[MS-MWBF]:
Microsoft Web Browser Federated Sign-On Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Copyright © 2013 Microsoft Corporation.

Revision Summary

Date	Revision History	Revision Class	Comments
10/22/2006	0.01		MCPP Milestone 1 Initial Availability
01/19/2007	1.0		MCPP Milestone 1
03/02/2007	1.1		Monthly release
04/03/2007	1.2		Monthly release
05/11/2007	1.3		Monthly release
06/01/2007	1.3.1	Editorial	Revised and edited the technical content.
07/03/2007	1.3.2	Editorial	Revised and edited the technical content.
07/20/2007	1.3.3	Editorial	Revised and edited the technical content.
08/10/2007	1.4	Minor	Updated the technical content.
09/28/2007	1.4.1	Editorial	Revised and edited the technical content.
10/23/2007	1.5	Minor	Updated the technical content.
11/30/2007	1.6	Minor	Updated the technical content.
01/25/2008	1.6.1	Editorial	Revised and edited the technical content.
03/14/2008	1.6.2	Editorial	Revised and edited the technical content.
05/16/2008	1.6.3	Editorial	Revised and edited the technical content.
06/20/2008	2.0	Major	Content changes for Release codenamed "Geneva".
07/25/2008	2.0.1	Editorial	Revised and edited the technical content.
08/29/2008	3.0	Major	Removed "Geneva" content.
10/24/2008	4.0	Major	Updated and revised the technical content.
12/05/2008	4.0.1	Editorial	Revised and edited the technical content.
01/16/2009	4.0.2	Editorial	Revised and edited the technical content.
02/27/2009	4.0.3	Editorial	Revised and edited the technical content.
04/10/2009	4.1	Minor	Updated the technical content.
05/22/2009	4.1.1	Editorial	Revised and edited the technical content.
07/02/2009	5.0	Major	Updated and revised the technical content.
08/14/2009	6.0	Major	Updated and revised the technical content.

Copyright © 2013 Microsoft Corporation.

Date	Revision History	Revision Class	Comments
09/25/2009	6.1	Minor	Updated the technical content.
11/06/2009	6.1.1	Editorial	Revised and edited the technical content.
12/18/2009	6.1.2	Editorial	Revised and edited the technical content.
01/29/2010	6.2	Minor	Updated the technical content.
03/12/2010	6.2.1	Editorial	Revised and edited the technical content.
04/23/2010	6.2.2	Editorial	Revised and edited the technical content.
06/04/2010	6.2.3	Editorial	Revised and edited the technical content.
07/16/2010	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	6.2.3	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	6.3	Minor	Clarified the meaning of the technical content.
09/23/2011	6.3	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	7.0	Major	Significantly changed the technical content.
03/30/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2013 Microsoft Corporation.

Date	Revision History	Revision Class	Comments
01/31/2013	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	8.0	Major	Significantly changed the technical content.
11/14/2013	9.0	Major	Significantly changed the technical content.

Copyright © 2013 Microsoft Corporation.

Contents

Copyright © 2013 Microsoft Corporation.

Copyright © 2013 Microsoft Corporation.

Copyright © 2013 Microsoft Corporation.

1 Introduction

The Microsoft Web Browser Federated Sign-On Protocol is primarily a restriction of the protocol specified in [WSFedPRP]. The restrictions are designed to enable greater interoperability by reducing the number of variations that must be implemented. This document specifies minor additions to [WSFedPRP] to handle common scenarios. This protocol is designed to enable the communication of a requestor's identity and attributes for the purpose of enabling access to a protected HTTP web application or its resources.

This protocol is based on the Web Service (WS) Federation Protocol described in [WSFederation] and [WSFedPRP].

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

domain naming service name

The following terms are specific to this document:

claim: A declaration made by an entity (for example, name, identity, key, group, privilege, and capability). For more information, see [WSFedPRP] sections 1.4 and 2.

digest: A cryptographic checksum of a data (octet) stream.

federation: A collection of **security realms** that have established **trust**.

identity provider (IP): A **security token service (STS)** that performs identity verification as part of its processing. For more information, see [WSFedPRP].

identity provider/security token service (IP/STS): An **STS** that may or may not be an **identity provider (IP)**. This term is used as shorthand to see both identity that verifies token services and general token services that do not verify identity. Note that the "/" symbol implies an "or" relationship.

relying party: A web application or service that consumes **security tokens** issued by an **STS**.

requestor IP/STS: An **IP/STS** in the same **security realms** as the **web browser requestor**. The **requestor IP/STS** has an existing relationship with the **user** that enables it to issue **security tokens** containing **user** information.

resource IP/STS: An **IP/STS** in the same **security realm** as the **web service (WS) resource**. The **resource IP/STS** has an existing relationship with the **WS resource** that enables it to issue **security tokens** that are trusted by the **WS resource**.

security realm or security domain: Represents a single unit of security administration or **trust** (for example, a Kerberos realm, for more information, see [RFC4120]; or a Windows Domain, for more information, see [MSFT-ADC]).

security token: Represents a collection of one or more **claims**.

Copyright © 2013 Microsoft Corporation.

security token service (STS): A web service that issues **security tokens**. That is, it makes assertions based on evidence that it **trusts** for consumption by whoever trusts it. For more information, see [WSFedPRP] sections 1.4 and 2. For this protocol, **STS** refers to services that support (either directly or via a front-end) the HTTP protocol defined in this specification.

signature: A value computed with a cryptographic algorithm (often involving a **digest**) and bound to data in such a way that intended recipients of the data can use the **signature** to verify that the data has not been altered since it was signed by the signer. For more information, see [WSFedPRP] sections 1.4 and 2.

sign-out: The process by which a **user** (or an agent acting on the user's behalf) indicates that it will no longer be using its **security token**, and **relying parties** across **security realms** can destroy their **security token** caches for the **user**. For more information, see [WSFedPRP] sections 1.4 and 2. Note that the use of the term **sign-out** is based on [WSFedPRP].

single sign on (SSO): An optimization of **user** authentications to remove the burden of repeating actions placed on the end **user** (for example, prompting for user names and passwords multiple times). To facilitate **SSO**, an **IP/STS** can provide evidence of authentication events and **user** account information to third parties requesting information about the requestor (subject to policy and authorization restrictions). For more information, see [WSFedPRP] sections 1.4 and 2. Note that the use of the term **sign on** is based on [WSFedPRP].

subject key identifier (SKI): The **SKI** extension provides a means of identifying certificates that contain a particular public key. For more information, see [RFC3280] section 4.2.1.2.

trust: The characteristic that one entity is willing to rely on a second entity to execute a set of actions and/or to make a set of assertions about a set of subjects and/or scopes. For more information, see [WSFedPRP] sections 1.4 and 2.

user: A person who employs a **web browser requestor** to access a **WS resource**.

user principal name (UPN): In the context of this specification, a **UPN** contains a **user** account name (sometimes referred to as the user logon name) and a **DNS name** that together uniquely identify the **user**. The format is in the form of an email address; for example, someone@example.com.

web browser requestor: An HTTP 1.1 web browser client that transmits protocol messages between an **IP/STS** and a **relying party**.

web service (WS) resource: A destination HTTP 1.1 web application or an HTTP 1.1 resource serviced by the application. In the context of this protocol, it refers to the application or manager of the resource that receives identity information and assertions issued by an **IP/STS** using this protocol. The **WS resource** is a **relying party** in the context of this protocol. For more information, see [WSFedPRP] sections 1.4 and 2.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as specified in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

Copyright © 2013 Microsoft Corporation.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [Windows Protocol].

