1 OASIS

Web Services Security UsernameToken Profile

Working Draft 2, Sunday, 23 February 2003

Document identifier:

{draft}-{*WSS: SOAP Message Security* }-{UsernameToken Profile }-{*1.0*} (Word) (PDF)

Location:

http://www.oasis-open.org/committees/wss

Editor:

TBD <email address goes here>

Contributors:

TEXT TO BE REVISED TO INCLUDE CONTRIBUTORS

Abstract:

This document describes how to use the UsernameToken with the Web Services Security (WSS) specification.

Status:

This is a working draft submitted for consideration by the OASIS Web Services Security (WSS) technical committee. Please send comments to the editors.

If you are on the wss@lists.oasis-open.org list for committee members, send comments there. If you are not on that list, subscribe to the wss-comment@lists.oasis-open.org list and send comments there. To subscribe, send an email message to wss-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For patent disclosure information that may be essential to the implementation of this specification, and any offers of licensing terms, refer to the Intellectual Property Rights section of the OASIS Security Services Technical Committee (SSTC) web page at http://www.oasis-open.org/who/intellectualproperty.shtml.

Table of Contents

1 Introduction

This document describes how to use the UsernameToken with the Web Services Security (WSS) specification. More specifically, it describes how a web service consumer can supply a UsernameToken as a means of identifying the requestor by "username", and optionally using a password (or shared secret, or password equivalent) to authenticate that identity to the web service producer

Section 1 is non-normative.

2 Terminology

The key words *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 [12].

Namespace URIs (of the general form "some-URI") represent some application-dependent or context-dependent URI as defined in RFC 2396 [13].

This specification design is intended to work with any version the general SOAP [3] message structure and processing model, though the SOAP 1.2 namespace URI is used in examples.

Commonly used security terms are defined in the Internet Security Glossary [14].

The namespaces used in this document are shown in the following table.

Prefix	Namespace
S	http://www.w3.org/2001/12/soap-envelope
wsse	http://schemas.xmlsoap.org/ws/2002/xx/secext

3 Acronyms and Abbreviations

Term	Definition
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol

URI	Uniform Resource Identifier
UCS	Universal Character Set
UTF8	UCS Transformation Format, 8-bit form
XML	Extensible Markup Language

4 UsernameToken Extensions

67

4.1 Usernames and Passwords

68

69 The `<wsse:UsernameToken>` element is introduced in the WSS-SOAP Message Security
70 documents as a way of providing a username.
71
72 Within this element, a `<wsse:Password>` element may be specified. Passwords of type
73 `wsse:PasswordText` are not limited to actual passwords, although this is a common case. Any
74 password equivalent such as a derived password or S/KEY (one time password) can be used.
75 Having a type of `wsse:PasswordText` merely implies that the information held in the password
76 is "in the clear", as opposed to holding a "digest" of the information..For example, if a server does
77 not have access to the clear text of a password but does have the hash, then the hash is
78 considered a *password equivalent* and can be used anywhere where a "password" is indicated in
79 this specification. It is not the intention of this specification to require that all implementations
80 have access to clear text passwords.
81
82 Passwords of type wsse:PasswordDigest are defined as being the Base64 [16] encoded, SHA-1
83 hash value, of the UTF8 [17] encoded password (or equivalent).*.* However, unless this digested
84 password is sent on a secured channel, the digest offers no real additional security over use of
85 `wsse:PasswordText`.
86
87 To address this issue, two optional elements are introduced in the `<wsse:UsernameToken>`
88 element: `<wsse:Nonce>` and `<wsu:Created>`. If either or both of these are present, they must
89 be
90 included in the digest value as follows:
91
92 Password_Digest = Base64 (SHA-1 (nonce + created + password))
93
94 That is, concatenate the nonce, creation timestamp, and the password (or shared secret or
95 password equivalent), digest the combination using the SHA-1 has algorithm, then include the
96 Base64 encoding of that result as the Password (digest). This helps obscure the password and
97 offers a basis for preventing replay attacks. For web service providers to effectively thwart replay
98 attacks, three counter measures are recommended:
99   1. First, it is recommended that web service providers reject any UsernameToken *not*
100   using *both* nonce *and* creation timestamps.
101   2. Second, it is recommended that web service producers provide a timestamp
102   "freshness" limitation, and that any UsernameToken with "stale" timestamps be

103   rejected. As a guideline, a value of five minutes can be used as a minimum to
104   detect, and thus reject, replays.

105   3.   Third, it is recommended that used nonces be cached for a period at least as long
106   as the timestamp freshness limitation period, above, and that UsernameTokens with
107   nonces that have already been used (and are thus in the cache) be rejected

108

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

112

113   Note that passwords of either type (wsse:PasswordText or wsse:PasswordDigest) can only be
114   used if the plain text password (or password equivalent) is available to both the requestor and the
115   recipient..

116

117   The following illustrates the XML [2] syntax of this element:

118

```
119    <wsse:UsernameToken wsu:Id="Example-1">
120       <wsse:Username> ... </wsse:Username>
121       <wsse:Password Type="..."> ... </wsse:Password>
122       <wsse:Nonce EncodingType="..."> ... </wsse:Nonce>
123       <wsu:Created> ... </wsu:Created>
124    </wsse:UsernameToken>
```

125

126   The following describes the attributes and elements listed in the example above:
127   */wsse:UsernameToken/Password*
128   This optional element provides password information (or equivalent such as a hash). It is
129   recommended that this element only be passed when a secure transport is being used.

130

131   */wsse:UsernameToken/Password/@Type*
132   This optional attribute specifies the type of password being provided. The following table
133   identifies the pre-defined types:

134

135

Value	Description
wsse:PasswordText (default)	The actual password for the username, the password hash, or derived password or S/KEY.
wsse:PasswordDigest	The digest of the password (and optionally nonce and/or creation timestame) for the username using the algorithm described above.

