

1

2 Web Services Security 3 OtpToken Profile 1.0

4 Input document – October, 27, 2005

5 **Document identifier:**

6 verisign-wss-otp-1-0.pdf

7 **Document Location:**

8 tbd

9 **Errata Location:**

10 tbd

11 **Editors:**

Hans	Granqvist	VeriSign	12

13

14 **Contributors:**

John	Linn	RSA Laboratories

15

16 **Abstract:**

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

19 **Status:**

20 This is a technical committee input document submitted for consideration by the OASIS
21 Web Services Security (WSS) technical committee. See Appendix B Notices for legal
22 statements.

23 **Table of Contents**

24 1 Introduction 3
25 2 Notations and terminology 4
26 2.1 Notational conventions..... 4
27 2.2 Namespaces 4
28 3 XMLToken extension 5
29 3.1 One-time-passwords 5
30 4 Token reference 10
31 4.1 Error codes..... 10
32 5 Open issues 12
33 5.1 Referencing an OTP 12
34 5.2 Key derivation 14
35 6 Security considerations..... 15
36 7 References..... 16
37 Appendix A. Revision History 17
38 Appendix B. Notices 18
39

40 1 Introduction

41 This document describes how to use the OtpToken with the WSS: SOAP Message Security
42 specification [WSS]. More specifically, it describes how a web service consumer can supply an
43 OtpToken as a means of identifying the requestor by a one time password and to authenticate
44 that identity to the web service producer.

45 The OtpToken is designed to support devices in three basic modes:

46 **Time-based mode**, in which the token device generates a result value as a function of an internal
47 clock and other internal data, without external inputs.

48 **Counter mode**, in which the token device generates a result value as a function of an internal
49 counter and other internal data, without external inputs.

50 **Challenge-response mode**, in which the token device generates its result as a function of an
51 input challenge value as well as data maintained internal to the device.

52 This section is non-normative.

53 2 Notations and terminology

54 This section specifies the notations, namespaces, and terminology used in this specification.

55 2.1 Notational conventions

56 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
57 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be
58 interpreted as described in [RFC 2119].

59 2.2 Namespaces

60 The namespaces used in this document are shown in the following table (note that for brevity, the
61 examples use the prefixes listed below but do not include the URIs – those listed below are
62 assumed).

Prefix	Namespace
s	http://schemas.xmlsoap.org/soap/envelope/
wsse	http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-wssecurity-secext-1.1.xsd
wsu	http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-wssecurity-utility-1.1.xsd
otp	http://tbd

63 The following table lists the full URI for each URI fragment referred to in this specification.

URI Fragment	Full URI
TBD	TBD

64 3 XMLToken extension

65 The [WSS] specification defines how to attach security tokens. There exist three predefined token
66 types: one for user names, <wsse:UsernameToken>, one for binary tokens,
67 <wsse:BinarySecurityToken>, and an arbitrary token extension mechanism.

68 A one-time-password is similar to a username/password combination, but attempts to define OTP
69 use in [WSS-UNT] proved unnecessarily convoluted, even if such an overloaded use have its
70 merits.

71 3.1 One-time-passwords

72 The <otp:OtpToken> element is an example of an extended XML token (defined in [WSS]
73 section 6.4) as a way of providing a one-time-password:

```
74 <s:Envelope>  
75   <s:Header>  
76     <wsse:Security>  
77       ...  
78       <otp:OtpToken Algorithm="..." wsu:Id="Example-1">  
79         <otp:State EncodingType="..."/>  
80         <otp:Entity Type="..."/>  
81         <otp:Otp EncodingType="..."/>  
82         <otp:Pin/>  
83         <wsse:Nonce EncodingType="..."/>  
84         <wsu:Created/>  
85       </otp:OtpToken>  
86       ...  
87     </wsse:Security>  
88   </s:Header>  
89   <s:Body>  
90     ...  
91   </s:Body>  
92 </s:Envelope>
```

93 The <otp:OtpToken> element represents a one-time-password. Inside the element, a number
94 of elements may be specified.

95 Two optional elements were introduced in the `<wsse:UsernameToken>` profile to provide a
96 countermeasure for replay attacks: `<wsse:Nonce>` and `<wsu:Created>`. This profile follows
97 the same usage and recommendations.

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

99 `/otp:otpToken/@Algorithm`

100 This URI attribute specifies the algorithm of one-time-password being provided. The table
101 below identifies the pre-defined algorithms (note that the URI fragments are relative to
102 the URI for this specification).