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[Excl-C14N] Boyer, J., Eastlake 3rd, D. E., and Reagle, J., "Exclusive XML Canonicalization Version 1.0", July 2002, http://www.w3.org/TR/xml-exc-c14n/

[HTML] World Wide Web Consortium, "HTML 4.01 Specification", December 1999, http://www.w3.org/TR/html4/

[MS-ADA1] Microsoft Corporation, "Active Directory Schema Attributes A-L".

[MS-ADA3] Microsoft Corporation, "Active Directory Schema Attributes N-Z".

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-MWBE] Microsoft Corporation, "Microsoft Web Browser Federated Sign-On Protocol Extensions".

[RFC1738] Berners-Lee, T., Masinter, L., and McCahill, M., "Uniform Resource Locators (URL)", RFC 1738, December 1994, http://www.ietf.org/rfc/rfc1738.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2822] Resnick, P., Ed., "Internet Message Format", STD 11, RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.ietf.org/rfc/rfc4648.txt

[SAMLASchema] OASIS Standard, "Security Assertion Markup Language (SAML) V1.1 XML Schema", September 2003, http://www.oasis-open.org/committees/download.php/3408/oasis-sstc-saml-schema-assertion-1.1.xsd

[SAMLCore] Maler, E., Mishra, P., Philpott, R., et al., "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1", September 2003, http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf

Copyright © 2013 Microsoft Corporation.

[WSAddressing] Box, D., Christensen, E., Ferguson, D., et al., "Web Services Addressing (WS-Addressing)", August 2004, http://www.w3.org/Submission/ws-addressing/

If you have any trouble finding [WSAddressing], please check here.

[WSFederation] Kaler, C., Nadalin, A., Bajaj, S., et al., "Web Services Federation Language (WS-Federation)", Version 1.1, December 2006, http://specs.xmlsoap.org/ws/2006/12/federation/ws-federation.pdf

If you have any trouble finding [WSFederation], please check here.

[WSFedPRP] IBM, BEA Systems, Microsoft, VeriSign, RSA Security, "WS-Federation: Passive Requestor Profile", version 1.0, July 2003, http://msdn.microsoft.com/en-us/library/bb608217.aspx

If you have any trouble finding [WSFedPRP], please check here.

[WSPolicyAtt] BEA Systems, IBM, Microsoft Corporation, SAP, Sonic Software, VeriSign, "Web Services Policy 1.2 - Attachment (WS-PolicyAttachment)", April 2006, http://www.w3.org/Submission/WS-PolicyAttachment/

[WSTrust] IBM, Microsoft, Nortel, VeriSign, "WS-Trust V1.0", February 2005, http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf

[X500] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Services", Recommendation X.500, August 2005, http://www.itu.int/rec/T-REC-X.500-200508-S/en

Note There is a charge to download the specification.

[X509] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks", Recommendation X.509, August 2005, http://www.itu.int/rec/T-REC-X.509/en

Note There is a charge to download the specification.

[XMLDSig] Bartel, M., Boyer, J., Fox, B., LaMacchia, B., and Simon, E., "XML-Signature Syntax and Processing", February 2002, http://www.w3.org/TR/xmldsig-core/

[XMLEnc] Imamura, T., Dillaway, B., and Simon, E., "XML Encryption Syntax and Processing", W3C Recommendation, December 2002, http://www.w3.org/TR/xmlenc-core/

1.2.2 Informative References

[FIPS180] FIPS PUBS, "Secure Hash Standard", FIPS PUB 180-1, April 1995, http://www.itl.nist.gov/fipspubs/fip180-1.htm

[IDFF] Liberty Alliance Project, "Liberty ID-FF Protocols and Schema Specification, Version: 1.2-errata-v2.0", 2004, http://www.projectliberty.org/liberty/content/download/1992/13890/file/draft-liberty-idff-protocols-schema-1.2-errata-v2.0.pdf

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MS-PASS] Microsoft Corporation, "Passport Server Side Include (SSI) Version 1.4 Protocol".

[MSFT-ADC] Microsoft Corporation, "Active Directory Collection", March 2003, http://technet2.microsoft.com/WindowsServer/en/library/6f8a7c80-45fc-4916-80d9-16e6d46241f91033.mspx

Copyright © 2013 Microsoft Corporation.

[RFC2246] Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt

[RFC2965] Kristol, D., and Montulli, L., "HTTP State Management Mechanism", RFC 2965, October 2000, http://www.ietf.org/rfc/rfc2965.txt

[RFC3447] Jonsson, J., and Kaliski, B., "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003, http://www.ietf.org/rfc/rfc3447.txt

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., et al., "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003, http://www.ietf.org/rfc/rfc3546.txt

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.txt

[SAML20Prof] Hughes, J., Cantor, S., Hodges, J., et al., "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[XML] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", W3C Recommendation, August 2006, http://www.w3.org/TR/2006/REC-xml-20060816/

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H.S., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., and Malhotra, A., Eds., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.3 Overview

Specification [WSFedPRP] is designed to address two problems related to communicating **user** information to **web service (WS) resources**.

First, in order to properly control access to information in WS resources, those WS resources must have information about the users who are accessing them. Previous solutions required the application to identify the user and to use that identification information to access information about the user. Second, users were forced to be prompted multiple times for user names and passwords to securely identify themselves across multiple WS resources.

[WSFedPRP] addresses these two problems by enabling the user to securely communicate account information across security realms to multiple WS resources without requiring multiple prompts for user names and passwords.

[WSFedPRP] solves these problems by moving the responsibility for authenticating the user away from the WS resource application to an **identity provider/security token service (IP/STS)** that already has an account for the user. This IP/STS issues **security tokens** that contain information about the user. When accessing a WS resource, the user's **web browser requestor** presents a security token obtained from an IP/STS to the WS resource application. The **signature** of the security token allows the WS resource to verify its validity, and the content of the security token contains **claims** about the authentication with the IP/STS as well as the relevant user account information for the WS resource. These claims can then be used for authorization decisions by the

Copyright © 2013 Microsoft Corporation.

WS resource. A sequence diagram describing the detailed exchange can be found in [WSFedPRP] section 4.

The Microsoft Web Browser Federated Sign-On Protocol increases interoperability of [WSFedPRP] by restricting the protocol options and the variations of security tokens that may be included in [WSFedPRP]. The following list outlines restrictions to [WSFedPRP]:

- The HTTP verbs are restricted for different message types.
- The allowable message types are scoped down to message types directly related to sign-on or **sign-out** operations, as specified in [WSFedPRP] sections 2.1 and 2.2.
- The parameters specified in [WSFedPRP] are restricted for each message type.

This protocol also adds optional parameters to address existing limitations to the protocol.

In section 2.1, this document specifies restrictions on the choice of message transport allowed in [WSFedPRP]. The [WSFedPRP] specification allows wsignin1.0, wsignout1.0, and wsignoutcleanup1.0 operations to be transmitted using either GET or POST methods, as specified in [RFC2616]. This specification restricts wsignin1.0, wsignout1.0, and wsignoutcleanup1.0 requests to use only the GET method. This protocol also restricts wsignin1.0 responses to be transmitted to **relying parties** using the POST method.

Parameter removals, restrictions, and additions are specified in section 2.2. The parameter restrictions and removals are designed to aid interoperability by reducing the possible variations in the protocol. The parameter additions address issues such as communicating the expected authentication method and communicating a user's **requestor IP/STS**. The Microsoft Web Browser Federated Sign-On Protocol restricts the content of the *wresult* parameter (using the standards as specified in [WSTrust] and [SAMLCore]) to enable interoperable communications of security tokens. The restrictions on the *wresult* parameter are specified in sections 2.2.4.1 and 2.2.4.2. The semantics and protocol details of these changes are addressed in section 3.

1.4 Relationship to Other Protocols

The Microsoft Web Browser Federated Sign-On Protocol uses standard web protocols. The reader should be familiar with the IETF specifications:

- Uniform Resource Locators (URLs), as specified in [RFC1738].
- Uniform Resource Identifiers (URIs), as specified in [RFC2396].
- Hypertext Transfer Protocol (HTTP), as specified in [RFC2616].
- The HTTP State Management Mechanism (for more information, see [RFC2965]).

