



1

2

Web Services Security UsernameToken Profile 1.0

3

4

Monday, 15 March 2004

5

Document identifier:

6

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

7

Location:

8

<http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0>

9

<http://www.oasis-open.org/committees/documents.php>

10

Editors:

Anthony	Nadalin	IBM	11
Phil	Griffin	Individual	
Chris	Kaler	Microsoft	
Phillip	Hallam-Baker	VeriSign	
Ronald	Monzillo	Sun	

12

Contributors:

Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Lab
Merlin	Hughes	Baltimore Technologies
Irving	Reid	Baltimore Technologies
Peter	Dapkus	BEA
Hal	Lockhart	BEA
Symon	Chang	CommerceOne
Srinivas	Davanum	Computer Associates
Thomas	DeMartini	ContentGuard
Guillermo	Lao	ContentGuard
TJ	Pannu	ContentGuard
Shawn	Sharp	Cyclone Commerce
Ganesh	Vaideeswaran	Documentum
Sam	Wei	Documentum

John	Hughes	Entegrity
Tim	Moses	Entrust
Toshihiro	Nishimura	Fujitsu
Tom	Rutt	Fujitsu
Yutaka	Kudo	Hitachi
Jason	Rouault	HP
Paula	Austel	IBM
Bob	Blakley	IBM
Joel	Farrell	IBM
Satoshi	Hada	IBM
Maryann	Hondo	IBM
Michael	McIntosh	IBM
Hiroshi	Maruyama	IBM
David	Melgar	IBM
Anthony	Nadalin	IBM
Nataraj	Nagaratnam	IBM
Wayne	Vicknair	IBM
Kelvin	Lawrence	IBM (co-Chair)
Don	Flinn	Individual
Bob	Morgan	Individual
Bob	Atkinson	Microsoft
Keith	Ballinger	Microsoft
Allen	Brown	Microsoft
Paul	Cotton	Microsoft
Giovanni	Della-Libera	Microsoft
Vijay	Gajjala	Microsoft
Johannes	Klein	Microsoft
Scott	Konersmann	Microsoft
Chris	Kurt	Microsoft
Brian	LaMacchia	Microsoft
Paul	Leach	Microsoft
John	Manferdelli	Microsoft

John	Shewchuk	Microsoft
Dan	Simon	Microsoft
Hervey	Wilson	Microsoft
Chris	Kaler	Microsoft (co-Chair)
Prateek	Mishra	Netegrity
Frederick	Hirsch	Nokia
Senthil	Sengodan	Nokia
Lloyd	Burch	Novell
Ed	Reed	Novell
Charles	Knouse	Oblix
Steve	Anderson	OpenNetwork (Sec)
Vipin	Samar	Oracle
Jerry	Schwarz	Oracle
Eric	Gravengaard	Reactivity
Stuart	King	Reed Elsevier
Andrew	Nash	RSA Security
Rob	Philpott	RSA Security
Peter	Rostin	RSA Security
Martijn	de Boer	SAP
Blake	Dournaee	Sarvega
Pete	Wenzel	SeeBeyond
Jonathan	Tourzan	Sony
Yassir	Elley	Sun Microsystems
Jeff	Hodges	Sun Microsystems
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet
Michael	Nguyen	The IDA of Singapore
Don	Adams	TIBCO
John	Weiland	US Navy
Phillip	Hallam-Baker	VeriSign
Mark	Hays	Verisign
Hemma	Prafullchandra	VeriSign

13

14 **Abstract:**

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

17 **Status:**

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

20 If you are on the wss@lists.oasis-open.org list for committee members, send comments
21 there. If you are not on that list, subscribe to the wss-comment@lists.oasis-open.org list
22 and send comments there. To subscribe, send an email message to [wss-comment-](mailto:wss-comment-request@lists.oasis-open.org)
23 [request@lists.oasis-open.org](mailto:wss-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

24 For patent disclosure information that may be essential to the implementation of this
25 specification, and any offers of licensing terms, refer to the Intellectual Property Rights
26 section of the OASIS Web Services Security Technical Committee (WSS TC) web page
27 at <http://www.oasis-open.org/committees/wss/ipr.php>. General OASIS IPR information
28 can be found at <http://www.oasis-open.org/who/intellectualproperty.shtml>.