136

137   */wsse:UsernameToken/Password/@{any}*
138   This is an extensibility mechanism to allow additional attributes, based on schemas, to be
139   added to the element.

140

141   */wsse:UsernameToken/wsse:Nonce*
142   This optional element specifies a cryptographically random nonce. Each message
143   including a Nonce element should use a new nonce value in order for web service providers to
144   detect replay attacks

145
146 */wsse:UsernameToken/wsse:Nonce/@EncodingType*
147   This optional attribute specifies the encoding type of the nonce (see the definition of
148   <wsse:BinarySecurityToken> for valid values). If this attribute isn't specified then the
149   default of Base64 encoding is used.
150
151 */wsse:UsernameToken/wsu:Created*
152   This optional element which specifies a timestamp. The element is used to indicate the
153   creation time.
154
155 All compliant implementations must be able to process the `<wsse:UsernameToken>` element.
156 The following example illustrates the use of this element. In this example the password is sent as
157 clear text and therefore this message should be sent over a confidential channel:
158

```
159  <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
160     xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext">
161     <S:Header>
162        ...
163        <wsse:Security>
164           <wsse:UsernameToken >
165              <wsse:Username> Zoe </wsse:Username>
166              <wsse:Password> ILoveDogs </wsse:Password>
167           </wsse:UsernameToken>
168        </wsse:Security>
169        ...
170     </S:Header>
171     ...
172  </S:Envelope>
```

173
174 The following example illustrates using a digest of the password along with a nonce and creation
175 timestamp:
176

```
177  <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
178     xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext">
179     <S:Header>
180        ...
181        <wsse:Security>
182           <wsse:UsernameToken
183              xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext"
184              xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/xx/utility">
185              <wsse:Username> NNK </wsse:Username>
186              <wsse:Password Type="wsse:PasswordDigest">
187                 D2A12DFE8D9F0C6BB82C89B091DF5C8A872F94DC
188              </wsse:Password>
189              <wsse:Nonce> EFD89F06CCB28C89 </wsse:Nonce>
190              <wsu:Created> 2001-10-13T09:00:00Z </wsu:Created>
191           </wsse:UsernameToken>
192        </wsse:Security>
193        ...
194     </S:Header>
195     ...
196  </S:Envelope>
```

197

198   4.2 Error Codes

199   Implementations may use custom error codes defined in private namespaces if needed. But it is
200   recommended that they use the error handling codes defined in the WSS: SOAP Message
201   Security specification for signature, decryption, encoding and token header errors. When using
202   custom error codes, implementations should be careful not to introduce security vulnerabilities
203   that may assist an attacker in the error codes returned.

204   4.3 Threat Model

205   The use of the UsernameToken introduces no new threats beyond those already identified for
206   other types of SecurityTokens. Replay attacks can be addressed by using message timestamps,
207   nonces, and caching, as well as other application-specific tracking mechanisms. Token
208   ownership is verified by use of keys and man-in-the-middle attacks are generally mitigated.
209   Transport-level security may be used to provide confidentiality and integrity of both the Username
210   token and the entire message body.

211

212   5 References

213   [1]   W3C Extensible Markup Language (XML) 1.0 (Second Edition), W3C
214   Recommendation, Copyright © [6 October 2000] World Wide Web
215   Consortium, (Massachusetts Institute of Technology, Institut National de
216   Recherche en Informatique et en Automatique, Keio University),
217   http://www.w3.org/TR/2000/REC-xml-20001006/.

218   [2]   W3C SOAP 1.1:2000, Simple Object Access Protocol (Note), W3C
219   Recommendation, Copyright © 2000 World Wide Web Consortium,
220   (Massachusetts Institute of Technology, Institut National de Recherche
221   en Informatique et en Automatique, Keio University,
222   http://www.w3.org/TR/SOAP/.

223   [3]   S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,
224   http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

225   [4]   T. Berners-Lee, Uniform Resource Identifiers (URI): General Syntax,
226   http://www.ietf.org/rfc/rfc2396.txt, IETF RFC 2396, August 1998.

227   [5]   R. Shirley, Internet Security Glossary, http://www.ietf.org/rfc/rfc2828.txt,
228   IETF RFC 2828, May 2000.

229   [6]   N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions
230   (MIME) Part 1: Format of Internet Message Bodies,
231   http://www.ietf.org/rfc/rfc2045.txt, IETF RFC 2045, November 1996.

232   [7]   The Unicode Standard, Version 3.2.0:2002. The Unicode Consortium.
233   (Reading, MA Addison-Wesley)

Appendix A. Acknowledgments

234

235 The following individuals were members of the committee during the development of this
236 specification:

237

238 • TBD

Appendix B. Revision History

Rev	Date	By Whom	What
Wd-1.0	2002-12-16	Phil Griffin	Initial version cloned from the WSS core specification
Wd-1.1	2003-01-26	Anthony Nadalin	Bring in line with WSS-Core Update
Wd-1.2	2003-02-23	Anthony Nadalin	Editorial Updates

Appendix C. Notices

240

241 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
242 that might be claimed to pertain to the implementation or use of the technology described in this
243 document or the extent to which any license under such rights might or might not be available;
244 neither does it represent that it has made any effort to identify any such rights. Information on
245 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
246 website. Copies of claims of rights made available for publication and any assurances of licenses
247 to be made available, or the result of an attempt made to obtain a general license or permission
248 for the use of such proprietary rights by implementors or users of this specification, can be
249 obtained from the OASIS Executive Director.

250 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
251 applications, or other proprietary rights which may cover technology that may be required to
252 implement this specification. Please address the information to the OASIS Executive Director.