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

4.1 Usernames and Passwords

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

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

Passwords of type wsse:PasswordDigest are defined as being the Base64 [16] encoded, SHA-1 hash value, of the UTF8 [17] encoded password (or equivalent).. However, unless this digested password is sent on a secured channel, the digest offers no real additional security over use of `wsse:PasswordText`.

To address this issue, two optional elements are introduced in the `<wsse:UsernameToken>` element: `<wsse:Nonce>` and `<wsu:Created>`. If either or both of these are present, they must be included in the digest value as follows:

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

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

1. First, it is recommended that web service providers reject any UsernameToken *not* using *both* nonce *and* creation timestamps.
2. Second, it is recommended that web service producers provide a timestamp “freshness” limitation, and that any UsernameToken with “stale” timestamps be

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

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

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

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

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

```
<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>
```

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

/wsse:UsernameToken/Password

This optional element provides password information (or equivalent such as a hash). It is recommended that this element only be passed when a secure transport is being used.

/wsse:UsernameToken/Password/@Type

This optional attribute specifies the type of password being provided. The following table identifies the pre-defined types:

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.

/wsse:UsernameToken/Password/@{any}

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

/wsse:UsernameToken/wsse:Nonce

This optional element specifies a cryptographically random nonce. Each message including a Nonce element should use a new nonce value in order for web service providers to detect replay attacks

/wsse:UsernameToken/wsse:Nonce/@EncodingType

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

/wsse:UsernameToken/wsu:Created

This optional element which specifies a timestamp. The element is used to indicate the creation time.

All compliant implementations must be able to process the `<wsse:UsernameToken>` element. The following example illustrates the use of this element. In this example the password is sent as clear text and therefore this message should be sent over a confidential channel:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
   xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext">
   <S:Header>
      ...
      <wsse:Security>
         <wsse:UsernameToken >
            <wsse:Username> Zoe </wsse:Username>
            <wsse:Password> ILoveDogs </wsse:Password>
         </wsse:UsernameToken>
      </wsse:Security>
      ...
   </S:Header>
   ...
</S:Envelope>
```

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

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
   xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext">
   <S:Header>
      ...
      <wsse:Security>
         <wsse:UsernameToken
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext"
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/xx/utility">
            <wsse:Username> NNK </wsse:Username>
            <wsse:Password Type="wsse:PasswordDigest">
               D2A12DFE8D9F0C6BB82C89B091DF5C8A872F94DC
            </wsse:Password>
            <wsse:Nonce> EFD89F06CCB28C89 </wsse:Nonce>
            <wsu:Created> 2001-10-13T09:00:00Z </wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
      ...
   </S:Header>
   ...
</S:Envelope>
```

4.2 Error Codes

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

4.3 Threat Model

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

5 References

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

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

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

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

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

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

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

Appendix A. Acknowledgments

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

- TBD

239 # 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

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's procedures with respect to rights in OASIS specifications can be found at 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 implementors or users of this specification, can be obtained from the OASIS Executive Director.

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

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

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 paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document 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 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.