URLs and URIs are used to describe the data used in the protocol. This protocol depends on HTTP and its dependencies to communicate among IP/STSs, relying parties, and web browser requestors.

The Microsoft Web Browser Federated Sign-On Protocol uses XML. The reader should be familiar with the following W3C specifications:

- Extensible Markup Language (XML) 1.0 (Fourth Edition), as specified in [XML].
- Namespaces in XML, as specified in [XMLNS].
- XML Schema Part 1: Structures Second Edition, as specified in [XMLSCHEMA1].
- XML Schema Part 2: Datatypes Second Edition, as specified in [XMLSCHEMA2].

Copyright © 2013 Microsoft Corporation.

- XML-Signature Syntax and Processing, as specified in [XMLDSig].
- Exclusive XML Canonicalization Version 1.0, as specified in [Excl-C14N].

These specifications are used to describe the requirements for the XML syntax involved in the protocol.

The Microsoft Web Browser Federated Sign-On Protocol uses the model specified in section 2 of [WSFedPRP] and restricts the message and parameters in [WSFedPRP] section 3 to improve interoperability (that is, fewer protocol variations). The reader should be familiar with the following specifications used in [WSFedPRP]:

- WS-Trust, as specified in [WSTrust].
- WS-Addressing, as specified in [WSAddressing].
- WS-Policy, as specified in [WSPolicyAtt].
- WS-Federation, as specified in [WSFederation].

The Microsoft Web Browser Federated Sign-On Protocol uses Security Assertion Markup Language (SAML) 1.1 assertions to communicate security tokens. The reader should be familiar with the OASIS specification for SAML 1.1 assertions (as specified in [SAMLCore]) and the SAML 1.1 XML Schema (as specified in [SAMLASchema]). These specifications build on XML syntax to communicate specific semantics for exchanging security tokens.

Currently, there are no known protocols that depend on the Microsoft Web Browser Federated Sign-On Protocol.

The Microsoft Web Browser Federated Sign-On Protocol is an alternative to the Web Browser SSO Protocol (for more information, see [SAML20Prof]), the Single Sign-On and Federation Protocol (for more information, see [IDFF]), or the Passport Protocol (for more information, see [MS-PASS]). While the Microsoft Web Browser Federated Sign-On Protocol and the Passport Protocol (for more information, see [MS-PASS]) are similar, there are two primary differences between them. The Microsoft Web Browser Federated Sign-On Protocol incorporates a rich abstract data model for communicating user information beyond a user name, while the Passport Protocol is primarily focused on communicating a user name across security realms. The Microsoft Web Browser Federated Sign-On Protocol is also intended to enable XML web service standards-based interoperability, while the Passport Protocol is not based on XML web service standards.

1.5 Prerequisites/Preconditions

For a relying party to verify information from an IP/STS, it must have the key material necessary to verify the digital signature on the message being communicated. The details of IP/STS configuration are vendor-specific.

For a requestor to initiate and engage in this protocol, a web browser requestor must be capable of both submitting HTML forms and following HTTP 302 redirects, as specified in [HTML] and [RFC2616].

The appropriate **trusts** must be in place to allow security tokens issued by the requestor IP/STS to be trusted by the **resource IP/STS**, and those issued by the resource IP/STS to be trusted by the WS resource. The trusts establishment methods are not addressed in this specification.

The web browser requestor is assumed to have knowledge of the URLs that correspond to WS resources that are protected using this protocol. The communication method of this information is not addressed in this specification.

Copyright © 2013 Microsoft Corporation.

1.6 Applicability Statement

The Microsoft Web Browser Federated Sign-On Protocol is used when a web browser requestor needs to communicate user information from one **security realm** to one or more HTTP WS resources in other security realms while avoiding repeated requests for authentication. This protocol is not applicable for non-HTTP-based applications.<1>

1.7 Versioning and Capability Negotiation

1.7.1 Versioning

This specification defers all versioning issues to the specifications [WSFederation], [WSFedPRP], and [SAMLCore]. No additional versioning mechanisms are introduced in this specification.

1.7.2 Capability Negotiation

This specification defers all capability negotiation to the specifications [WSFederation], [WSFedPRP], [SAMLCore], and [RFC2616]. No additional capability negotiation mechanisms are introduced in this specification.

1.8 Vendor-Extensible Fields

As specified in section 2, the Microsoft Web Browser Federated Sign-On Protocol uses the SAML 1.1 token format, as specified in [SAMLCore], for security tokens. Vendors may<2> extend the SAML Advice element to communicate extended data in the security token. [MS-MWBE] specifies extensions to this protocol using the SAML Advice element. The XML elements placed under the SAML 1.1 Advice element may be guaranteed to be unique if the vendor registers the XML namespace URN with the Internet Assigned Numbers Authority (IANA).

Vendors may<3> use the existing extensibility points, as specified in [WSFedPRP]. As described in that specification, new URL parameters can be used to communicate extended information as part of the protocol. There is no process for guaranteeing that URL parameters added to the protocol by individual vendors are uniquely named across multiple vendors.

1.9 Standards Assignments

There are no standards assignments for the Microsoft Web Browser Federated Sign-On Protocol beyond those specified in [WSFedPRP], [WSTrust], and [SAMLCore].

Copyright © 2013 Microsoft Corporation.

2 Messages

This section specifies the transport and syntax of request and response messages in normative detail. References to section 3 are included when knowledge of the protocol details are necessary to understand the context of message transport or syntax.

This protocol references commonly used data types as defined in [MS-DTYP].

2.1 Transport

The wsignin1.0, wsignout1.0, and wsignoutcleanup1.0 requests MUST be transmitted using the GET method; they MUST NOT be transmitted using the POST method. This protocol also restricts wsignin1.0 responses; such responses SHOULD<4> be transmitted to relying parties using the POST method.

The requestor IP/STS and relying party SHOULD<5> use the HTTPS URL scheme to identify each IP/STS and WS resource for processing wsignin1.0 requests, wsignin1.0 responses, wsignout1.0 requests, and wsignoutcleanup1.0 requests. If the HTTPS URL scheme is not used, each IP/STS and WS resource is assumed to have other protection mechanisms or decided that protection is not necessary.

This protocol uses the redirection facilities of HTTP 1.1, as specified in [WSFedPRP], to automate message exchange using the web browser requestor. Compliant web browser requestors MUST support HTTP 1.1 status code 302 redirection.

For wsignin1.0 requests, as depicted in the diagram in [WSFedPRP] section 2.1, the web browser requestor MUST authenticate each IP/STS and the WS resource, using SSL/TLS transport security (for more information, see [RFC2246] and [RFC3546]) and X.509 certificates (as specified in [X509]). Before a security token is issued in response to a wsignin1.0 request message, the user MUST be authenticated to the IP/STS. The method by which the IP/STS verifies the user's identity is not addressed in this protocol. It is assumed here that the user's identity is verified in some way by the IP/STS. Requests for wsignout1.0 and wsignoutcleanup1.0, as depicted in the diagram of [WSFedPRP] section 2.2, are not required<6> to be authenticated.

2.2 Message Syntax

Implementations conforming to this protocol MUST support the following messages from specification [WSFedPRP].

Message	[WSFedPRP]
wsignin1.0	Section 3.1
wsignout1.0	Section 3.4
wsignoutcleanup1.0	Section 3.4

Implementations conforming to this protocol SHOULD NOT emit the following messages from [WSFedPRP]. Conforming implementations that receive one of these messages SHOULD<7> return an HTTP 403 error message if they receive such messages.

