Web Services Security Addendum

18 August 2002

This version:
http://www.ibm.com/developerworks/library/secureadd.html
Authors:
Giovanni Della-Libera, Microsoft
Phillip Hallam-Baker, VeriSign
Maryann Hondo, IBM
Hiroshi Maruyama, IBM
Anthony Nadalin, IBM
Nataraj Nagaratnam, IBM
Hemma Prafullchandra, VeriSign
John Shewchuk, Microsoft
Kent Tamura, IBM
Hervey Wilson, Microsoft
Editor:
Chris Kaler, Microsoft

Abstract

This document describes clarifications, enhancements, best practices, and errata of the WS-Security specification.

Status of this Document

WS-Security and related specifications are provided as-is and for review and evaluation only. IBM and Microsoft and VeriSign hope to solicit your contributions and suggestions in the near future. IBM and Microsoft and VeriSign make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

---

1. Introduction

Since the publication of the WS-Security specification, additional reviews and implementation experiences suggest some additions, clarifications, and corrections to the original specification.

1.1. 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.

Namespace URIs (of the general form "some-URI") represent some application-dependent or context-dependent URI as defined in RFC2396 [URI].

WS-Security is designed to work with the general SOAP[SOAP] message structure and message processing model, and WS-Security should be applicable to any version of SOAP[SOAP]. The current SOAP 1.2 namespace URI is used herein to provide detailed examples, but there is no intention to limit the applicability of this specification to a single version of SOAP[SOAP].

Readers are presumed to be familiar with the terms in the Internet Security Glossary.

1.2. Namespaces

The XML namespace[XML-ns] URIs that MUST be used by implementations of this addendum are as follows (note that different elements are from different namespaces):

```
          http://schemas.xmlsoap.org/ws/2002/07/secext
          http://schemas.xmlsoap.org/ws/2002/07/utility
```

Prefix	Namespace
S	http://www.w3.org/2001/12/soap-envelope
ds	http://www.w3.org/2000/09/xmldsig#
xenc	http://www.w3.org/2001/04/xmlenc#
m	http://schemas.xmlsoap.org/rp
wsse	http://schemas.xmlsoap.org/ws/2002/07/secext
wsu	http://schemas.xmlsoap.org/ws/2002/07/utility
xsd	http://www.w3.org/2001/XMLSchema

2. Errata

In section 4.6.2 of the original specification, the `<xenc:EncryptedData>` element should not have a `<ds:KeyInfo>` element as the key information. Rather, the key name should be specified in the `<xenc:ReferenceList>` element within the `<wsse:Security>` header.

3. ID References

In section 4.5 of the original specification, we discuss the use of the `<ds:Signature>` element. When specifying elements to sign, ID references MAY be used. However, because arbitrary ID attributes require the schemas to be available and processed, we restrict the ID attributes which can be referenced in a signature to the following list:

- ID attributes from XML Signature
- ID attributes from XML Encryption
- wsu:Id global attribute described below

In addition, when signing a part of an envelope such as the body, it is RECOMMENDED that an ID reference is used instead of a more general transformation, especially XPath. This is to simplify processing.

It should be noted that with this specification, the "Id" attributes are dropped from the *wsse* namespace in order to adopt the global namespace qualifier attribute in the *wsu* namespace. Consequently, the examples in the original WS-Security specification will not work with the new wsse schema unless the "Id" attributes are changed to "wsu:Id" and the *wsu* namespace is defined.

3.1. Id Attribute

There are many situations where elements within SOAP[SOAP] messages need to be referenced. For example, when signing a SOAP message, selected elements are included in the signature. XML Schema Part 2[XML-Schema2] provides several built-in data types that may be used for identifying and referencing elements, but their use requires that consumers of the SOAP message either to have or be able to obtain the schemas where the identity or reference mechanisms are defined. In some circumstances, for example, intermediaries, this can be problematic.

Consequently, we need a mechanism for identifying and referencing elements, based on the SOAP foundations, that does not rely upon complete schema knowledge of the context in which an element is used. This functionality can be integrated into SOAP processors so that elements can be identified and referred to without dynamic schema discovery and processing.

In this section we specify a namespace-qualified global attribute for identifying an element which can be applied to any element that either allows arbitrary attributes or specifically allows this attribute.