URI	Description
<code>#oath-hotp</code>	OATH HMAC OTP [HOTP].
<code>#securId-algor</code>	RSA SecurID®.
<code>#securId-aes</code>	New RSA SecurID® AES.

103 `/otp:OtpToken/otp:State`

104 This optional element allows additional state to be communicated. This is useful when the
105 token is an OTP authentication token and a challenge-response scheme is used.

106 `/otp:OtpToken/otp:State/@EncodingType`

107 This optional attribute URI specifies the encoding type of the state.

URI	Description
<code>#string (default)</code>	XML Schema string encoding.
<code>#Base64</code>	XML Schema base 64 encoding.

108 `/otp:OtpToken/otp:Entity`

109 This repeatable element specifies the entity performing authentication.

110 `/otp:OtpToken/otp:Entity/@Type`

111 This optional URI attribute specifies the type of entity being provided. The table below
112 identifies the pre-defined types (note that the URI fragments are relative to the URI for
113 this specification).

URI	Description
<code>#UserName (default)</code>	This entity is a username.
<code>#TokenId</code>	This entity is a Token Identifier.

114 `/otp:OtpToken/otp:Otp`

115 This optional element provides the OTP authentication information.

116 `/otp:OtpToken/otp:Otp/@EncodingType`

117 This optional attribute URI specifies the encoding type of the OTP. See
118 `/otp:OtpToken/otp:State/@EncodingType`.

119 `/otp:OtpToken/otp:Pin`

120 This optional element provides PIN information.

121 `/otp:OtpToken/otp:Pin/@EncodingType`

122 This optional attribute URI specifies the encoding type of the PIN. See
123 `/otp:OtpToken/otp:State/@EncodingType`.

124 `/otp:OtpToken/wsse:Nonce`

125 This optional element specifies a cryptographically random nonce. Each message
126 including a `<wsse:Nonce>` element MUST use a new nonce value in order for web
127 service producers to detect replay attacks.

128 `/otp:otpToken/wsse:Nonce/@EncodingType`

129 This optional attribute URI specifies the encoding type of the nonce. See
130 /otp:OtpToken/otp:State/@EncodingType.

131 /otp:otpToken/wsua:Created

132 The optional <wsua:Created> element specifies a timestamp used to indicate the
133 creation time. It is defined as part of the <wsua:Timestamp> definition.

134 /otp:OtpToken/@{any}

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

137 All compliant implementations MUST be able to process the <otp:otpToken> element. Where
138 the specification requires that an element be "processed" it means that the element type MUST
139 be recognized to the extent that an appropriate error is returned if the element is not supported.

140 The following example illustrates two sample uses of the OTP Token element. Note that in both
141 samples, the OTP Token is sent as clear text and therefore this message should be sent over a
142 confidential channel.

143 Here Alice authenticates with the OATH HOTP algorithm. The optional <otp:State> element is
144 used to carry a counter value.

```
145 <wsse:Security>  
146   <otp:OtpToken Algorithm="...#oath-hotp">  
147     <otp:State>422</otp:State>  
148     <otp:Entity>Alice</otp:Entity>  
149     <otp:Otp>834723</otp:Otp>  
150     <otp:Pin>1234</otp:Pin>  
151   </otp:OtpToken>  
152 </wsse:Security>
```

153 The following example illustrates authenticating with the RSA SecurID AES algorithm:

```
154 <wsse:Security>  
155   <otp:OtpToken Algorithm="...#securid-aes">  
156     <otp:Entity>Alice</otp:Entity>  
157     <otp:Entity Type="...#TokenID">AlicesToken</otp:Entity>  
158     <otp:Otp>748374</otp:Otp>  
159     <otp:Pin>1234</otp:Pin>
```

160
161
162
163

```
<wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>  
<wsu:Created>2003-07-16T01:24:32Z</wsu:Created>  
</otp:OtpToken>  
</wsse:Security>
```

164

4 Token reference

165 When an OtpToken is referenced using `<wsse:SecurityTokenReference>` the `ValueType`
166 attribute is not required. If specified, the value of `#OtpToken` MUST be specified.

167 The following encoding formats are pre-defined (note that the URI fragments are relative to the
168 URI for this specification):

URI	Description
<code>#OtpToken</code>	OTP token.

