WS-Trust 1.3 Errata 01

OASIS Approved Errata

25 April 2012

Specification URIs

This version:

http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/os/ws-trust-1.3-errata01-os.doc (Authoritative)
http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/os/ws-trust-1.3-errata01-os.html
http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/os/ws-trust-1.3-errata01-os.pdf

Previous version:

http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-03.doc (Authoritative)
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-03.html
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-03.pdf

Latest version:

http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/ws-trust-1.3-errata01.doc (Authoritative)
http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/ws-trust-1.3-errata01.html
http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/ws-trust-1.3-errata01.pdf

Technical Committee:

OASIS Web Services Secure Exchange (WS-SX) TC

Chairs:

Kelvin Lawrence (klawrenc@us.ibm.com), IBM
Chris Kaler (ckaler@microsoft.com), Microsoft

Editors:

Anthony Nadalin (tonynad@microsoft.com), Microsoft
Marc Goodner (mgoodner@microsoft.com), Microsoft
Abbie Barbir (abbie.barbir@bankofamerica.com), Bank of America

Related work:

This specification is related to:

- *WS-Trust 1.3*. 19 March 2007. OASIS Standard. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

Abstract:

This document lists errata for *WS-Trust 1.3* produced by the WS-SX Technical Committee.

Status:

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

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/ws-sx/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the

Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/ws-sx/ipr.php).

Citation format:

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

[WS-Trust-1.3-errata]

WS-Trust 1.3 Errata 01. 25 April 2012. OASIS Approved Errata. http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/errata01/os/ws-trust-1.3-errata01-os.html.

Notices

Table of Contents

1 Issues Addressed

The following issues related to WS-Trust 1.3 as recorded in the [WS-SX Issues] have been addressed in this document.

Issue	Description
ER012	Review normative RFC 2119 language in WS-Trust
ER016	SecondaryParameters element missing from WS-Trust schema xsd
ER019	Wrong target namespace in WSDL for WS-Trust 1.3
ER021	Bearer key type URI inconsistent
i169	Sample wsdl in conflict w WS-I BSP in WS-Trust1.3, 1.4
i170	Update XML Signature references to refer to XML Signature, Second Edition, update c14n reference in ws-trust
i171	Incorrect URI provided for Canonical XML 1.0 when defining C14n abbreviation

2 Typographical/Editorial Errors

2.1 Normative references

Insert after line 185

W3C Recommendation, "Canonical XML Version 1.1", 2 May 2008. http://www.w3.org/TR/2008/REC-xml-c14n11-20080502/

Insert after line 201

[W3C Recommendation, D. Eastlake et al. XML Signature Syntax and Processing (Second Edition). 10 June 2008. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

2.2 Normative language capitalization changes

The following changes do not affect the normative meaning of the text, they are only to properly capitalize 2119 terms. The changes listed below document the changes as they appear in the text. There were many instances of the terms OPTIONAL and REQUIRED in the schema exemplar descriptions that appeared un-capitalized that are not captured below but that have also been addressed. All other 2119 terms that remain un-capitalized are used in their English sense.

Line 212

Authentication of requests is based on a combination of OPTIONAL network and transport-provided security and information (claims) proven in the message

Line 231

This model is illustrated in the figure below, showing that any requestor MAY also be a service, and that the Security Token Service is a Web service (that is, it MAY express policy and require security tokens).

Line 242

In the figure above the arrows represent possible communication paths; the requestor MAY obtain a token from the security token service, or it MAY have been obtained indirectly. The requestor then demonstrates authorized use of the token to the Web service. The Web service either trusts the issuing security token service or MAY request a token service to validate the token (or the Web service MAY validate the token itself).

In summary, the Web service has a policy applied to it, receives a message from a requestor that possibly includes security tokens, and MAY have some protection applied to it using [WS-Security] mechanisms.

Line 254

In brokered trust models, the signature MAY NOT verify the identity of the claimant – it MAY verify the identity of the intermediary, who MAY simply assert the identity of the claimant.

Line 259

The trust engine MAY need to externally verify or broker tokens

Line 265

In this specification we define how security tokens are requested and obtained from security token services and how these services MAY broker trust and trust policies so that services can perform step 3.