3.2. Id Schema

To simplify the processing for intermediaries and receivers, we define a common attribute for identifying an element. This attribute utilizes the XML Schema ID type and specifies a common attribute for indicating this information for elements.

The syntax for this attribute is as follows:

```
<anyElement wsu:Id="...">...</anyElement>
```

The following describes the attribute illustrated above:

.../@wsu:Id

This attribute, defined as type `xsd:ID`, provides a well-known attribute for specifying the local ID of an element.

Two `wsu:Id` attributes within an XML document MUST NOT have the same value. Implementations MAY rely on XML Schema validation to provide rudimentary enforcement for intra-document uniqueness. However, applications SHOULD NOT rely on schema validation alone to enforce uniqueness.

We do not specify how this will be used and expect that other specifications MAY add additional semantics (or restrictions) for their usage of this attribute.

The following example illustrates use of this attribute to identify an element:

```
<x:myElement wsu:Id="ID1" xmlns:x="..."
           xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"/>
```

Conformant processors that do support XML Schema MUST treat this attribute as if it was defined using a global attribute declaration.

Conformant processors that do not support XML Schema or DTDs are strongly encouraged to treat this attribute information item as if its PSVI has a [type definition] whose {target namespace} is "http://www.w3.org/2001/XMLSchema" and whose {name} is "Id." Specifically, implementations MAY support the value of the `wsu:Id` as the valid identifier for use as an XPointer[XPointer] shorthand pointer.

4. Placement of X.509 Certificates

The WS-Security specification indicates that X.509 certificates MAY be described inside of a `<ds:KeyInfo>` element, however, it is RECOMMENDED that they be specified using a `<wsse:BinarySecurityToken>`. If, however, an implementation needs to use `<ds:KeyInfo>`, it SHOULD place the `<ds:KeyInfo>` element as a child of the `<wsse:Security>` header rather than embedded within the signature. This allows receivers to have a single processing model.

5. Message Timestamps

When requestors and services are exchanging messages, it is often important to be able to understand the *freshness* of a message. In some cases, a message may be so *stale* that the receiver may decide to ignore it.

We do not provide a mechanism for synchronizing time. We assume either that the receiver is using a mechanism to synchronize time (e.g. NTP) or, more likely for federated applications, that they are making assessments about time based on three factors: creation time of the message, transmission checkpoints, and transmission delays.

To assist a receiver in making an assessment of staleness, a requestor may wish to indicate a suggested expiration time, beyond which the requestor recommends ignoring the message. We provide XML elements by which the requestor may express the expiration time of a message, the requestor's clock time at the moment the message was created, checkpoint timestamps (when an actor received the message) along the communication path, and the delays introduced by transmission and other factors subsequent to creation. The quality of the delays is a function of how well they reflect the actual delays (e.g., how well they reflect transmission delays).

It should be noted that this is not a protocol for making assertions or determining when, or how fast, a service produced or processed a message.

In this specification we define and illustrate time references in terms of the *dateTime* type defined in XML Schema. It is RECOMMENDED that all time references use this type. It is further RECOMMENDED that all references be in UTC time. If, however, other time types are used, then the *ValueType* attribute (described below) MUST be specified to indicate the data type of the time format.

5.1. Model

This specification provides several tools for receivers to use to assess the expiration time presented by the requestor. The first is the Creation time. Receivers can use this value to assess possible clock synchronization issues. However, to make some assessments, the time required to go from the requestor to the receiver may also be useful in making this assessment. Two mechanisms are provided for this. The first is that Intermediaries may add timestamp elements indicating when they received the message. This knowledge can be useful to get a holistic view of clocks along the message path. The second is that intermediaries can specify any delays they imposed on message delivery. It should be noted that not all Intermediaries delays can be accounted for, such as wire time and parties that don't report. Receivers need to take this into account when evaluating clock trust.

5.2. Timestamp Elements

This specification defines the following message timestamp elements. We define these for use with the `<Timestamp>` header for SOAP messages, but they can be used anywhere that creation, expiration, and intermediary markers are needed.

