *When an encryption key to be used in subsequent messages is provided in an INTERNAL_SECURITY_TOKEN, a* **SIGNED_INFO** *MAY include a SIG_REFERENCE that refers to the signer's SECURITY_TOKEN in order to prevent substitution with another SECURITY_TOKEN that uses the same key.* C

If a key is sent in a message which the receiver is expected to use to encrypt data in some future message, there is a risk that an attacker could substitute some other key and thereby be able to read unauthorized data. This is true even if the key is contained in a signed certificate, but is not bound to the current message in some way. If the future encryption key is used to sign the initial request, by verifying the signature, the receiver can determine that the key is the one that was intended.

19.4 Protecting against removal and modification of XML Elements

XML Signatures using Shorthand XPointer References (AKA IDREF) protect against the removal and modification of XML elements; but do not protect the location of the element within the XML Document.

Whether or not this is security vulnerability depends on whether the location of the signed data within its surrounding context has any semantic import. This consideration applies to data carried in the SOAP Body or the Header.

Of particular concern is the ability to relocate signed data into a SOAP Header block which is unknown to the receiver and marked mustUnderstand="false". This could have the effect of causing the receiver to ignore signed data which the sender expected would either be processed or result in the generation of a mustUnderstand fault.

A similar exploit would involve relocating signed data into a SOAP Header block targeted to a S11:actor or S12:role other than that which the sender intended, and which the receiver will not process.

While these attacks could apply to any portion of the message, their effects are most pernicious with SOAP header elements which may not always be present, but must be processed whenever they appear.

In the general case of XML Documents and Signatures, this issue may be resolved by signing the entire XML Document and/or strict XML Schema specification and enforcement. However, because elements of the SOAP message, particularly header elements, may be legitimately modified by SOAP intermediaries, this approach is usually not appropriate. It is RECOMMENDED that applications signing any part of the SOAP body sign the entire body.

Alternatives countermeasures include (but are not limited to):

- References using XPath transforms with Absolute Path expressions,
- A Reference using an XPath transform to include any significant location-dependent elements and exclude any elements that might legitimately be removed, added, or altered by intermediaries,
- Using only References to elements with location-independent semantics,
- Strict policy specification and enforcement regarding which message parts are to be signed. For example:
    - Requiring that the entire SOAP Body and all children of SOAP Header be signed,
    - Requiring that SOAP header elements which are marked mustUnderstand="false" and have signed descendents MUST include the mustUnderstand attribute under the signature.

19.5 Only What is Signed is Protected

A receiver should treat the origin of any unsigned content as suspect. Only the subset of message content that is signed by an entity can be trusted as having been produced or otherwise acknowledged by that entity. All other content should be treated as if it were placed in the message by an untrusted third party (potentially an adversary).

XML Signatures allow fine grained selection of elements to be signed through a variety of means including Shorthand XPointer references to selected elements and references using transforms (such as XPath) that cause a value other than what would be obvious by examining the serialized content of the element (by adding, modifying, or removing a value) to be used for digest computation. This could allow values to be added, modified, or removed without detection. Therefore, a receiver should not consider it sufficient to verify that an element is referenced, but should also verify that the result of the application of the indicated transform algorithms produce the expected, trusted value.

19.6 Use of SHA

Basic Security Profile 1.1 recommends using SHA based algorithms for interoperability purposes. If SHA1 based algorithms are deemed unsuitable, other algorithms (like SHA256) can be used.

19.7 Uniqueness of ID attributes

XML 1.0 requires that all attributes of type ID in a given XML document have unique values, but only validating XML processors have such type information. As various aspects of SOAP Message Security use ID based references it is recommended that applications ensure that ID attributes are unique by some mechanism.

19.8 Signing Security Tokens

In general, tokens contain claims made by an authority, usually about some system entity. Obviously a party relying on these claims must trust that authority to make them. The relying party must generally

verify these claims. The method of doing this depends on the token type and is specified by the corresponding token profile.