Line 280

As part of a message flow, a request MAY be made of a security token service to exchange a security token (or some proof) of one form for another

Line 289

the security token service generating the new token MAY NOT need to trust the authority that issued the original token provided by the original requestor since it does trust the security token service that is engaging in the exchange for a new security token

Line 300

An administrator or other trusted authority MAY designate that all tokens of a certain type are

Line 303

or the security token service MAY provide this function as a service to trusting services.

Line 306

These mechanisms are non-normative and are NOT REQUIRED in any way.

Line 313

Trust hierarchies – Building on the trust roots mechanism, a service MAY choose to allow hierarchies of trust so long as the trust chain eventually leads to one of the known trust roots. In some cases the recipient MAY require the sender to provide the full hierarchy. In other cases, the recipient MAY be able to dynamically fetch the tokens for the hierarchy from a token store.

Line 335

or they MAY return a token with their chosen parameters that the requestor MAY then choose to discard because it doesn't meet their needs

Line 339

Other specifications MAY define specific bindings and profiles of this mechanism for additional purposes.

Line 341

in some cases an anonymous request MAY be appropriate

Line 343

If not a fault SHOULD be generated (but is NOT REQUIRED to be returned for denial-of-service reasons).

Line 415 (this one changes a “shouldn’t”)

In general, the returned token SHOULD be considered opaque to the requestor. That is, the requestor SHOULD NOT be required to parse the returned token.

Line 429

and the value of the OPTIONAL @Context attribute

Line 432

In such cases, the RSTR MAY be passed in the body or in a header block.

Line 475

the ellipses below represent the different containers in which this element MAY appear

Line 518

This binding supports the OPTIONAL use of exchanges during the token acquisition process as well as the OPTIONAL use of the key extensions described in a later section.

Line 522

the following OPTIONAL elements

Line 561

This REQUIRED attribute contains a URI that indicates the syntax used to specify the set of requested claims along with how that syntax SHOULD be interpreted.

Line 574

The format is assumed to be understood by the requestor because the value space MAY be

Line 580

The issuer is not obligated to honor this range – they MAY

Line 587

The difference in time SHOULD be minimized.

Line 697

Each request MAY generate more than one RSTR sharing the same Context attribute value

Line 711

Note: that these operations require that the service can either succeed on all the RST requests or MUST NOT perform any partial operation.

Line 722

If any error occurs in the processing of the RSTC or one of its contained RSTs, a SOAP fault MUST be generated for the entire batch request so no RSTC element will be returned.

Line 741

the following OPTIONAL elements

Line 833

The token issuer can OPTIONALLY provide

Line 990

As a result, the proof-of-possession tokens, and possibly lifetime and other key parameters elements, MAY be different

Line 1071

If confidentiality protection of the <wst:IssuedTokens> header is REQUIRED then the entire header MUST be encrypted using the <wsse11:EncryptedHeader> construct.

Line 1131

and the OPTIONAL <wst:Lifetime> element

Line 1167

This OPTIONAL element indicates that returned tokens SHOULD allow requests for postdated tokens.

Line 1225

If a client needs to ensure the validity of a token, it MUST validate the token at the issuer.

Line 1292

this section defines an OPTIONAL binding

Line 1354

The result MAY be a status, a new token, or both.

Line 1370

The request provides a token upon which the request is based and OPTIONAL tokens. As well, the OPTIONAL <wst:TokenType> element

Line 1371

This MAY be any supported token type or it MAY be the following URI indicating that only status is desired:

Line 1378

which is OPTIONAL

Line 1467

However, there are many scenarios where a set of exchanges between the parties is REQUIRED prior to returning (e.g., issuing) a security token.

Line 1487

with the issued security token and OPTIONAL proof-of-possession token

Line 1502

(and MAY contain initial negotiation/challenge information)

Line 1504

Optionally, this MAY return token information

Line 1572

Exchange requests MAY also utilize existing binary formats

Line 1579

ellipses below indicate that this element MAY be placed in different containers

Line 1602

In some cases it MAY be necessary to provide a key exchange token so that the other party (either requestor or issuer) can provide entropy or key material as part of the exchange. Challenges MAY NOT always provide a usable key as the signature may use a signing-only certificate.