5.2.1. Expiration

The `<Expires>` element specifies the expiration timestamp. The exact meaning and processing rules for expiration depend on the context in which the element is used. The syntax for this element is as follows:

```
      <wsu:Expires  ValueType="..." wsu:Id="...">...</wsu:Expires>
```

The following describes the attributes and elements listed in the schema above:

/Expires

This element's value represents an expiration time. The time specified SHOULD be a UTC format as specified by the ValueType attribute (default is XML Schema[XML-Schema2] type dateTime).

/Expires/@ValueType

This optional attribute specifies the type of the time data. This is specified as the XML Schema type. If this attribute isn't specified, the default value is `xsd:dateTime`.

/Expires/@wsu:Id

This optional attribute specifies an XML Schema ID that can be used to reference this element.

The expiration is relative to the requestor's clock. In order to evaluate the expiration time, receivers need to recognize that the requestor's clock may not be synchronized to the receiver's clock. The receiver, therefore, will need to make a assessment of the level of trust to be placed in the requestor's clock, since the receiver is called upon to evaluate whether the expiration time is in the past relative to the requestor's, not the receiver's, clock. The receiver may make a judgment of the requestor's likely current clock time by means not described in this specification, for example an out-of-band clock synchronization protocol. The receiver may also use the creation time and the delays introduced by intermediate actors to estimate the degree of clock synchronization.

One suggested formula for estimating synchronization is

```
skew = receiver's arrival time - creation time - transmission time
```

Transmission time may be estimated by summing the values of delay elements, if present. It should be noted that wire-time is only part of this if delays include it in estimates. Otherwise the transmission time will not reflect the on-wire time. If no delays are present, no special assumptions about processing time.

5.2.2. Creation

The `created` element specifies a creation timestamp. The exact meaning and semantics are dependent on the context in which the element is used. The syntax for this element is as follows:

```
<wsu:Created ValueType="..." wsu:Id="...">...</wsu:Created>
```

The following describes the attributes and elements listed in the schema above:

/Created

This element's value is a creation timestamp. The time specified SHOULD be a UTC format as specified by the ValueType attribute (default is XML Schema[XML-Schema1] type dateTime).

/Created/@ValueType

This optional attribute specifies the type of the time data. This is specified as the XML Schema type. If this attribute isn't specified, the default value is `xsd:dateTime`.

/Created/@wsu:Id

This optional attribute specifies an XML Schema ID that can be used to reference this element.

5.2.3. Intermediaries

The `Received` element specifies a receipt timestamp with an optional processing delay. The exact meaning and semantics are dependent on the context in which the element is used. The syntax for this element is as follows:

```
<wsu:Received Actor="..." Delay="..." ValueType="..."
              wsu:Id="...">...</wsu:Received>
```

The following describes the attributes and elements listed in the schema above:

/Received

This element's value is a receipt timestamp. The time specified SHOULD be a UTC format as specified by the ValueType attribute (default is XML Schema[XML-Schema1] type dateTime).

/Received/@Actor

A required attribute, Actor, indicates which actor is indicating receipt. Actors SHOULD include this attribute, with a value matching the actor value indicated by the corresponding WS-Routing[WS-Routing] via element, whenever a WS-Routing[WS-Routing] header appears in the message.

/Received/@Delay

The value of this attribute is the delay associated with the actor expressed in milliseconds.

/Received/@ValueType

This optional attribute specifies the type of the time data (the element value). This is specified as the XML Schema type. If this attribute isn't specified, the default value is `xsd:dateTime`.

/Received/@wsu:Id

This optional attribute specifies an XML Schema ID that can be used to reference this element.

The delay attribute indicates the time delay attributable to an actor (intermediate processor). In some cases this isn't known; for others it can be computed as *actor's send time - actor's receipt time*.

Each delay amount is indicated in units of milliseconds, without fractions. If a delay amount would exceed the maximum value expressible in the datatype, the value should be set to the maximum value of the datatype.

5.3. Timestamp Header

A `<Timestamp>` header provides a mechanism for expressing the creation and expiration times of a message and, optionally, the delays introduced throughout the message path. Specifically, is uses the previously defined elements in the context of message creation, receipt, and processing.

All times SHOULD be in UTC format as specified by the XML Schema[XML-Schema2] type (dateTime). It should be noted that times support time precision as defined in the XML Schema[XML-Schema2] specification.