Message	[WSFedPRP]
xml-attribute-request	Section 3.5

Copyright © 2013 Microsoft Corporation.

Message	[WSFedPRP]
xml-pseudonym-request	Section 3.6

The following sections define the message syntax for the protocol, starting with common syntax for requests and responses. All parameters for protocol request and response messages MUST use HTTP encoding rules, as specified in [RFC2616].<8> For processing semantics on messages with parameters not specified in this document, see section 3.1.5.2.

2.2.1 Common Syntax for Request Messages

[WSFedPRP] sections 3.1 and 3.2 specify the common syntax for requesting security tokens. For processing guidance on handling unsupported parameters, see section 3.1.5.2.

To simplify implementation and improve interoperability by restricting protocol variations, implementations conforming to this protocol SHOULD NOT<9> support the following parameters:

- *wres* (optional in [WSFedPRP]): This URL parameter specifies the URL for the resource accessed. Note that this parameter will be deprecated in the next version of [WSFedPRP].
- *wp* (optional in [WSFedPRP]): This optional parameter is a URL for policy that can be obtained using an HTTP GET.
- *wreq* (optional in [WSFedPRP]): This optional parameter specifies a token request using either a wsse:RequestSecurityToken element or a full request message, as specified in [WSTrust].
- *wreqptr* (optional in [WSFedPRP]): A URL where the requestor IP/STS can find (using an HTTP GET) the relying party's request.

If an implementation chooses to support these parameters (in addition to supporting the full protocol), it will still be compliant with the Microsoft Web Browser Federated Sign-On Protocol.

2.2.2 Common Syntax for Response Messages

[WSFedPRP] section 3.3 specifies the common mechanisms for returning security tokens. For processing guidance on handling unsupported parameters, see section 3.1.5.2. To simplify implementation and improve interoperability by restricting protocol variations, the following parameters are restricted by this protocol:

- *wresult*: The issued security token that MUST be encoded as a RequestSecurityTokenResponse (RSTR) element, as specified in [WSTrust] section 6.2. This format is detailed in section 2.2.4.1.
- *wctx* (optional in specification [WSFedPRP]): An opaque context value that MUST be returned with the response if it is passed in the request by the relying party.<10>

To simplify implementation and improve interoperability by restricting protocol variations, implementations in conformance to this protocol SHOULD NOT<11> support the following parameter:

- *wresultptr* (optional in specification [WSFedPRP]): A URL where the relying party can find (using an HTTP GET) the **security token service's** response.

If an implementation chooses to support this parameter (in addition to supporting the full protocol), it will still be compliant with the Microsoft Web Browser Federated Sign-On Protocol.

Copyright © 2013 Microsoft Corporation.

2.2.3 wsignin1.0 Request Message

The wsignin1.0 request message is sent to the IP/STS to request that a security token be issued for a specific user to allow access to resources managed by the relying party. For normative descriptions and details on this request message, see [WSFedPRP] section 3.2. This message MUST contain either a *wtrealm* parameter or a *wreply* parameter. This message consists of an HTTP GET with the following query string parameters, formatted as specified in [WSFedPRP] sections 3.1 and 3.2:

- *wa*: The value MUST be the literal string "wsignin1.0".
- *wtrealm*: This parameter MUST be included in a request message to a different security realm from the relying party. If present, this value MUST be a URI that the requestor IP/STS and the relying party have agreed to use to identify the security realm of the relying party in messages to the requestor IP/STS. If present, the *wreply* parameter MUST NOT be present. For processing semantics on *wtrealm* and *wreply*, see section 3.1.5.4.2.
- *wreply*: This parameter MUST be included in request messages to the same security realm as the relying party. If present, this value MUST be a URL at the relying party to which responses MUST be directed. If present, the *wtrealm* parameter MUST NOT be present. For processing semantics on *wtrealm* and *wreply*, see section 3.1.5.4.2.
- *wctx* (optional): This value is an opaque context that MAY be passed in the request by the relying party.<12>
- *wct* (optional): This value is the current time at the relying party that MUST be the string encoding of time, using the XML schema <datetime> time with Coordinated Universal Time (UTC) notation.<13>
- *wauth* (optional): This value is a URI that indicates the method of authentication wanted.<14>
- *whr* (optional): This value is a URI that uniquely identifies the requestor IP/STS that SHOULD receive the wsignin1.0 request message.<15>
- *ClientRequestID* (optional): This value is a string that is used to specify a request identifier that is used when logging events, including errors or failures that occur while processing the request.<16>
- *login_hint* (optional): This value is a string that is used to provide a hint about the login identifier the end-user might use to log in.<17>
- *username* (optional): This value is a string that is used to provide a hint about the login identifier the end-user might use to log in.<18>

Note *login_hint* and *username* are aliases that signify the same query parameter and either of these query parameters can be used to provide a hint about the login identifier the end-user might use to log in.

2.2.4 wsignin1.0 Response Message

The wsignin1.0 response message is sent to the relying party that requested the security token to be issued. For normative descriptions and specifications on this response message, see [WSFedPRP] section 3.3. This message consists of an HTTP POST with the following parameters encoded in the POST body, as specified in [WSFedPRP] section 3.3:

- *wa*: This value MUST be the literal string "wsignin1.0".

Copyright © 2013 Microsoft Corporation.

- *wresult*: This value MUST be the issued security token encoded as a wst:RequestSecurityTokenResponse. The restrictions on the format of this element are specified further in section 2.2.4.1.

- *wctx* (optional): This value is an opaque context that MUST be returned with the response if it was included in the request by the relying party.<19>

2.2.4.1 High-Level Format of wresult Parameter

The syntax for successful wsignin1.0 response message requires that the *wresult* parameter contain a security token that MUST be encoded as an RSTR element, as specified in [WSTrust] section 6.2. The RSTR MUST contain a RequestedSecurityToken element, as specified in [WSTrust] section 6.2. This child element MUST contain either a security token that is constructed as an Assertion element, as specified in [SAMLCore] section 2.3.2, or encrypted content in an **EncryptedData** element, as specified in [XMLEnc] section 3.4.

If present, the syntax of the **Assertion** element MUST conform to a subset of the SAML assertion syntax, as specified in section 2.2.4.2.

The RSTR MAY<20> contain an AppliesTo element, as specified in [WSPolicyAtt] section 3.4. If present, this child element MUST contain an EndpointReference element, as specified in [WSAddressing] section 2.2. The body of the Address element, also specified in [WSAddressing] section 2.2, SHOULD specify the resource provider's security realm URI. Note that this data is redundant and MUST duplicate the information in the security token's Audience element, as specified in [SAMLCore] section 2.3.2.1.3.

Implementations that conform to this protocol MAY<21> include other optional elements or attributes in the RSTR. Such elements are informative to the requestor to indicate how the issuer processed the request. For further specification, see the RSTR example in section 4.2.1.

2.2.4.2 Security Token Format

As stated, the security token contained in the RequestedSecurityToken element, specified in section 2.2.4.1, MUST be formatted as an Assertion element, as specified in [SAMLCore] section 2.3.2, with the restrictions detailed in this section. For more specifications, [SAMLASchema] describes the full XML schema of a SAML assertion and describes all of the element names used in this section, unless otherwise specified. For processing semantics on SAML security tokens that do not conform to this message specification, see section 3.1.1.1.

The following restrictions are placed on the SAML assertion, as specified in [SAMLCore] section 2.3.2:

- The returned SAML assertion MUST use the schema specified in [SAMLASchema], corresponding to the namespace urn:oasis:names:tc:SAML:1.0:assertion.

- The **MajorVersion** attribute of the Assertion element MUST be 1, and the **MinorVersion** attribute of the Assertion element MUST be 1. These attributes are as specified in [SAMLCore] section 2.3.2.