169 When an OtpToken is referenced from a `<ds:KeyInfo>` element, it can be used to derive a key
170 for a message authentication algorithm using the OTP value. Section "5.2 Key derivation"
171 describes specific mechanisms for key derivation. Implementations should agree on a key
172 derivation algorithm in order to be interoperable.

173 There is no definition of a `KeyIdentifier` for an OtpToken. Consequently, `KeyIdentifier` references
174 MUST NOT be used when referring to an OtpToken.

175 Similarly, there is no definition of a `KeyName` for an OtpToken. Consequently, `KeyName`
176 references MUST NOT be used when referring to an OtpToken.

177 All references refer to the `wsu:Id` for the token.

4.1 Error codes

179 Implementations may use custom error codes defined in private namespaces if needed. But it is
180 RECOMMENDED that they use the error handling codes defined in the WSS: SOAP Message
181 Security specification for signature, decryption, and encoding and token header errors to improve
182 interoperability.

183 When using custom error codes, implementations should be careful not to introduce security
184 vulnerabilities that may assist an attacker in the error codes returned.

185 When an <otp:OtpToken> cannot be successfully validated, but other aspects of SOAP
186 processing have succeeded, a SOAP fault will be returned carrying a Fault/Code/Subcode/Value
187 of wsse:FailedAuthentication.

188

5 Open issues

189 This section is non-normative and serves to provide a point of discussion within the proposal
190 document. The intent is to have an open discussion on the issues, and as they are resolved they
191 will be removed from this section.

192 5.1 Referencing an OTP

193 Some processors may need to identify the target for which a particular generated OTP value is
194 appropriate.

195 There seems to be three ways of accomplishing this, depending on where to execute reference
196 processing.

197 1. **Use the SOAP actor/role.** A single recipient can assume multiple roles to inhabit multiple
198 targets.

199 The following example illustrates such use:

```
200 <soap:Header>  
201   <wsse:Security soap:actor="target1">  
202     <otp:OtpToken>  
203       <otp:Otp>456543</otp:Otp>  
204     </otp:OtpToken>  
205   </wsse:Security>  
206   <wsse:Security soap:actor="target2">  
207     <otp:OtpToken>  
208       <otp:Otp>675765</otp:Otp>  
209     </otp:OtpToken>  
210   </wsse:Security>  
211 </soap:Header>
```

212 This usage seems to follow the SOAP processing model. It is usable if a recipient implements
213 multiple roles.

214 2. **Use security token reference.** OtpToken can be referenced by way of using
215 `<wsse:SecurityTokenReference>` according to [WSS11] Appendix B:
216 SecurityTokenReference Model, Non-Signature References.

217 The following example illustrates such use:

```
218 <soap:Header>
219   <wsse:Security>
220     <otp:OtpToken wsu:Id="target1">
221       <otp:Otp>456543</otp:Otp>
222       ...
223     </otp:OtpToken>
224     <otp:OtpToken wsu:Id="target2">
225       <otp:Otp>675765</otp:Otp>
226       ...
227     </otp:OtpToken>
228   </wsse:Security>
229 </soap:Header>
230 <soap:Body>
231   <MyStuff>
232     ...
233     <wsse:SecurityTokenReference wsse:TokenType="...#OtpToken">
234       <wsse:Reference URI="#target1">
235         </wsse:SecurityTokenReference>
236       </MyStuff>
237 </soap:Body>
```

238 This is useful if the concern is mainly SOAP body processing.

239 **3. Add an element to OtpToken.** An optional element can be added inside <otp:OtpToken>.

240 The following example illustrates such use:

```
241 <wsse:Security>
242   <otp:OtpToken>
243     <otp:Target>target1</otp:Target>
244     ...
245   </otp:OtpToken>
246 </wsse:Security>
```

247 This solution adds an element to the token itself, thus making it straight-forward to realize which
248 recipient is responsible to process which element.

249 **5.2 Key derivation**

250 Both [WSS] and [WSS11] consider key derivation to be out of scope. However, [WSS-UNT11]
251 introduces the concept in "4. Key Derivation."

252 To derive a key from a username token, [WSS-UNT11] defines new optional elements inside
253 `<wsse:UsernameToken>` to handle salt and number hashing iterations. The algorithm is fixed to
254 SHA-1. An added constraint is that the Username Token element MUST NOT contain a
255 password.

256 It is unclear whether this profile needs to define a way of Key derivation.

257

6 Security considerations

258 When using OTP values, care has to be taken on the SOAP processing level to not evaluate the
259 OTP value more than once, since, by definition, the OTP would be accepted only the first time.
260 Appropriate use of the SOAP actor facility can contribute to ensuring once-only processing of
261 OTP values.