Multiple `<Timestamp>` headers can be specified if they are targeted at different actors. The ordering within the header is as illustrated below.

The ordering of elements in this header is fixed and MUST be preserved by intermediaries.

To preserve overall integrity of each `Timestamp` header, it is strongly RECOMMENDED that each actor create or update the appropriate Timestamp header destined to the particular actor. It is also strongly RECOMMENDED that each actor sign its elem ents by referencing their ID, NOT by signing the Timestamp header as the header is mutable.

The schema outline for the `<Timestamp>` header is as follows:

```
      <wsu:Timestamp wsu:Id="...">
          <wsu:Created>...</wsu:Created>
          <wsu:Expires>...</wsu:Expires>
          <wsu:Received>...</wsu:Received>
          ...
      </wsu:Timestamp>
```

The following describes the attributes and elements listed in the schema above:

/Timestamp

This is the header for indicating message timestamps.

/Timestamp/Created

This represents the Creation time of the message. This element is optional, but can only be specified once in a `Timestamp` header. Within the SOAP processing model, creation is the instant that the infoset is serialized for transmission. The creation time of the message SHOULD NOT differ materially from its transmission time.

/Timestamp/Expires

This represents the Expiration of the message. This is optional, but can appear at most once in a `Timestamp` header. Upon expiration, the requestor asserts that the message is no longer valid. It is strongly RECOMMENDED that receivers (anyone who processes this message) discard (ignore) any message that has passed its expiration. A Fault code (wsu:MessageExpired) is provided if the receiver wants to inform the requestor that its message was expired. A service MAY issue a Fault indicating the message has expired.

/Timestamp/Received

This represents the point in time at which the message was received by a specific actor (See **5.2.3. Intermediaries**). This is optional, but SHOULD appear at most once per actor in a `Timestamp` header (multiple entries MAY exist if looping is present, but the value MUST be different).

/Timestamp/{any}

This is an extensibility mechanism to allow additional elements to be added to the header.

/Timestamp/@wsu:Id

This optional attribute specifies an XML Schema ID that can be used to reference this element.

/Timestamp/@{any}

This is an extensibility mechanism to allow additional attributes to be added to the header.

The following example illustrates the use of the `<Timestamp>` element and its content.

```
  <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
```

```
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
  <S:Header>
    <wsu:Timestamp>
       <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
       <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires>
    </wsu:Timestamp>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

The following example illustrates the use of the `<Timestamp>` header and its content with a received element indicating a processing delay of one minute subsequent to the receipt which was two minutes after creation.

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
  <S:Header>
    <wsu:Timestamp>
       <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
       <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires>
       <wsu:Received Actor="http://x.com/" Delay="60000">
                2001-09-13T08:44:00Z</wsu:Received>
    </wsu:Timestamp>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

6. Passing Passwords

In section 4.1 of the original specification we describe the `<wsse:UsernameToken>` element. Within this element, a `<wsse:Password>` element can be specified. The password has an associated type - either `wsse:PasswordText` or `wsse:PasswordDigest`. The specification describes `wsse:PasswordText` as the *"The actual password for the username."* However the `PasswordText` is not limited to only the actual password. Any password equivalent such as a derived password or S/KEY (one time password) can be used.

The specification also describes `wsse:PasswordDigest` as *"The digest of the password for the username. The value is a base64-encoded SHA1 hash value of the UTF8-encoded password."* However, unless this digested password is sent on a secured channel, the digest offers no real additional security than `wsse:PasswordText`.

To address this, we introduce two new optional elements in the `<wsse:UsernameToken>`: `<wsse:Nonce>` and `<wsu:Created>`. If either of these is present, they are included in the digest value as follows:

```
Password_digest = SHA1 ( nonce + created + password )
```

That is, concatenate the nonce, creation timestamp, and the password (or shared secret or password equivalent) and pass the digest of the combination. This helps obscure the password and offers a basis for preventing replay attacks. It is RECOMMENDED that timestamps and nonces be cached for a minimum of five minutes to detect replays, and that timestamps older than five minutes be rejected.

Note that the nonce is hashed using the octet sequence of its decoded value while the timestamp is hashed using the octet sequence of its UTF8 encoding as specified in the contents of the element.

The following illustrates the syntax of these elements:

```
<wsse:UsernameToken>
    <wsse:Username>...</wsse:Username>
    ...
    <wsse:Nonce EncodingType="...">...</wsse:Nonce>
    <wsu:Created>...</wsu:Created>
</wsse:UsernameToken>
```

The following describes the attributes and elements listed in the example above:

/wsse:Nonce

This optional element specifies a cryptographically random nonce.

/wsse:Nonce/@EncodingType

This optional attribute specifies the encoding type of the nonce (see WS-Security's definition of BinarySecurityToken for valid values). If this attribute isn't specified then the default of Base64 encoding is used.

/wsu:Created

This optional element which specifies a timestamp.

These extensions SHOULD NOT be used unless the plain text password, secret, or password-equivalent is available to both the requestor and the receiver.

The following example illustrates a hashed password using both a nonce and a timestamp:

The following illustrates the syntax of these elements:

```
<wsse:UsernameToken
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext"
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
    <wsse:Username>NNK</wsse:Username>
    <wsse:Password Type=" wsse:PasswordDigest ">FEdR...</wsse:Password>
    <wsse:Nonce>FKJh...</wsse:Nonce>
    <wsu:Created>2001-10-13T09:00:00Z </wsu:Created>
</wsse:UsernameToken>
```

7. Key Identifiers

It is RECOMMENDED to use a key identifier to specify/reference a security token instead of a key name. The `<wsse:KeyIdentifier>` element is placed in the `<wsse:SecurityTokenReference>` element to reference a token using an identifier. This element SHOULD be used for all key identifiers.

The processing model assumes that the key identifier for a security token is constant. Consequently, processing a key identifier is simply looking for a security token whose key identifier matches the specified value.

The following is an overview of the syntax:

```
<wsse:SecurityTokenReference>
   <wsse:KeyIdentifier wsu:Id="..."
                       ValueType="..."
                       EncodingType="...">
      ...
   </wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
```

The following describes the attributes and elements listed in the example above:

/KeyIdentifier

This element is used to include a binary-encoded key identifier.

/KeyIdentifier/@wsu:Id

An optional string label for this identifier.

/KeyIdentifier/@ValueType

The ValueType attribute is used to optionally indicate the type of token with the specified identifier. If specified, this is a hint to the receiver. Any value specified for binary security tokens, or any XML token element QName can be specified here (e.g. wsse:X509v3). If this attribute isn't specified, then the identifier applies to any type of token.

/KeyIdentifier/@EncodingType

The optional EncodingType attribute is used to indicate, using a QName, the encoding format of the binary data (e.g., wsse:Base64Binary). We use the base values defined in WS-Security:

QName	Description
wsse:Base64Binary	XML Schema base 64 encoding (default)
wsse:HexBinary	XML Schema hex encoding

/KeyIdentifier/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

8. Key Names

As previously stated, it is strongly RECOMMEND to use key identifiers, however, if key names are used, then it is strongly RECOMMENDED that `<ds:KeyName>` elements conform to the attribute names in section 2.3 of RFC 2253 (this is recommended by XML Signature for `<X509SubjectName>`) for interoperability.

Additionally, we define the following convention for e-mail addresses, which SHOULD conform to RFC 822:

```
        EmailAddress=ckaler@microsoft.com
```

9. Token Reference Lookup Processing Order

There are a number of mechanisms described in XML Signature, WS-Security, and this specification for referencing security tokens. To resolve ambiguities, the following processing order SHOULD be used:

1. Resolve any `<wsse:Reference>` elements (specified within `<wsse:SecurityTokenReference>`).
2. Resolve any `<wsse:KeyIdentifier>` elements (specified within `<wsse:SecurityTokenReference>`).
3. Resolve any `<ds:KeyName>` elements.
4. Resolve any other `<ds:KeyInfo>` elements.

10. Encrypted Keys

While XML Encryption specifies that `<xenc:EncryptedKey>` elements MAY be specified in `<xenc:EncryptedData>` elements, we strongly RECOMMEND that `<xenc:EncryptedKey>` elements be placed in the `<wsse:Security>` header.

11. Decrypted Transformation

The ordering semantics of the `<wsse:Security>` header are sufficient to determine if signatures are over encrypted or unencrypted data. However, when a signature is included in one `<wsse:Security>` header and the encryption takes place in another `<wsse:Security>` header, the order may not be explicitly understood.

If the sender wishes to sign a message that is subsequently encrypted by an intermediary along the transmission path, the sender MAY use the Decryption Transform for XML Signature to explicitly specify the order of decryption.

12. Certificate Collections

When one service wishes to pass a certificate collection, such as its certificate authority hierarchy, the collection SHOULD be passed using `<wsse:BinarySecurityToken>`. We define the following new token types for passing standard collection formats:

QName	Description
wsse:PKCS7	A PKCS#7 SignedData object, with the only significant field being certificates. In particular, the signature and the contents are ignored. If no certificates are present, a zero-length CertPath is assumed. Warning: PKCS#7 does not maintain the order of certificates in a certification path. This means that if a CertPath is converted to PKCS#7 encoded bytes and then converted back, the order of the certificates may change, potentially rendering the CertPath invalid. Users should be aware of this behavior. See http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/index.html for more information.
wsse:PKIPath	An ASN.1 DER encoded sequence of certificates, defined as follows: PkiPath ::= SEQUENCE OF Certificate Within the sequence, the order of certificates is such that the subject of the first certificate is the issuer of the second certificate, etc. Each certificate in PkiPath shall be unique. No certificate may appear more than once in a value of Certificate in PkiPath. The PkiPath format is defined in defect report 279 against X.509 (2000) and is incorporated into Draft Technical Corrigenda 2 for the fourth edition (2000) of X.509 at ftp://ftp.bull.com/pub/OSIdirectory/DefectResolution/TechnicalCorrigenda/ApprovedTechnicalCorrigendaToX.509/8%7CX.509-TC1(4th).pdf.

13. Security Considerations

In order to *trust* Ids and timestamps, they SHOULD be signed using the mechanisms outlined in WS-Security[WS-Security]. This allows readers of the IDs and timestamps information to be certain that the IDs and timestamps haven't been forged or altered in any way. It is strongly RECOMMENDED that IDs and timestamp elements be signed. Note that since the `Timestamp` header is mutable, signatures need to be associated with individual elements.

Timestamps can also be used to mitigate replay attacks. Signed timestamps MAY be used to keep track of messages (possibly by caching the most recent timestamp from a specific service) and detect replays of previous messages. It is RECOMMENDED that timestamps and nonces be cached for a minimum of five minutes to detect replays, and that timestamps older than five minutes be rejected in interactive scenarios.

14. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including:

Keith Ballinger, Microsoft

Joel Farrell, IBM

Mark Hayes, Verisign

Dan Simon, Microsoft

Wayne Vicknair, IBM

15. References

KEYWORDS
KEYWORDS S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

RFC822
RFC822 "Standard for the Format of ARPA Internet Messages", RFC 822, University of Delaware, August 1982

RFC2253
RFC2253 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", Wahl, Kille, Howes, December 1997.

RFC2459
RFC2459 "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", Housley, Ford, Polk, Solo, January 1999

SHA-1
SHA-1 FIPS PUB 180-1. Secure Hash Standard. U.S. Department of Commerce / National Institute of Standards and Technology. http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.txt

SOAP
SOAP W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

URI
URI T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

XML-Encrypt
XML-Encrypt W3C Working Draft, "XML Encryption Syntax and Processing," 04 March 2002.

XML-ns
XML-ns W3C Recommendation, "Namespaces in XML," 14 January 1999.

XML-Schema1
XML-Schema1 W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001.

XML-Schema2
XML-Schema2 W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.

XML Signature
XML Signature W3C Recommendation, "XML Signature Syntax and Processing," 12 February 2002.

XPointer
XPointer "XML Pointer Language (XPointer) Version 1.0", DeRose, Maler, Daniel, 11 September 2001

WS-Routing
WS-Routing "Web Services Routing Protocol", Microsoft, October 1991

WS-Security
WS-Security "Web Services Security Language", IBM, Microsoft, VersiSign, April 2002

X509
X509 S. Santesson, et al,"Internet X.509 Public Key Infrastructure Qualified Certificates Profile," http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-I

XPath
XPath W3C Recommendation, "XML Path Language", 16 November 1999