- The returned SAML assertion MUST specify the **AssertionId**, **Issuer**, and **IssueInstant** attributes on the Assertion element. These attributes are as specified in [SAMLCore] section 2.3.2.

- The assertion MUST contain a Conditions element, which MUST specify the **NotBefore** and **NotOnOrAfter** attributes. These attributes are as specified in [SAMLCore] section 2.3.2.1.1.

Copyright © 2013 Microsoft Corporation.

- The Conditions element MUST contain an AudienceRestrictionCondition element that restricts the audience to the resource IP/STS. This element is specified in [SAMLCore] section 2.3.2.1.3.
- The AudienceRestrictionCondition element MUST contain one and only one Audience element that specifies the URI of the relying party.
- The Advice element, specified in [SAMLCore] section 2.3.2.2, MAY<22> be present in assertions conforming to this specification.

2.2.4.2.1 Assertion Statements

The following restriction is laced on the SAML statements used in the SAML assertion:

- Statements other than the AuthenticationStatement (specified in [SAMLCore] section 2.4.3) and the AttributeStatement (specified in [SAMLCore] section 2.4.4) MUST NOT be placed in the SAML assertion.

2.2.4.2.1.1 Authentication Statements

The following restrictions are placed on the SAML AuthenticationStatement used in the SAML assertion:

- The SAML assertion MUST contain one and only one AuthenticationStatement.
- An AuthenticationStatement MUST have a Subject element.
- The Subject element, as specified in [SAMLCore] section 2.4.2.1, MUST conform to the guidance of section 2.2.4.2.1.3.
- If an AttributeStatement is present, the Subject element in the AuthenticationStatement MUST match the Subject element in the AttributeStatement.
- The AuthenticationMethod and AuthenticationInstant attributes MUST be specified.
- The optional AuthenticationStatement elements SubjectLocality (specified in [SAMLCore] section 2.4.3.1) and AuthorityBinding (specified in [SAMLCore] section 2.4.3.2) MUST NOT be present in the security token.

2.2.4.2.1.2 Attribute Statements

The following restrictions are placed on a SAML AttributeStatement used in the SAML assertion:

- The SAML assertion MAY have one AttributeStatement.
- The SAML assertion MAY have no AttributeStatement.
- The SAML assertion MUST NOT have more than one AttributeStatement.
- The AttributeStatement, if present, MUST have a Subject element.
  - The Subject element MUST match the Subject element in the AuthenticationStatement.
  - The Subject element MUST conform to the guidance of section 2.2.4.2.1.3.
- The AttributeStatement, if present, MUST contain one or more Attribute elements, as specified in [SAMLCore] section 2.4.4.1. Each Attribute element encapsulates a name/value claim.

Copyright © 2013 Microsoft Corporation.

- The Attribute element MUST have **AttributeName** and the corresponding **AttributeNamespace** attributes specified. These attributes are specified in [SAMLCore] section 2.4.4.1. The **AttributeName** attribute specifies the name of the claim, and one or more AttributeValue elements (specified in [SAMLCore] section 2.4.4.1.1) specify the value (or values) of the claim.<23>
- All Attribute elements in the AttributeStatement SHOULD<24> have the namespace URL, http://schemas.xmlsoap.org/claims, for the AttributeNamespace attribute value.

For more information, an example of a SAML attribute may be found in section 4.2.2. Values for the **AttributeName** attribute that correspond to claims are specified in the abstract data model in section 3.

2.2.4.2.1.3 Subject Element

The Subject element is used in both the AuthenticationStatement and the AttributeStatement. The Subject element MUST specify the NameIdentifier element, as specified in [SAMLCore] section 2.4.2.2. The SubjectConfirmation element, as specified in [SAMLCore] section 2.4.2.3, MAY<25> be omitted.

For more details, the schema of a Subject element may be found in [SAMLASchema].

The value for the NameIdentifier element MUST be the value of the EmailAddress, **user principal name (UPN)**, or CommonName claim, as specified in the Abstract Data Model in section 3.1.1.4. The NameIdentifier element MUST specify the **Format** attribute, as specified in [SAMLCore] section 2.4.2.2. The corresponding value of the **Format** attribute MUST be one of the following, as specified in the Abstract Data Model (see section 3.1.1.4).

Claim name	Format attribute URI
EmailAddress	urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
UPN	http://schemas.xmlsoap.org/claims/UPN
CommonName	http://schemas.xmlsoap.org/claims/CommonName

The **NameQualifier** attribute of the NameIdentifier element, as specified in [SAMLCore] section 2.4.2.2, MUST NOT be present. For more details, the schema of a NameIdentifier may be found in [SAMLASchema].

2.2.4.2.2 Security Token Signature

The security token MUST contain an enveloped XML digital signature, as specified in [XMLDSig]. The signature MUST be performed over exclusively canonicalized XML, as specified in [Excl-C14N]. All transforms performed on signed elements MUST be included in the Transforms element, as specified in [XMLDSig] section 4.3.3.4. The signature MUST be produced using the requestor IP/STS private key. The KeyInfo element, as specified in [XMLDSig] section 4.4, MUST either directly include an X.509 V.3 certificate (as specified in [X509]) or reference an X.509 V.3 certificate using the X.509 V.3 **subject key identifier (SKI)**, as specified in [RFC3280] section 4.2.1.2. For further specifications, see [XMLDSig] section 4.4.4. It is recommended<26> that the certificate be included directly in the KeyInfo element, using the X509Certificate element.

The X509SKI element contains the base64-encoded ([RFC4648] section 4) plain (that is, non-DER-encoded) value of an X509 V.3 SKI extension. If the **SubjectKeyIdentifier** field is not present in the certificate, the certificate itself MUST be included directly in KeyInfo. Examples of these fields are found in sections 4.2.3 and 4.2.4.

Copyright © 2013 Microsoft Corporation.

Note that the message format of the security token does not incorporate encryption beyond the encryption provided by SSL/TLS.

2.2.5 wsignout1.0 Request Message

The wsignout1.0 request message is specified in [WSFedPRP] section 3.4 and is a request to a requestor IP/STS to delete the cached session state for a specific user. The protocol does not specify a response and does not guarantee the operation will complete. The wsignout1.0 request message consists of an HTTP GET with the following query string parameters, as specified in [WSFedPRP] section 3.4:

- *wa*: The value MUST be the literal string "wsignout1.0".
- *wreply* (optional): The value is a URL at the relying party to which the web browser requestor SHOULD<27> be redirected once sign-out processing is complete.

2.2.6 wsignoutcleanup1.0 Request Message

The wsignoutcleanup1.0 request message is specified in [WSFedPRP] section 3.4 and is a request to a relying party to delete the cached session state for a specific user. The wsignoutcleanup1.0 request message consists of an HTTP GET with the following query string parameters, as specified in [WSFedPRP] section 3.4 (for processing semantics of wsignoutcleanup1.0, see section 3.3.5.4.5):

- *wa*: The value MUST be the literal string "wsignoutcleanup1.0".
- *wreply* (optional): This value is a URL at the requestor IP/STS to which the web browser requestor SHOULD<28> be redirected once clean-up processing is complete.

The wsignoutcleanup1.0 message is an instruction to relying parties by a requestor IP/STS to delete the cached session state for the specified user. The protocol does not specify a response back to the requestor IP/STS initiating the wsignoutcleanup1.0 message.

2.3 Directory Service Schema Elements

This protocol accesses the following Directory Service schema classes and attributes listed in the following table.

For the syntactic specifications of the following **<Class>** or **<Class><Attribute>** pairs, refer [MS-ADTS], [MS-ADA1], [MS-ADA3].

Class	Attribute
User	All

Copyright © 2013 Microsoft Corporation.

3 Protocol Details

This section addresses the message processing model for the protocol. It includes related information required by an implementation to successfully emit or consume protocol messages, such as an abstract data model for maintaining configuration or state information.