Line 1606

The section describes two OPTIONAL elements

Line 1608

ellipses below indicate that this element MAY be placed in different containers

Line 1617

This OPTIONAL element is used to indicate that the receiving party (either the original requestor or issuer) SHOULD provide a KET to the other party on the next leg of the exchange.

Line 1822

This MAY be built into the exchange messages

Line 1832

To this end, the following computed key algorithm is defined to be OPTIONALLY used in these scenarios

Line 1837

However, until the exchange is actually completed it MAY be (and is often) inappropriate to use the computed keys. As well, using a token that hasn't been returned to secure a message may (no change, English) complicate processing since it crosses the boundary of the exchange and the underlying message security. This means that it MAY NOT be appropriate to sign the final leg of the exchange using the key derived from the exchange.

Line 1874

This <wst:CombinedHash> element is OPTIONAL

Line 1878

since all types of requests MAY issue security tokens they could apply to other bindings

Line 1924

The syntax for these OPTIONAL elements is as follows

Line 1950

That is, requestors SHOULD be familiar with the recipient policies

Line 1996

This element either contains a security token or a <wsse:SecurityTokenReference> element that references the security token containing the key that SHOULD be used in the returned token.

Line 2037

EncryptionAlgorithm – used to indicate the symmetric algorithm that the STS SHOULD use to encrypt the T (e.g. AES256)

Line 2043

EncryptionAlgorithm – used to indicate the symmetric algorithm that the STS SHOULD use to encrypt T for RP (e.g. AES256)

KeyWrapAlgorithm – used to indicate the KeyWrap algorithm that the STS SHOULD use to wrap the generated key that is used to encrypt the T for RP

Line 2052

EncryptionAlgorithm – used to indicate the symmetric algorithm that the STS SHOULD use to encrypt T for RP (e.g. AES256)

Line 2059

EncryptionAlgorithm - used to indicate the symmetric algorithm that the STS SHOULD use to encrypt T for RP (e.g. AES256)

KeyWrapAlgorithm – used to indicate the KeyWrap algorithm that the STS SHOULD use to wrap the generated key that is used to encrypt the T for RP

Line 2140

This OPTIONAL element, of type xs:boolean, specifies whether the requested security token SHOULD be marked as "Forwardable”

Line 2145

This OPTIONAL element, of type xs:boolean, specifies whether the requested security token SHOULD be marked as "Delegatable".

Line 2224

Arbitrary types MAY be used to specify participants

Line 2248

OPTINALLY the <wst:TokenType> element can be specified in the request and can indicate

Line 2363

Other specifications and profiles MAY provide additional details on key exchange

Line 2376

In these cases both parties SHOULD contribute entropy to the key exchange by means of the <wst:entropy> element

Line 2403

If the requestor provides key material that the recipient doesn't accept, then the issuer SHOULD reject the request.

Line 2492

A third party MAY also act as a broker to transfer keys

Line 2631

The perfect forward secrecy property MAY be achieved by

2.3 Section 9.2 Key and Encryption Requirements

Changed Bearer URI in table at 1956 from

http://docs.oasis-open.org/ws-sx/wstrust/200512/Bearer

to

http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer

2.4 WSDL changes

The WSDL was replaced with a more representative example that better illustrates usage of the protocol.

2.5 Schema Changes

Missing SecondaryParameters element added to section 3.1 in external schema file.

Corrected Bearer URI in KeyTypeEnum simple type from

http://docs.oasis-open.org/wssx/wstrust/200512/Bearer

to

http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer

3 Normative Errors

None.

4 References

[WS-SX Issues]	WS-SX TC Issues List http://docs.oasis-open.org/ws-sx/issues/Issues.xml
[WS-Trust]	OASIS Standard, “WS-Trust 1.3", March 2007 http://docs.oasis-open.org/ws-sx/ws-trust/200512

Appendix A. Acknowledgements

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

TC Members during the development of this specification:

Don Adams, Tibco Software Inc.

Jan Alexander, Microsoft Corporation

Steve Anderson, BMC Software

Donal Arundel, IONA Technologies

Howard Bae, Oracle Corporation

Abbie Barbir, Nortel Networks Limited

Charlton Barreto, Adobe Systems

Mighael Botha, Software AG, Inc.

