2002
PC Magazine Award
for Technical Excellence

FINALIST
OASIS WS-Security
OASIS

Web Services Security UsernameToken Profile

Working Draft 4, Monday, 11 August 2003

Document identifier:

{draft}-{*WSS: SOAP Message Security* }-{UsernameToken Profile }-{*4.0*} (Word) (PDF)

Location:

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

Editors:

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

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
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
Bob	Blakley	IBM
Joel	Farrell	IBM
Satoshi	Hada	IBM
Maryann	Hondo	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	Konermann	Microsoft
Chris	Kurt	Microsoft

Brian	LaMacchia	Microsoft
Paul	Leach	Microsoft
John	Manferdell	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
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

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 Notations and Terminology

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

2,1 Notational Conventions

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

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

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

This specification is designed to work with the general SOAP message structure and message processing model, and should be applicable to any version of SOAP. The current SOAP 1.2 namespace URI is used herein to provide detailed examples, but there is no intention to limit the applicability of this specification to a single version of SOAP.

Readers are presumed to be familiar with the terms in the Internet Security Glossary.

3 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/2003/06/secext
wsu	http://schemas.xmlsoap.org/ws/2003/06/utility

4 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

3 UsernameToken Extensions

Token Types

This profile defines the syntax of, and processing rules for the Username Token:

Token	QName	Description
Username	wsse:UsernameToken	A Username Token

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.

Two optional elements are introduced in the <wsse:UsernameToken> element to provide a countermeasure for replay attacks: <wsse:Nonce> and <wsu:Created>. A nonce is a random value that the sender creates to include in each Username token that it sends. Although using a nonce is an effective countermeasure against replay attacks, it requires a server to maintain a cache of used nonces, consuming server resources. Combining a nonce with a creation timestamp has the advantage of allowing a server to limit the cache of nonces to a "freshness" time period, establishing a bound on resource requirements. If either or both of <wsse:Nonce> and <wsu:Created> 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 hash 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 <wsu:Created> element specifies a timestamp used to indicate the creation time. It is defined as part of the <wsu:Timestamp> definition.

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/2003/06/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/2003/06/secext">
   <S:Header>
      ...
      <wsse:Security>
         <wsse:UsernameToken
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
            <wsse:Username>NNK</wsse:Username>
            <wsse:Password Type="wsse:PasswordDigest">
               weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==
            </wsse:Password>
            <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
            <wsu:Created>2003-07-16T01:24:32Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
      ...
   </S:Header>
   ...
</S:Envelope>
```

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.

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.

4 References

[DIGSIG]	Informational RFC 2828, "Internet Security Glossary," May 2000.
[KEYWORDS]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997
[SOAP11]	W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
[SOAP12]	W3C Working Draft, "SOAP Version 1.2 Part 1: Messaging Framework", 26 June 2002.
[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.
[WS-Security]	"Web Services Security Language", IBM, Microsoft, VeriSign, April 2002. "WS-Security Addendum", IBM, Microsoft, VeriSign, August 2002. "WS-Security XML Tokens", IBM, Microsoft, VeriSign, August 2002.
[XML-C14N]	W3C Recommendation, "Canonical XML Version 1.0," 15 March 2001
[EXC-C14N]	W3C Recommendation, "Exclusive XML Canonicalization Version 1.0," 8 July 2002.
[XML-Encrypt]	W3C Working Draft, "XML Encryption Syntax and Processing," 04 March 2002 W3C Recommendation, "Decryption Transform for XML Signature", 10 December 2002.
[XML-ns]	W3C Recommendation, "Namespaces in XML," 14 January 1999.
[XML-Schema]	W3C Recommendation, "XML Schema Part 1: Structures,"2 May 2001. W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.
[XML Signature]	W3C Recommendation, "XML Signature Syntax and Processing," 12 February 2002.
[XPath]	W3C Recommendation, "XML Path Language", 16 November 1999
[XPointer]	"XML Pointer Language (XPointer) Version 1.0, Candidate Recommendation", DeRose, Maler, Daniel, 11 September 2001.

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

Appendix B. 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.