The protocol inherently defines three distinct classes of functionality (or roles) for the entities that emit, transport, and consume protocol messages. The requestor IP/STS, relying party, and web browser requestor roles are described in separate subsections. To improve readability, a fourth section is included at the beginning that describes common details for the requestor IP/STS and relying party roles.

3.1 Common Details for Requestor IP/STS and Relying Party Roles

The requestor IP/STS and relying party roles share some common message processing behavior, and they need the same kind of configuration data. This section describes that shared information and common processing semantics. As specified in section 3.3, it is possible to factor a relying party into separate components, which are defined in the glossary (section 1.1) as a resource IP/STS and a WS resource. Throughout this section, the generic term relying party is used unless it is necessary to distinguish that a topic applies only to a resource IP/STS or only to a WS resource.

3.1.1 Abstract Data Model

Proper operation of the protocol requires that a requestor IP/STS and a relying party maintain configuration information that describes the entities with which they exchange protocol messages. This section describes an abstract data model for maintaining that configuration information.

The following subsections describe a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to help explain how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Note The conceptual data model can be implemented using a variety of techniques. Windows behavior is described for each data item at the end of the appropriate subsection.

3.1.1.1 Security Token

The primary protocol data unit transported by this protocol is a security token constructed as a subset of the SAML 1.1 assertion element syntax, as specified in section 2.2.4.2. Because this is the only security token format supported by this protocol, no abstract data model is introduced to represent a security token. Throughout section 3, wherever it is necessary to discuss internal constructs of a security token, the SAML terminology from section 2.2.4.2 will be used. For further specification, see [SAMLCore].

If a security token returned by this protocol is not formatted as required in section 2.2.4.2, the relying party MUST return an HTTP 1.1 status code 500 server error to the web browser requestor.

3.1.1.2 User Authentication Context

When a requestor IP/STS or a resource IP/STS issues a security token, it MUST authenticate the user to obtain the data required to construct a security token. How the user is authenticated is implementation-specific and not addressed in this protocol. Common practices are discussed in sections 3.2 and 3.3 under the topic of message processing. In addition, when a relying party

Copyright © 2013 Microsoft Corporation.

accepts a security token to authenticate a user, it is necessary to map the security token AuthenticationStatement and AttributeStatement data into the structure that is used by the local system to make access control decisions. This is implementation-specific and not addressed in this protocol. It is useful here to define an abstract data model to record the data returned from the user authentication process. The following is a potential representation to organize this data:<29>

Authentication Context: This record contains data returned from user authentication. An IP/STS MUST maintain a separate record per user. The fields of this record are as follows:

AuthIdentity: A string field that uniquely and authoritatively identifies the user.

AuthMethod: A string field that identifies the authentication mechanism used, as required for the SAML assertion in a security token, as described in section 2.2.4.2.

AuthTime: A date-time field that identifies the most recent time that the user was authenticated.

AuthStart: A date-time field used to hold the beginning of the validity interval for a security token that is specified by the **NotBefore** attribute, as discussed in section 2.2.4.2.

AuthStop: A date-time field used to hold the end of the validity interval for a security token that is specified by the **NotOnOrAfter** attribute, as discussed in section 2.2.4.2.

AuthGroups (optional): A string field used to contain a list of group names, if any exist for this user, that are returned by the **AuthMethod**.

AuthClaim (optional): This record holds a claim from a security token, as defined in section 3.1.1.4. There MUST be one claim per **AuthClaim** record; thus multiple records may be present.

3.1.1.3 Federation Partner

A **federation** partner is a generic term used to refer to a security realm that operates either a requestor IP/STS or a relying party service (or both) capable of performing the mandatory Microsoft Web Browser Federated Sign-On Protocol operations defined in this specification.

A requestor IP/STS and a relying party MUST exchange configuration metadata before they begin exchanging protocol messages. How this metadata should be exchanged is implementation-specific and not addressed in this protocol. Administrators MAY<30> exchange files by an out-of-band process (such as email attachments) and enter the partner's metadata into a local configuration file or database.

A requestor IP/STS and a relying party MUST exchange at least the following metadata before they begin exchanging protocol messages:

- Unique identifiers that indicate the security realms that they represent, and distinguish them from other possible federation partners.
- URLs that indicate where protocol messages are to be sent.

Also, the relying party MUST obtain the X.509 certificate that contains the public key that corresponds to the private key that the requestor IP/STS uses for signing security tokens.

The following is a potential representation for organizing this data. The data is organized as a series of records, each representing a federation partner. The fields of this record are as follows:

- **Identifier**: A string field that uniquely identifies the security realm of the partner. It must be the value that is used for the wtrealm parameter for wsignin1.0 request messages. For details, see section 2.2.3.

Copyright © 2013 Microsoft Corporation.

- **Role**: A string field that identifies the role (or roles) played by the partner, and thus the types of protocol messages that it can send or receive. There are two possible values for this field, requestor IP/STS and relying party. This field must contain at least one value. If a federation partner is capable of playing both roles, this may be represented by a single record with both requestor IP/STS and relying party values present, or it MAY be represented by two separate records with different values for the **Role** field.<31>
- **URL**: A field indicating the URL to which all protocol messages that are intended for the federation partner identified by this record must be sent.
- **Certificate** (optional): A field indicating an X.509 certificate that holds the public key that corresponds to the private key used by the partner for signing security tokens. If the **Role** field contains requestor IP/STS, a certificate must be present. Otherwise this field does not apply. Note that the field can contain multiple certificates. If the requestor IP/STS is a farm of servers, all of the servers may use the same private key, or each server may use a different private key.<32>

A security token, as described in section 2.2.4.2, MUST contain an AuthenticationStatement that specifies the identity of the user in the Subject element. The wsignin1.0 request message does not contain a provision to specify additional claims that the relying party requires to determine if a user is authorized to access a particular WS resource. If specific claims are to be included in a security token, federation partners MUST agree on them before protocol messages are exchanged. Specific claims can be requested as part of the metadata exchange and included in the federation partner record.

The following is a potential representation for organizing this data. Two fields are added to the abstract data model of a federation partner record. For further specifications on the AttributeStatement element of a security token mentioned in the following data definitions, see section 2.2.4.2:

- **ClaimsOut** (optional): A list of claims, as defined in section 3.1.1.4, that SHOULD be included in an AttributeStatement of a security token sent in a wsignin1.0 response message to the federation partner. This SHOULD match the corresponding **ClaimsIn** record at the federation partner.<33>
- **ClaimsIn** (optional): A list of claims, as specified in section 3.1.1.4, that MAY be included in the AttributeStatement of a security token received in a wsignin1.0 response message from the federation partner. This SHOULD match the corresponding **ClaimsOut** record at the federation partner.<34>

This protocol does not address whether or not these additional claims should be treated as optional or mandatory by the requestor IP/STS. There is no guarantee that a particular user will have the attribute data necessary to satisfy every claim on the **ClaimsOut** list for every relying party that requests a security token for that user. Whether or not a requestor IP/STS should fail a wsignin1.0 request message if it cannot set every claim is an implementation-specific detail that is not addressed in this protocol. It is recommended that a security token be issued and the relying party be allowed to decide if it has sufficient information about the user to grant access to the protected resource in question.<35>

3.1.1.4 Claim

A security token MAY<36> contain an AttributeStatement, with one or more Attribute elements, each of which contains a single claim, as specified in section 2.2.4.2. A claim is uniquely identified by its **AttributeName** attribute and <AttributeValue> element.

Copyright © 2013 Microsoft Corporation.

This protocol restricts the syntax and the interpretation of the semantics of these five claims to the following definitions. See section 2.2.4.2 for further specification on the AttributeStatement element of a security token and the usage of **AttributeName**, **AttributeNamespace**, and <AttributeValue> in the following claim definitions:

EmailAddress claim (optional): This claim is used to identify a Subject via an email address.