29 **Table of Contents**

30	1 Introduction	6
31	2 Notations and Terminology.....	6
32	2.1 Notational Conventions	6
33	2.2 Namespaces	6
34	2.3 Acronyms and Abbreviations	7
35	3 UsernameToken Extensions	7
36	3.1 Usernames and Passwords	7
37	3.2 Token Reference.....	10
38	3.3 Error Codes	11
39	4 Security Considerations.....	11
40	5 References	12
41	Appendix A. Revision History	13
42	Appendix B. Notices	14
43		

44 1 Introduction

45 This document describes how to use the UsernameToken with the WSS: SOAP Message
46 Security specification [WSS]. More specifically, it describes how a web service consumer can
47 supply a UsernameToken as a means of identifying the requestor by "username", and optionally
48 using a password (or shared secret, or password equivalent) to authenticate that identity to the
49 web service producer.

50 This section is non-normative.

51 2 Notations and Terminology

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

53 2.1 Notational Conventions

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

57 When describing abstract data models, this specification uses the notational convention used by
58 the XML Infoset. Specifically, abstract property names always appear in square brackets (e.g.,
59 [some property]).

60 When describing concrete XML schemas [XML-Schema], this specification uses the notational
61 convention of WSS: SOAP Message Security. Specifically, each member of an element's
62 [children] or [attributes] property is described using an XPath-like [XPath] notation (e.g.,
63 /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element
64 wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard
65 (<xs:anyAttribute/>).

66 Commonly used security terms are defined in the Internet Security Glossary [SECGLO]. Readers
67 are presumed to be familiar with the terms in this glossary as well as the definition in the Web
68 Services Security specification.

69 2.2 Namespaces

70 Namespace URIs (of the general form "some-URI") represents some application-dependent or
71 context-dependent URI as defined in RFC 2396 [URI]. This specification is designed to work with
72 the general SOAP [SOAP11, SOAP12] message structure and message processing model, and
73 should be applicable to any version of SOAP. The current SOAP 1.1 namespace URI is used
74 herein to provide detailed examples, but there is no intention to limit the applicability of this
75 specification to a single version of SOAP.

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

79

Prefix	Namespace
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope

wsse	http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd

80 The URLs provided for the *wsse* and *wsu* namespaces can be used to obtain the schema files.

81 2.3 Acronyms and Abbreviations

82 The following (non-normative) table defines acronyms and abbreviations for this document.

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

83 3 UsernameToken Extensions

84 3.1 Usernames and Passwords

85 The `<wsse:UsernameToken>` element is introduced in the WSS: SOAP Message Security
86 documents as a way of providing a username.

87 Within `<wsse:UsernameToken>` element, a `<wsse>Password>` element may be specified.
88 Passwords of type `wsse:PasswordText` and `wsse>PasswordDigest` are not limited to
89 actual passwords, although this is a common case. Any password equivalent such as a derived
90 password or S/KEY (one time password) can be used. Having a type of `wsse:PasswordText`,
91 `wsse>PasswordDigest` merely implies that the information held in the password is “in the
92 clear”, as opposed to holding a “digest” of the information. For example, if a server does not have
93 access to the clear text of a password but does have the hash, then the hash is considered a
94 *password equivalent* and can be used anywhere where a “password” is indicated in this
95 specification. It is not the intention of this specification to require that all implementations have
96 access to clear text passwords.

97 Passwords of type `wsse:PasswordText` and `wsse>PasswordDigest` are defined as being
98 the Base64 [XML-Schema] encoded, SHA-1 hash value, of the UTF8 encoded password (or
99 equivalent). However, unless this digested password is sent on a secured channel or the token is
100 encrypted, the digest offers no real additional security over use of `wsse:PasswordText` and
101 `wsse>PasswordDigest`.