19.9 Signing Username Tokens

When a wsse:UsernameToken contains only a wsse:Username and wsse:PasswordText and is simply presented for Authentication where replay is not a concern, it does not need to be signed because the act of checking it against a stored value has the effect of verifying it. When a wsse:Nonce and/or wsu:Created are used with the wsse:Username and wsse:PasswordText to prevent replay, the wsse:UsernameToken must be signed to prevent undetected alteration of these fields. If a wsse:PasswordText is being used to derive a key for a subsequent encryption of a response, it should be signed to ensure that an attacker does not substitute an alternative, but valid wsse:Username and wsse:PasswordText. This is equivalent to the key substitution attack available when an X.509 Token is used for a similar purpose.

19.10 Signing Binary Tokens

The content of a binary token will be a binary object which is integrity protected by a mechanism specific to the object type. For example, an X.509 certificate will be signed by the issuing authority. The outer wrapper of the binary token merely contains type indication information which does not have to be integrity protected in order to be able to rely in the claims.

19.11 Signing XML Tokens

XML tokens should be digitally signed in a manner described by their profile (or documents referenced by it), or delivered directly from their issuer over an integrity protected channel.

19.12 Replay of Username Token

A sender that includes a Nonce child in a UsernameToken element should anticipate that the receiver may refuse to process the message due to either an accidental collision or transport layer delays. Therefore, if it decides to retry transmission, it should do so with a new Nonce.

Unless other mechanisms are used to protect against replay of the username token, service providers should retain nonces in a store that is shared between all SOAP nodes (and within a distributed SOAP node all "components") that can be authorized using the same passwords.

The policy that allows service providers to forget nonces may be based on any considerations that the service considers relevant. When a nonce is forgotten the server should ensure that in the future it rejects UsernameTokens with a Created time that is earlier than the forgotten nonce.

19.12.1 Replay of Username Token

C4210 *Any* **USERNAME_TOKEN** *that contains a NONCE and PASSWORD with a Type attribute value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText" SHOULD be referenced by a SIG_REFERENCE in order to prevent replay.*

C4211 *Any* **USERNAME_TOKEN** *that contains a CREATED and a PASSWORD with a Type attribute value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText" SHOULD be referenced by a SIG_REFERENCE in order to prevent replay.*

19.13 Use of Digest vs. Cleartext Password

A sending application utilizing password authentication must decide whether to use a cleartext password or a password digest (The sender needs to know via some out-of-band mechanism and/or prior arrangement which mechanisms the receiver supports). The digest should always be preferred if it can be used, as the digest algorithm does not reveal the password and can protect against replay of the password. (It does not however, protect against offline guessing or brute force attacks.)

Password digests can only be used in situations where both sender and receiver can start with the same secret value (e.g., the cleartext password or a hash of the password). The following are criteria for considering when to use digests vs. cleartext:

1. If the receiver can access the cleartext password, a digest may be used.

2. If the receiver can access a value that can be derived by the sender directly from the cleartext password (e.g., the receiver has access to a SHA1 hash of the password), the derived value (e.g. the hash) may be used in the digest.

3. If the sender needs extra information to derive the value available to the receiver, it will not be feasible to use password digest, even though the information is not intentionally secret. For example, UNIX systems add a salt value to each password before hashing it. It is infeasible for the sender to discover the salt value required for a specific username.

4. If the receiver does not have access to any password value, derived or otherwise, but merely the ability to test a username/password combination for validity, a digest may not be used. An example of this is when the username/password combination is presented to a database, directory or mainframe system for verification.

When sending any form of a password, cleartext or digest, confidentiality services are strongly recommended to prevent its value from being revealed or from offline guessing.

19.14 Encryption with Signatures

When a message contains a data value which does not have a significant number of probable variations and that data is signed and then encrypted, it is recommended that the sender either include some suitably random value such as a wsse:Nonce in the data, or encrypt the related ds:DigestValue element in order to protect the confidentiality of the data.

An adversary can compute the digest for each of the data values and compare them against the digests in the signature thereby deducing the encrypted data value. This type of attack is most likely to be successful when there are a relatively small set of probable data values. Therefore the threat can be mitigated by introduction of some random value into the original data or encryption of the digest.