- The **AttributeName** attribute MUST be "EmailAddress".
- The **AttributeNamespace** attribute MUST be the URL http://schemas.xmlsoap.org/claims.
- The <AttributeValue> element content MUST conform to "addr-spec", as specified in [RFC2822]. The value MUST be unique within the security realm of the requestor IP/STS that issued the security token such that a relying party could use it to make an access control decision.
- When this claim is used for the value of the Subject/NameIdentifier element of an AuthenticationStatement or AttributeStatement, the value of the **Format** attribute MUST be URI urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress.

UPN claim (optional): This claim is used to identify a Subject via a UPN.

- The **AttributeName** attribute MUST be "UPN".
- The **AttributeNamespace** attribute MUST be URL http://schemas.xmlsoap.org/claims.
- The <AttributeValue> element content MUST be a UPN. The value MUST be unique within the security realm of the requestor IP/STS that issued the security token such that a relying party could use it to make an access control decision.
- When this claim is used for the value of the Subject/NameIdentifier element of an AuthenticationStatement or AttributeStatement, the value of the Format attribute MUST be the URL http://schemas.xmlsoap.org/claims/UPN.

CommonName claim (optional): This claim is used to identify a Subject via a common name (CN) value consistent with X.500 naming conventions.

- The **AttributeName** attribute MUST be "CommonName".
- The **AttributeNamespace** attribute MUST be the URL http://schemas.xmlsoap.org/claims.
- The <AttributeValue> element content MUST conform to CommonName, as specified in [X500]. The value of this claim is not necessarily unique and MUST NOT be used by a relying party to make an access control decision. It is suitable for displaying a friendly name for personalization.
- When this claim is used for the value of the Subject/NameIdentifier element of an AuthenticationStatement or AttributeStatement, the value of the Format attribute MUST be the URL http://schemas.xmlsoap.org/claims/CommonName.

Group claim (optional): This claim is used to indicate the association of the subject with other users that share a common characteristic. The semantic meaning of that association is application specific, but the common interpretation is group or role membership.

- The **AttributeName** attribute MUST be "Group".
- The **AttributeNamespace** attribute MUST be the URL http://schemas.xmlsoap.org/claims.
- The <AttributeValue> element content MUST be a string.

Copyright © 2013 Microsoft Corporation.

For more information about Group claims, see Appendix A: Windows Behavior.<37>

Implementations MAY define additional claims with prior agreement between federation partners and MUST conform to the following structure:<38>

Custom claim (optional): This claim is used to identify an application-specific attribute possessed by the subject.

- The **AttributeName** attribute MUST be a string constant, per agreement between federation partners.
- The **AttributeNamespace** attribute SHOULD be the URL http://schemas.xmlsoap.org/claims.
- The <AttributeValue> element content MAY be an arbitrary data type per agreement between federation partners.

3.1.1.5 Federation Partner Session Lists for Web Browser Requestors

The protocol wsignout1.0 request message and wsignoutcleanup1.0 messages do not explicitly identify the user who has triggered the sign-out operation. The web browser requestor that transports these messages is the only link to the user. Thus, to process the wsignout1.0 request message and the wsignoutcleanup1.0 request message, a user's activity MUST be tracked in terms of the activity of the user's web browser requestor. Federation partners MUST uniquely identify individual web browser requestors. This MAY<39> be done by setting an HTTP session cookie that contains a unique identifier. For more information, see [RFC2965].

web browser requestor session: This is a list of security tokens that was issued to (or received from) a specific instance of a web browser requestor in response to Microsoft Web Browser Federated Sign-On Protocol messages. A web browser requestor session is delimited by the user starting and stopping an instance of the software that implements a web browser requestor. For example, if a user was to start two instances of the same web browser requestor software in parallel to obtain security tokens, this would be treated as two sessions. A requestor IP/STS MUST maintain the list in terms of relying parties using the value of the Audience element from each security token issued during the web browser requestor session. A relying party MUST maintain the list in terms of requestor IP/STSs using the value of the Issuer element from each security token received during the web browser requestor session.

3.1.1.5.1 Requestor IP/STS Web Browser Requestor Sessions List

The following is a potential representation for a requestor IP/STS to organize the data for tracking web browser requestor sessions to support processing of a wsignout1.0 request and a wsignoutcleanup1.0 request message. The data is organized as a list of records, each representing a particular web browser requestor session.

The following is a potential representation for organizing the data record that represents a particular web browser requestor session on the list.

Outbound Sessions List: This data element marks the beginning of the list of web browser requestor sessions.

WebBrowserRequestorSession: This record holds the list of security tokens issued for a particular web browser requestor session. The fields of this record are as follows:<40>

- **Requestor Session Identifier**: This uniquely identifies a particular web browser requestor session. A WebBrowserRequestorSession record is added to the Outbound Sessions List when the first security token is issued for a particular web browser requestor session.

Copyright © 2013 Microsoft Corporation.

- **Session Entry**: This identifies a relying party based on the Audience element content from a security token. A Session Entry is added to the record when a security token is issued for the web browser requestor session.

3.1.1.5.2 Relying Party Web Browser Requestor Sessions List

The following is a potential representation for a relying party to organize the data for tracking web browser requestor sessions to support processing of wsignout1.0 and wsignoutcleanup1.0 messages. The data is organized as a list of records, each representing a particular web browser requestor session.

The following is a potential representation for organizing the data record that represents a particular web browser requestor session on the list.

Inbound Sessions List: This data element marks the beginning of the list of web browser requestor sessions.

WebBrowserRequestorSession: This record holds the list of security tokens received for a particular web browser requestor session. The fields of this record are as follows:<41>

- **Requestor Session Identifier**: This uniquely identifies a particular web browser requestor session. A WebBrowserRequestorSession record is added to the Inbound Sessions List when the first security token is received for a particular web browser requestor session.
- **Session Entry**: This identifies a requestor IP/STS based on the Issuer attribute value from a security token. A Session Entry is added to the record when a security token is received for the web browser requestor session.

3.1.2 Timers

This protocol does not require timers beyond those that may be used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS.

The security tokens transported in protocol messages have a specific time interval during which they are considered to be valid. This MUST be set by the security token service that issues the security tokens using the **NotBefore** and **NotOnOrAfter** attributes of the Conditions element. For further specifications, see [SAMLCore] section 2.3.2.1.1. A relying party SHOULD NOT accept security tokens if the current time is equal to or greater than the value of the **NotOnOrAfter** attribute or equal to or less than the value of the **NotBefore** attribute.

When an Authentication Context is created from security tokens, the **AuthStart** and **AuthStop** fields MUST be set from the **NotBefore** and **NotOnOrAfter** attributes of the security tokens. A resource IP/STS SHOULD NOT use an Authentication Context to grant access to a WS resource if the current time falls outside the validity interval defined by the **AuthStart** and **AuthStop** values.<42>

Implementations MAY<43> use a timer to indicate when the validity interval of a security token or an Authentication Context expires to control maintenance operations (for example, flushing caches), but the protocol does not require the use of a timer.

3.1.3 Initialization

Requestor IP/STSs and relying parties MUST exchange metadata and initialize federation partner records in local configuration data, as specified in section 3.1.1.3. There is no protocol-specific initialization required for a web browser requestor.

Copyright © 2013 Microsoft Corporation.

3.1.4 Higher-Layer Triggered Events

Administrative actions starting or stopping the requestor IP/STS or relying party services, or changing service configuration data do not trigger protocol message exchanges. Protocol request messages are triggered by user action via a web browser requestor.

The wsignin1.0 request message is triggered when a web browser requestor sends an HTTP GET to a WS resource at a WS resource that requires users to be authenticated using this protocol.

