WS-Trust 1.3 Errata

Committee Draft 02

15 August 2008

Specification URIs:
This Version:
> http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-02.doc
> http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-02.pdf
> http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-02.html

Previous Version:
> http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-01.doc
> http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-01.pdf
> http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-errata-cd-01.html

Latest Approved Version:
> N/A

Technical Committee:
> OASIS WS-TX TC

Chair(s):
> Kelvin Lawrence, IBM
> Chris Kaler, Microsoft

Editor(s):
> Anthony Nadalin, IBM
> Marc Goodner, Microsoft
> Abbie Barbir, Nortel

Related work:
> This specification errata is related to WS-Trust v1.3.

Abstract:
> This document lists errata for **WS-Trust 1.3 OASIS Standard** [WS-Trust] produced by the WS-SX Technical Committee. The standard was approved by the OASIS membership on 1 March 2007.

Status:
> This document was last revised or approved by the WS-SX TC on the above date. The level of approval is also listed above. Check the "Latest Approved 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 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 (www.oasis-open.org/committees/ws-sx/ipr.php).

> The non-normative errata page for this specification is located at www.oasis-open.org/committees/ws-sx.

Notices

Copyright © OASIS Open 2008. All Rights Reserved.

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.

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

2 Typographical/Editorial Errors

2.1 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

46   the security token service generating the new token MAY NOT need to trust the authority that issued the
47   original token provided by the original requestor since it does trust the security token service that is
48   engaging in the exchange for a new security token
49
50   Line 300
51   An administrator or other trusted authority MAY designate that all tokens of a certain type are
52
53   Line 303
54   or the security token service MAY provide this function as a service to trusting services.
55
56   Line 306
57   These mechanisms are non-normative and are NOT REQUIRED in any way.
58
59   Line 313
60   Trust hierarchies – Building on the trust roots mechanism, a service MAY choose to allow hierarchies of
61   trust so long as the trust chain eventually leads to one of the known trust roots. In some cases the
62   recipient MAY require the sender to provide the full hierarchy. In other cases, the recipient MAY be able
63   to dynamically fetch the tokens for the hierarchy from a token store.
64
65   Line 335
66   or they MAY return a token with their chosen parameters that the requestor MAY then choose to discard
67   because it doesn't meet their needs
68
69   Line 339
70   Other specifications MAY define specific bindings and profiles of this mechanism for additional purposes.
71
72   Line 341
73   in some cases an anonymous request MAY be appropriate
74
75   Line 343
76   If not a fault SHOULD be generated (but is NOT REQUIRED to be returned for denial-of-service reasons).
77
78   Line 415 (this one changes a "shouldn't")
79   In general, the returned token SHOULD be considered opaque to the requestor. That is, the requestor
80   SHOULD NOT be required to parse the returned token.
81
82   Line 429
83   and the value of the OPTIONAL @Context attribute
84
85   Line 432
86   In such cases, the RSTR MAY be passed in the body or in a header block.
87
88   Line 475

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

90

91   Line 518

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

94

95   Line 522

96   the following OPTIONAL elements

97

98   Line 561

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

101

102   Line 574

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

104

105   Line 580

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

107

108   Line 587

109   The difference in time SHOULD be minimized.

110

111   Line 697

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

113

114   Line 711

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

117

118   Line 722

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

121

122   Line 741

123   the following OPTIONAL elements

124

125   Line 833

126   The token issuer can OPTIONALLY provide

127

128   Line 990

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

131

132   Line 1071

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

135

136   Line 1131

137   and the OPTIONAL <wst:Lifetime> element

138

139   Line 1167

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

141

142   Line 1225

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

144

145   Line 1292

146   this section defines an OPTIONAL binding

147

148   Line 1354

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

150

151   Line 1370

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

154

155   Line 1371

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

158

159   Line 1378

160   which is OPTIONAL

161

162   Line 1467

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

165

166   Line 1487

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

168

169   Line 1502

170   (and MAY contain initial negotiation/challenge information)