19.14.1 Encrypt DigestValue

C5630 *Any* **SIGNATURE** *computed over data that is subsequently encrypted SHOULD also be encrypted in order to prevent plaintext guessing attacks when the probable set of data values is small.*

19.15 Possible Operational Errors

Under SOAP processing rules, there is no way a sender can be sure that a message containing a security header addressed to a given Role/Actor will ever reach a node that is taking on that Role/Actor. If not, the specified security processing will not occur.

Under SOAP processing rules, there is no way a sender can determine in what order nodes taking on specific Role/Actor's will be reached. If signatures and encryptions specified in different security headers overlap, verification and decryption operations may fail as a result of being processed in the wrong order. (Generally overlapping signatures will verify regardless of the order of verification.) This problem can be avoided by never specifying overlapping operations in distinct headers, however application requirements may not prevent this. For example, many senders may wish to include the entire Body under a signature, possibly before or after encrypting portions of it.

Under SOAP processing rules, there is no way a sender can determine which particular secrets are possessed by a node taking on a given Role/Actor. If a node is required to perform decryption or verify an HMAC and it does not posses the necessary secret, it will be unable to perform these operations. This will not only impact its operation, but in the case it is an intermediary may make it possible for nodes receiving the message subsequently from performing security processing correctly due to overlapping operations, even when that node does possess the necessary secrets.

If a namespace that is in fact visibly used within some text to be Canonicalized via the Exclusive C14N Algorithm is included in the PrefixList, then under some valid transformations of the transmitted document signature verification may spuriously fail, because the Canonicalized form shifts the location of a namespace declaration. This case is expected to be rare in practice.

Appendix A. Extensibility Points

This section identifies extensibility points, as defined in "Scope of the Basic Security Profile 1.1," for the Basic Security Profile 1.1's component specifications.

These mechanisms are out of the scope of the Basic Security Profile 1.1; their use may affect interoperability, and may require private agreement between the parties to a Web service.

In RFC 2246: The TLS Protocol Version 1.0 [RFC2246]:

- E0009 - **TLS Ciphersuites** - TLS allows for the use of arbitrary encryption algorithms. Note that while section 4.2 of the Basic Security Profile 1.1 mandates, recommends, and discourages support for certain ciphersuites, the Basic Security Profile 1.1 does not prohibit use of any specific ciphersuite.
- E0010 - **TLS Extensions** - TLS allows for extensions during the handshake phase.

In The SSL Protocol Version 3.0 [SSLV3]:

- E0011 - **SSL Ciphersuites** - SSL allows for the use of arbitrary encryption algorithms. Note that while section 4.2 of the Basic Security Profile 1.1 mandates, recommends, and discourages support for certain ciphersuites, the Basic Security Profile 1.1 does not prohibit use of any specific ciphersuite.

In Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification [WSS-SOAP]

- E0002 - **Security Tokens** - Security tokens may be specified in additional security token profiles.

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

- E0012 - **Certificate Authority** - The choice of the Certificate Authority is a private agreement between parties.
- E0013 - **Certificate Extensions** - X.509 allows for arbitrary certificate extensions.

About signature digest algorithms:

- E0014 – Signature Digest algorithm – version of SHA algorithm in use (SHA-1, or one of the SHA-2 algorithms).

Appendix B. Acknowledgments

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

Participants:

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

In addition, the Technical Committee thanks members of the WS-I Basic Security Profiles Working Group whose work provided the foundation for this document:

Jan Alexander (Microsoft Corporation), Steve Anderson (BMC), Paula Austel (IBM), Siddharth Bajaj (Verisign), Frank Balluffi (Deutsche Bank), Abbie Barbir (Nortel), David Baum (Kantega AS), Randy Bias (Grand Central Communications), Tim Bond (webMethods, Inc.), Heidi Buelow (Quovadx), David Burdett (Commerce One, Inc.), Ted Burghart (Hitachi, Ltd.), Symon Chang (TIBCO, Inc.), Richard Chennault (Kaiser Permanente), Dipak Chopra (SAP AG), Jamie Clark (OASIS), Edward Cobb (BEA Systems, Inc.), David Cohen (Merrill Lynch), Brett Cooper (Accenture), Ugo Corda (SeeBeyond Technology), Paul Cotton (Microsoft Corporation), Suresh Damodaran, (Rosettanet), Mark Davis (Intel), Alex Deacon (Verisign), Thomas DeMartini (ContentGuard, Inc.), Blake Dournaee (Intel), Rob Drew (Charles Schwab), Gregory Elkins (Reed Elsevier), Mark Ericson (Mindreef), Jon Oyvind Eriksen (Kantega AS), Chris Ferris (IBM), Bob Freund (Hitachi), Edwin Goei (Sun Microsystems), Grant Goodale (Reactivity, Inc.), Marc Goodner (SAP AG), Phil Goodwin (Sun Microsystems), Marc Graveline (Cognos, Inc.), Eric Gravengaard (Reactivity, Inc.), Thomas Gross (IBM), Martin Gudgin (Microsoft Corporation), Marc Hadley (Sun Microsystems), Mark Hapner (Sun Microsystems), Nathan Harris (Kaiser Permanente), Bret Hartman (IBM), Frederick Hirsch (Nokia), Jason Hogg (Microsoft Corporation), Maryann Hondo (IBM), Lawrence Hsiung (Quovadx), Tony Huber (Commerce Quest), Jim Hughes (Hewlett-Packard), Michael Hui (Computer Associates), Brian Jackson (Avanade, Inc.), Steve Jenisch (SAS Institute), Erik Johnson (Epicor), Chris Kaler (Microsoft Corporation), Anish Karmarkar (Oracle Corporation), Dana Kaufman, (Forum Systems), Manveen Kaur (Sun Microsystems), Slava Kavsan (RSA Security), Paul Knight (Nortel Networks), Chris Kurt (Microsoft Corporation), Kelvin Lawrence (IBM), Hal Lockhart (BEA Systems), Brad Lund (Intel Corporation), Jim Luth (OPC Foundation), Paul Madsen (Entrust, Inc.), Eve Maler (Sun Microsystems), Skip Marler (Parasoft), Monica Martin (Microsoft), Axl Mattheus (Sun Microsystems), Michael McIntosh (IBM), Craig Milhiser (Ascential), Chris Miller (Accenture), Prateek Mishra (Oracle Corporation), Dale Moberg (Cyclone Commerce), Ron Monzillo (Sun Microsystems), K. Scott Morrison (Layer 7), Tim Moses (Entrust, Inc.), Tony Nadalin (IBM), Nataraj Nagaratnam (IBM), Andrew Nash (RSA Security), Hsin Ning (Bestning Technologies), Eisaku Nishiyama (Hitachi, Ltd.), Mark Nottingham (BEA Systems, Inc.), TJ Pannu (ContentGuard, Inc.), Martine Pean (Quovadx), Robert Philpott (RSA Security), Dave Prout (BT), Joe Pruitt (F5 Networks, Inc.), Eric Rejkovic (Oracle Corporation), Matt Recupito (Accenture), Jason Rouault (Hewlett-Packard), Rich Salz (IBM), Matt Sanchez (Webify Solutions, Inc.), Jerry Schwarz (Oracle Corporation), Senthil Sengodan (Nokia), Shawn Sharp (Cyclone Commerce),

Aslak Siira (F5 Networks, Inc.), David Solo (Citigroup, Inc.), Davanum Srinivas (Computer Associates), Raghavan Srinivas (Sun Microsystems), John Stanton (Defense Information Systems Agency), Andrew Stone (Accenture), Julie Surer (MITRE), Wes Swenson (Forum Systems), Dino Vitale (Citigroup, Inc.), Jonathan Wenocur (IBM), Pete Wenzel (Sun Microsystems), Ian White (Micro Focus)

Appendix C. Revision History

Revision	Date	Editor	Changes Made
[WS01]	[3/6/2013]	[Tom Rutt]	[Transcription of WS-I deliverable moving reference appendices to normative references clause]
WD06	3/18/2014	Tom Rutt	Resolves all PR comments on CSD01