102 Two optional elements are introduced in the `<wsse:UsernameToken>` element to provide a
103 countermeasure for replay attacks: `<wsse:Nonce>` and `<wsu:Created>`. A nonce is a
104 random value that the sender creates to include in each UsernameToken that it sends. Although
105 using a nonce is an effective countermeasure against replay attacks, it requires a server to
106 maintain a cache of used nonces, consuming server resources. Combining a nonce with a

107 creation timestamp has the advantage of allowing a server to limit the cache of nonces to a
108 "freshness" time period, establishing an upper bound on resource requirements. If either or both
109 of <wsse:Nonce> and <wsu:Created> are present they MUST be included in the digest value
110 as follows:

111

112 Password_Digest = Base64 (SHA-1 (nonce + created + password))

113

114 That is, concatenate the nonce, creation timestamp, and the password (or shared secret or
115 password equivalent), digest the combination using the SHA-1 hash algorithm, then include the
116 Base64 encoding of that result as the password (digest). This helps obscure the password and
117 offers a basis for preventing replay attacks. For web service producers to effectively thwart replay
118 attacks, three counter measures are RECOMMENDED:

- 119 1. It is RECOMMENDED that web service producers reject any UsernameToken *not*
120 using *both* nonce *and* creation timestamps.
- 121 2. It is RECOMMENDED that web service producers provide a timestamp "freshness"
122 limitation, and that any UsernameToken with "stale" timestamps be rejected. As a
123 guideline, a value of five minutes can be used as a minimum to detect, and thus
124 reject, replays.
- 125 3. It is RECOMMENDED that used nonces be cached for a period at least as long as
126 the timestamp freshness limitation period, above, and that UsernameToken with
127 nonces that have already been used (and are thus in the cache) be rejected.

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

131 Note that `wsse:PasswordDigest` can only be used if the plain text password (or password
132 equivalent) is available to both the requestor and the recipient.

133 Note that the secret is put at the end of the input and not the front. This is because the output of
134 SHA-1 is the function's complete state at the end of processing an input stream. If the input
135 stream happened to fit neatly into the block size of the hash function, an attacker could extend
136 the input with additional blocks and generate new/unique hash values knowing only the hash
137 output for the original stream. If the secret is at the end of the stream, then attackers are
138 prevented from arbitrarily extending it -- since they have to end the input stream with the
139 password which they don't know. Similarly, if the nonce/created was put at the end, then an
140 attacker could update the nonce to be nonce+created, and add a new created time on the end to
141 generate a new hash.

142 The countermeasures above do not cover the case where the token is replayed to a different
143 receiver. There are several (non-normative) possible approaches to counter this threat, which
144 may be used separately or in combination. Their use requires pre-arrangement (possibly in the
145 form of a separately published profile which introduces new password type) among the
146 communicating parties to provide interoperability:

- 147 • including the username in the hash, to thwart cases where multiple user accounts
148 have matching passwords (e.g. passwords based on company name)
- 149 • including the domain name in the hash, to thwart cases where the same
150 username/password is used in multiple systems
- 151 • including some indication of the intended receiver in the hash, to thwart cases where
152 receiving systems don't share nonce caches (e.g., two separate application clusters
153 in the same security domain).

154 The following illustrates the XML syntax of this element:

155

156
157
158
159
160
161

```
<wsse:UsernameToken wsu:Id="Example-1">  
  <wsse:Username> ... </wsse:Username>  
  <wsse:Password Type="..."> ... </wsse:Password>  
  <wsse:Nonce EncodingType="..."> ... </wsse:Nonce>  
  <wsu:Created> ... </wsu:Created>  
</wsse:UsernameToken>
```

162

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

164 /wsse:UsernameToken/wsse:Password

165 This optional element provides password information (or equivalent such as a hash). It is
166 RECOMMENDED that this element only be passed when a secure transport (e.g.
167 HTTP/S) is being used or if the token itself is being encrypted.