Toufic Boubez, Layer 7 Technologies Inc.

Norman Brickman, Mitre Corporation

Melissa Brumfield, Booz Allen Hamilton

Lloyd Burch, Novell

Scott Cantor, Internet2

Greg Carpenter, Microsoft Corporation

Steve Carter, Novell

Symon Chang, BEA Systems, Inc.

Ching-Yun (C.Y.) Chao, IBM

Martin Chapman, Oracle Corporation

Kate Cherry, Lockheed Martin

Henry (Hyenvui) Chung, IBM

Luc Clement, Systinet Corp.

Paul Cotton, Microsoft Corporation

Glen Daniels, Sonic Software Corp.

Peter Davis, Neustar, Inc.

Martijn de Boer, SAP AG

Werner Dittmann, Siemens AG

Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory

Fred Dushin, IONA Technologies

Petr Dvorak, Systinet Corp.

Colleen Evans, Microsoft Corporation

Ruchith Fernando, WSO2

Mark Fussell, Microsoft Corporation

Vijay Gajjala, Microsoft Corporation

Marc Goodner, Microsoft Corporation

Hans Granqvist, VeriSign

Martin Gudgin, Microsoft Corporation
Tony Gullotta, SOA Software Inc.
Jiandong Guo, Sun Microsystems
Phillip Hallam-Baker, VeriSign
Patrick Harding, Ping Identity Corporation
Heather Hinton, IBM
Frederick Hirsch, Nokia Corporation
Jeff Hodges, Neustar, Inc.
Will Hopkins, BEA Systems, Inc.
Alex Hristov, Otecia Incorporated
John Hughes, PA Consulting
Diane Jordan, IBM
Venugopal K, Sun Microsystems
Chris Kaler, Microsoft Corporation
Dana Kaufman, Forum Systems, Inc.
Paul Knight, Nortel Networks Limited
Ramanathan Krishnamurthy, IONA Technologies
Christopher Kurt, Microsoft Corporation
Kelvin Lawrence, IBM
Hubert Le Van Gong, Sun Microsystems
Jong Lee, BEA Systems, Inc.
Rich Levinson, Oracle Corporation
Tommy Lindberg, Dajeil Ltd.
Mark Little, JBoss Inc.
Hal Lockhart, BEA Systems, Inc.
Mike Lyons, Layer 7 Technologies Inc.
Eve Maler, Sun Microsystems
Ashok Malhotra, Oracle Corporation
Anand Mani, CrimsonLogic Pte Ltd
Jonathan Marsh, Microsoft Corporation
Robin Martherus, Oracle Corporation
Miko Matsumura, Infravio, Inc.
Gary McAfee, IBM
Michael McIntosh, IBM
John Merrells, Sxip Networks SRL
Jeff Mischkinsky, Oracle Corporation
Prateek Mishra, Oracle Corporation
Bob Morgan, Internet2
Vamsi Motukuru, Oracle Corporation
Raajmohan Na, EDS
Anthony Nadalin, IBM
Andrew Nash, Reactivity, Inc.

Eric Newcomer, IONA Technologies
Duane Nickull, Adobe Systems
Toshihiro Nishimura, Fujitsu Limited
Rob Philpott, RSA Security
Denis Pilipchuk, BEA Systems, Inc.
Darren Platt, Ping Identity Corporation
Martin Raepple, SAP AG
Nick Ragouzis, Enosis Group LLC
Prakash Reddy, CA
Alain Regnier, Ricoh Company, Ltd.
Irving Reid, Hewlett-Packard
Bruce Rich, IBM
Tom Rutt, Fujitsu Limited
Maneesh Sahu, Actional Corporation
Frank Siebenlist, Argonne National Laboratory
Joe Smith, Apani Networks
Davanum Srinivas, WSO2
Yakov Sverdlov, CA
Gene Thurston, AmberPoint
Victor Valle, IBM
Asir Vedamuthu, Microsoft Corporation
Greg Whitehead, Hewlett-Packard
Ron Williams, IBM
Corinna Witt, BEA Systems, Inc.
Kyle Young, Microsoft Corporation