262 When an OTP value in an `<otp:OtpToken>` is used for authentication, any `<otp:Otp>` or
263 `<otp:OtpPin>` values need to be properly protected. If the underlying transport does not
264 provide enough protection against eavesdropping, WSS encryption SHOULD be used.

265 The reader should also review Section 13 of WSS: SOAP Message Security document for
266 additional discussion on threats and possible counter-measures.

267 This section is non-normative.

268

7 References

269

The following are normative references:

- 270 **[RFC2119]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels,"
271 RFC 2119, Harvard University, March 1997
- 272 **[OTPS]** J. Linn, "OTP-WSS-Token: Web Services Security One-Time-Password
273 (OTP) Token Profile," Version 1-0d2, 8 April 2005; otps-wst-1-0d2.
- 274 **[HOTP]** D. M'Raihi, et al., "HOTP: An HMAC-based One Time Password
275 Algorithm," <http://www.ietf.org/internet-drafts/draft-mraihi-oath-hmac-otp-04.txt>
276
- 277 **[WSS]** OASIS standard, "WSS: SOAP Message Security," <http://www.oasis-open.org/apps/org/workgroup/wss/download.php/6367/oasis-200401-wss-soap-message-security-1.0.pdf>
278
279
- 280 **[WSS-UNT]** OASIS standard, "Web Services Security UsernameToken Profile 1.0,"
281 [http://www.oasis-](http://www.oasis-open.org/apps/org/workgroup/wss/download.php/6366/oasis-200401-wss-username-token-profile-1.0.pdf)
282 open.org/apps/org/workgroup/wss/download.php/6366/oasis-200401-
283 wss-username-token-profile-1.0.pdf
- 284 **[WSS11]** OASIS working draft 11 May 2005, "Web Services Security: SOAP
285 Message Security 1.1," [http://www.oasis-](http://www.oasis-open.org/apps/org/workgroup/wss/download.php/12635/oasis-2005xx-wss-soap-message-security-1.1-changes.pdf)
286 open.org/apps/org/workgroup/wss/download.php/12635/oasis-2005xx-
287 wss-soap-message-security-1.1-changes.pdf
- 288 **[WSS-UNT11]** OASIS working draft, "Web Services Security UsernameToken Profile
289 1.1,"[http://www.oasis-](http://www.oasis-open.org/apps/org/workgroup/wss/download.php/12636/oasis-2005xx-wss-username-token-profile-1.1-changes.pdf)
290 open.org/apps/org/workgroup/wss/download.php/12636/oasis-2005xx-
291 wss-username-token-profile-1.1-changes.pdf
- 292 **[SOAP11]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
- 293 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers
294 (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox
295 Corporation, August 1998.
- 296 **[XML-Schema]** W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001.
297 W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.
- 298 **[XPath]** W3C Recommendation, "XML Path Language", 16 November 1999

Appendix A. Revision History

Rev	Date	By Whom	What
	2005-05-02	Hans Granqvist	Initial Verisign proposal, adapting selected elements from [OTPS]
	2005-05-16	Hans Granqvist	Incorporating John Linn's comments.
1.0	2005-10-27	Hans Granqvist	Updated for OASIS submission.

300 **Appendix B. Notices**

301 This draft version of the OTP token profile was developed by VeriSign. VeriSign agrees that
302 this contribution is compliant to the terms and conditions that are specified under OASIS.IPR.3.1
303 of the 'Legacy OASIS Intellectual Property Rights (IPR) Policy.

304 RSA, RSA Security and SecurID are registered trademarks or trademarks of RSA Security Inc. in
305 the United States and/or other countries. The names of other products or services mentioned
306 may be the trademarks of their respective owners.

307 RSA Security does not make any claims on the general constructions described in this document.
308 Specific underlying methods and techniques that may be supported and represented using
309 facilities defined in this document may be subject to claims. For example, the RSA SecurID
310 technology implementations of time-based mode authenticator token devices, and related
311 validation processing components, are covered by a number of US patents (and foreign
312 counterparts), in particular US Patent Nos. 4,885,778; 4,856,062; 5,097,505; 5,168,520 and
313 5,657,388. Additional patents are pending. As this specification can be implemented without the
314 use of time-based mode authentication technology, it is RSA Security's position that the
315 technology covered by these patents and applications is not required to implement this
316 specification.