168 /wsse:UsernameToken/wsse:Password/@Type

169 This optional URI attribute specifies the type of password being provided. The table
170 below identifies the pre-defined types (note that the URI fragments are relative to the URI
171 for this specification).

172

URI	Description
#PasswordText (default)	The actual password for the username, the password hash, or derived password or S/KEY. This type should be used when hashed password equivalents that do not rely on a nonce or creation time are used, or when a digest algorithm other than SHA1 is used.
#PasswordDigest	The digest of the password (and optionally nonce and/or creation timestamp) for the username using the algorithm described above.

173

174 /wsse:UsernameToken/wsse:Password/@{any}

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

177 /wsse:UsernameToken/wsse:Nonce

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

181 /wsse:UsernameToken/wsse:Nonce/@EncodingType

182 This optional attribute URI specifies the encoding type of the nonce (see the definition of
183 <wsse:BinarySecurityToken> for valid values). If this attribute isn't specified then
184 the default of Base64 encoding is used.

185 /wsse:UsernameToken/wsdu:Created

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

188 All compliant implementations MUST be able to process the <wsse:UsernameToken> element.
189 Where the specification requires that an element be "processed" it means that the element type
190 MUST be recognized to the extent that an appropriate error is returned if the element is not
191 supported.

192 Note that <wsse:KeyIdentifier> and <ds:KeyName> elements as described in the WSS:
 193 SOAP Message Security specification are not supported in this profile.
 194 The following example illustrates the use of this element. In this example the password is sent as
 195 clear text and therefore this message should be sent over a confidential channel:

196

```

197 <S11:Envelope xmlns:S11="..." xmlns:wsse="...">
198   <S11:Header>
199     ...
200     <wsse:Security>
201       <wsse:UsernameToken>
202         <wsse:Username>Zoe</wsse:Username>
203         <wsse:Password>IloveDogs</wsse:Password>
204       </wsse:UsernameToken>
205     </wsse:Security>
206     ...
207   </S11:Header>
208   ...
209 </S11:Envelope>
  
```

210

211 The following example illustrates using a digest of the password along with a nonce and a
 212 creation timestamp:

213

```

214 <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="...">
215   <S11:Header>
216     ...
217     <wsse:Security>
218       <wsse:UsernameToken>
219         <wsse:Username>NNK</wsse:Username>
220         <wsse:Password Type="...#PasswordDigest">
221           weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==
222         </wsse:Password>
223         <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
224         <wsu:Created>2003-07-16T01:24:32Z</wsu:Created>
225       </wsse:UsernameToken>
226     </wsse:Security>
227     ...
228   </S11:Header>
229   ...
230 </S11:Envelope>
  
```

231

232 3.2 Token Reference

233 When a UsernameToken is referenced using <wsse:SecurityTokenReference> the
 234 ValueType attribute is not required. If specified, the value of <wsse:UsernameToken> MUST
 235 be specified.

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

238

URI	Description
#UsernameToken	UsernameToken

239

240

241 When a UsernameToken is referenced from a `<ds:KeyInfo>` element, it can be used to derive
242 a key for a message authentication algorithm using the password. This profile considers specific
243 mechanisms for key derivation to be out of scope. Implementations should agree on a key
244 derivation algorithm in order to be interoperable.

245 There is no definition of a KeyIdentifier for a UsernameToken. Consequently, KeyIdentifier
246 references MUST NOT be used when referring to a UsernameToken.

247 Similarly, there is no definition of a KeyName for a UsernameToken. Consequently, KeyName
248 references MUST NOT be used when referring to a UsernameToken.

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

250 **3.3 Error Codes**

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

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

257 **4 Security Considerations**

258 The use of the UsernameToken introduces no additional threats beyond those already identified
259 for other types of SecurityTokens. Replay attacks can be addressed by using message
260 timestamps, nonces, and caching, as well as other application-specific tracking mechanisms.
261 Token ownership is verified by use of keys and man-in-the-middle attacks are generally
262 mitigated. Transport-level security may be used to provide confidentiality and integrity of both the
263 UsernameToken and the entire message body.