171

172   Line 1504

173   Optionally, this MAY return token information

174

175   Line 1572

176   Exchange requests MAY also utilize existing binary formats

177

178   Line 1579

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

180

181   Line 1602

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

185

186   Line 1606

187   The section describes two OPTIONAL elements

188

189   Line 1608

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

191

192   Line 1617

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

195

196   Line 1822

197   This MAY be built into the exchange messages

198

199   Line 1832

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

201

202   Line 1837

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

208

209   Line 1874

210   This <wst:CombinedHash> element is OPTIONAL

211

212   Line 1878

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

214

215   Line 1924

216   The syntax for these OPTIONAL elements is as follows

217

218   Line 1950

219   That is, requestors SHOULD be familiar with the recipient policies

220

221   Line 1996

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

224

225   Line 2037

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

228

229   Line 2043

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

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

234

235   Line 2052

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

238

239   Line 2059

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

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

244

245   Line 2140

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

248

249   Line 2145

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

252

253   Line 2224

254   Arbitrary types MAY be used to specify participants

255

256   Line 2248

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

258

259   Line 2363

260   Other specifications and profiles MAY provide additional details on key exchange

261

262 Line 2376

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

265

266 Line 2403

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

269

270 Line 2492

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

272

273 Line 2631

274 The perfect forward secrecy property MAY be achieved by

275 2.2 WSDL changes

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

3 Normative Errors

278 None.

4 References

280	[WS-SX Issues]	WS-SX TC Issues List
281		http://docs.oasis-open.org/ws-sx/issues/Issues.xml
282	[WS-Trust]	OASIS Standard, "WS-Trust 1.3", March 2007
283		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

324	Martin Gudgin, Microsoft Corporation
325	Tony Gullotta, SOA Software Inc.
326	Jiandong Guo, Sun Microsystems
327	Phillip Hallam-Baker, VeriSign
328	Patrick Harding, Ping Identity Corporation
329	Heather Hinton, IBM
330	Frederick Hirsch, Nokia Corporation
331	Jeff Hodges, Neustar, Inc.
332	Will Hopkins, BEA Systems, Inc.
333	Alex Hristov, Otecia Incorporated
334	John Hughes, PA Consulting
335	Diane Jordan, IBM
336	Venugopal K, Sun Microsystems
337	Chris Kaler, Microsoft Corporation
338	Dana Kaufman, Forum Systems, Inc.
339	Paul Knight, Nortel Networks Limited
340	Ramanathan Krishnamurthy, IONA Technologies
341	Christopher Kurt, Microsoft Corporation
342	Kelvin Lawrence, IBM
343	Hubert Le Van Gong, Sun Microsystems
344	Jong Lee, BEA Systems, Inc.
345	Rich Levinson, Oracle Corporation
346	Tommy Lindberg, Dajeil Ltd.
347	Mark Little, JBoss Inc.
348	Hal Lockhart, BEA Systems, Inc.
349	Mike Lyons, Layer 7 Technologies Inc.
350	Eve Maler, Sun Microsystems
351	Ashok Malhotra, Oracle Corporation
352	Anand Mani, CrimsonLogic Pte Ltd
353	Jonathan Marsh, Microsoft Corporation
354	Robin Martherus, Oracle Corporation
355	Miko Matsumura, Infravio, Inc.
356	Gary McAfee, IBM
357	Michael McIntosh, IBM
358	John Merrells, Sxip Networks SRL
359	Jeff Mischkinsky, Oracle Corporation
360	Prateek Mishra, Oracle Corporation
361	Bob Morgan, Internet2
362	Vamsi Motukuru, Oracle Corporation
363	Raajmohan Na, EDS
364	Anthony Nadalin, IBM
365	Andrew Nash, Reactivity, Inc.

366   Eric Newcomer, IONA Technologies
367   Duane Nickull, Adobe Systems
368   Toshihiro Nishimura, Fujitsu Limited
369   Rob Philpott, RSA Security
370   Denis Pilipchuk, BEA Systems, Inc.
371   Darren Platt, Ping Identity Corporation
372   Martin Raepple, SAP AG
373   Nick Ragouzis, Enosis Group LLC
374   Prakash Reddy, CA
375   Alain Regnier, Ricoh Company, Ltd.
376   Irving Reid, Hewlett-Packard
377   Bruce Rich, IBM
378   Tom Rutt, Fujitsu Limited
379   Maneesh Sahu, Actional Corporation
380   Frank Siebenlist, Argonne National Laboratory
381   Joe Smith, Apani Networks
382   Davanum Srinivas, WSO2
383   Yakov Sverdlov, CA
384   Gene Thurston, AmberPoint
385   Victor Valle, IBM
386   Asir Vedamuthu, Microsoft Corporation
387   Greg Whitehead, Hewlett-Packard
388   Ron Williams, IBM
389   Corinna Witt, BEA Systems, Inc.
390   Kyle Young, Microsoft Corporation