264 When a password (or password equivalent) in a `<UsernameToken>` is used for authentication,
265 the password needs to be properly protected. If the underlying transport does not provide enough
266 protection against eavesdropping, the password SHOULD be digested as described in this
267 document. Even so, the password must be strong enough so that simple password guessing
268 attacks will not reveal the secret from a captured message.

269 When a password is encrypted, in addition to the normal threats against any encryption, two
270 password-specific threats must be considered: replay and guessing. If an attacker can
271 impersonate a user by replaying an encrypted or hashed password, then learning the actual
272 password is not necessary. One method of preventing replay is to use a nonce as mentioned
273 previously. Generally it is also necessary to use a timestamp to put a ceiling on the number of
274 previous nonces that must be stored. However, in order to be effective the nonce and timestamp
275 must be signed. If the signature is also over the password itself, prior to encryption, then it would
276 be a simple matter to use the signature to perform an offline guessing attack against the
277 password. This threat can be countered in any of several ways including: don't include the
278 password under the signature (the password will be verified later) or sign the encrypted
279 password.

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

282 This section is non-normative.

283

5 References

284

The following are normative references:

285

[SECGLO]

Informational RFC 2828, "Internet Security Glossary," May 2000.

286

[RFC2119]

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

287

288

[WSS]

OASIS standard, "WSS: SOAP Message Security," TBD.

289

[SOAP11]

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

290

[SOAP12]

W3C Working Draft, "SOAP Version 1.2 Part 1: Messaging Framework", 26 June 2002.

291

292

[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.

293

294

295

[XML-Schema]

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

296

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

297

[XPath]

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

298

The following are non-normative references included for background and related material:

299

[WS-Security]

OASIS, "Web Services Security: SOAP Message Security" 19 January 2004, <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0>

300

301

302

[XML-C14N]

W3C Recommendation, "Canonical XML Version 1.0," 15 March 2001

303

[EXC-C14N]

W3C Recommendation, "Exclusive XML Canonicalization Version 1.0," 8 July 2002.

304

305

[XML-Encrypt]

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

306

307

W3C Recommendation, "Decryption Transform for XML Signature", 10 December 2002.

308

309

[XML-ns]

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

310

[XML Signature]

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

311

312

[XPointer]

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

313

314

Appendix A. 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
Wd-1.3	2003-06-30	Anthony Nadalin	Editorial Updates
Wd-1.4	2003-08-11	Anthony Nadalin	Editorial Updates
Cd-1.5	2003-12-09	Anthony Nadalin, Chris Kaler	Editorial Updates based on Issue List #30
Cd-1.5	2003-12-15	Anthony Nadalin, Chris Kaler	Editorial Updates based on Editorial feedback
Cd-1.6	2003-12-22	Anthony Nadalin	Editorial Updates based on Editorial feedback
Cd-1.7 & 1.8	2003-12-29	Anthony Nadalin, Chris Kaler	Editorial Updates based on Editorial feedback
Cd- 1.8	2004-01-19	Anthony Nadalin, Chris Kaler	Editorial corrections for name space and document name
Cd 1.9	2004-02-17	Anthony Nadalin	Editorial corrections per Karl Best

Appendix B. Notices

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

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

329 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
330 2002-2004. All Rights Reserved.

331 This document and translations of it may be copied and furnished to others, and derivative works
332 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
333 published and distributed, in whole or in part, without restriction of any kind, provided that the
334 above copyright notice and this paragraph are included on all such copies and derivative works.
335 However, this document itself does not be modified in any way, such as by removing the
336 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
337 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
338 Property Rights document must be followed, or as required to translate it into languages other
339 than English.

340 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
341 successors or assigns.

342 This document and the information contained herein is provided on an "AS IS" basis and OASIS
343 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
344 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
345 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
346 PARTICULAR PURPOSE.