The wsignout1.0 request message is triggered when a user explicitly requests to initiate the sign-out operation at a WS resource.

The wsignoutcleanup1.0 request message is triggered when a user explicitly requests to initiate the sign-out operation at a requestor IP/STS, or in response to the wsignout1.0 message discussed earlier.

3.1.5 Processing Events and Sequencing Rules

This section describes message processing functionality that is common to federation partners, regardless of the specific role they play.

3.1.5.1 Determining Message Type

A requestor IP/STS and a relying party process several different types of messages. Upon receipt of a message, they determine its type by inspection of the *wa* parameter. The *wa* parameter MUST be included in every message used by this protocol (for further specification, see section 2.2). The *wa* parameter, and all other protocol-specific parameters, MUST be transported in query string parameters when the protocol message is transported using an HTTP GET. These parameters MUST be transported in the body of the HTTP POST when the protocol message is transported using an HTTP POST. If protocol-specific parameters are included in query string parameters with an HTTP POST, they SHOULD<44> be ignored.

A web browser requestor is not required to understand the types or content of the protocol messages it transports between a requestor IP/STS and a relying party.<45>

3.1.5.2 Error Handling

If a protocol message is received that includes an optional parameter that is not supported by an implementation, or a parameter that is not specified by this protocol, a requestor IP/STS MAY<46> ignore the parameter and process the message as though it were not present.

Protocol messages are processed by a requestor IP/STS or a relying party. If a message cannot be processed successfully, the protocol allows for a SOAP fault to be returned to the appropriate federation partner via a web browser requestor using an HTTP POST method.

However, this protocol does not require end-to-end error message propagation. A requestor IP/STS or a relying party MAY return an HTTP 500 error response to the web browser requestor, or they MAY redirect it to an error page.<47>

This protocol does not include provisions for automated retry or recovery processing if there is no response to a protocol request message. This protocol is designed to rely on the intervention of the user to notice when a web browser requestor operation has failed and to manually retry the operation that triggered the protocol.

Copyright © 2013 Microsoft Corporation.

3.1.5.3 Requesting a Security Token by Issuing a wsignin1.0 Request Message

This section describes the generic message processing performed by a relying party when requesting a security token. The purpose is to authenticate a user by requesting a security token to be issued for a specific user with the relying party as the audience.

The generic processing model for sending a wsignin1.0 message MUST consist of the following steps, as described in the following subsections:

- Protocol Activation (section 3.1.5.3.1)
- Parameter Marshaling (section 3.1.5.3.2)
- Requestor IP/STS Security Realm Discovery (section 3.1.5.3.3)
- Message Transmission (section 3.1.5.3.4)

3.1.5.3.1 Protocol Activation

For a relying party, the protocol is triggered when a web browser requestor attempts to access a WS resource that requires users to be authenticated, and an Authentication Context does not exist for the users.

3.1.5.3.2 Parameter Marshaling

The relying party MUST set all the required query string parameters for a wsignin1.0 request message, as specified in section 2.2.3. It MUST set the *wtrealm* value from the configuration data. It SHOULD<48> store the original application URL provided by WS resource in *wctx*.

3.1.5.3.3 Requestor IP/STS Security Realm Discovery

Because the user has not yet been authenticated, the relying party may not know where to send a wsignin1.0 request message. The relying party MUST obtain the security realm identifier for the requestor IP/STS that issues security tokens for the user. When the relying party is factored into WS resource and resource IP/STS components, the WS resource relies on the resource IP/STS to discover the correct federation partner. The resource IP/STS MAY discover the security realm of the federation partner by interacting with the user via the user's web browser requestor. Or the information MAY be obtained from a parameter (such as *whr*, specified in section 2.2.1) that was included on the original request to WS resource.<49>

The resource IP/STS MUST use the security realm identifier to look up the correct federation partner record in configuration data and obtain the requestor IP/STS URL for protocol messages. Once obtained, the security realm identifier MAY<50> be preserved for subsequent sessions by writing an HTTP persistent cookie (for more information, see [RFC2965]) to the web browser requestor.

3.1.5.3.4 Message Transmission

The relying party sends a wsignin1.0 request message by returning an HTTP 302 response to the web browser requestor with the Location field set to the URL of the requestor IP/STS. All the query string parameters required for the protocol MUST be set properly, as specified in section 2.2.3.

3.1.5.4 Issuing a Security Token by Responding to a wsignin1.0 Request Message

This section describes the generic message processing performed by either a requestor IP/STS or a resource IP/STS (a component of a relying party specified in section 3.3) when issuing a security

Copyright © 2013 Microsoft Corporation.

token. IP/STS is used generically to see either a requestor IP/STS or a resource IP/STS in the following subsections.

The generic processing model for responding to a wsignin1.0 message is specified to consist of the following steps, as described in the following subsections:

- Protocol Activation (section 3.1.5.4.1)
- Message Validation (section 3.1.5.4.2)
- User Identification and Authentication (section 3.1.5.4.3)
- User Attribute Retrieval (section 3.1.5.4.4)
- Claim Mapping (section 3.1.5.4.5)
- SAML Assertion Construction (section 3.1.5.4.6)
- Response Message Processing (section 3.1.5.4.7)

3.1.5.4.1 Protocol Activation

The protocol is triggered by receipt of a wsignin1.0 request message. The syntax MUST be validated by ensuring that all required parameters are present and contain the correct type of data. For further specifications, see section 2.2.3.

3.1.5.4.2 Message Validation

The syntax MUST be validated by ensuring that all required parameters are present and contain the correct type of data. For further specifications, see section 2.2.3.

Before issuing a security token to protect the user's privacy, the IP/STS MUST verify that the entity that sent the wsignin1.0 request message is a federated partner that holds the role of relying party, as described in the Abstract Data Model. The relying party's identifier MUST be retrieved from the *wtrealm* parameter in the request (as specified in section 2.2.1) and compared against the federation partner configuration data (as specified in section 3.1.1.2).

If a resource IP/STS receives a request that has both the *wreply* and *wtrealm* parameters set, the resource IP/STS MUST return an HTTP 1.1 status code 500 server error. If a requestor IP/STS receives a request that has both the *wreply* and *wtrealm* parameters set, the requestor IP/STS MUST ignore the *wreply*.

The *wauth* parameter, which is described in section 2.2.3, MAY<51> be used.

3.1.5.4.3 User Identification and Authentication

The user's identity is not conveyed explicitly in the wsignin1.0 request message. The IP/STS MUST establish this by initiating a message exchange with the web browser requestor that will cause the users to identify themselves and prove their right to assert that identity.

The user authentication methods are implementation-specific and are not addressed in this protocol. It is recommended that the IP/STS employ a standard protocol to authenticate the user, such as Kerberos (for more information, see [RFC4120]). It MAY use an HTML form to collect credentials directly from the user (ideally using HTTPS) and compare them against a local database. Or it MAY authenticate the user by initiating the Microsoft Web Browser Federated Sign-On Protocol with another STS.<52>

Copyright © 2013 Microsoft Corporation.

Whatever method is used, the IP/STS MUST store the results in the Authentication Context, as defined in section 3.1.1.2.

If the user cannot successfully authenticate, the IP/STS MUST abort processing the request and return an error message to the user.

3.1.5.4.4 User Attribute Retrieval

If additional claims are required by a ClaimsOut entry for the relying party, as specified in section 3.1.1.2, the IP/STS MUST retrieve the correct values that correspond to the authenticated identity of the user. How this is performed is implementation-specific and not addressed in this protocol. The IP/STS SHOULD<53> retrieve the data from an authoritative user attribute authority based on the value of the AuthIdentity element from the user's Authentication Context.

3.1.5.4.5 Claim Mapping

If additional claims are required by a ClaimsOut entry for the relying party, the IP/STS MUST map user attributes retrieved in the previous step to their corresponding claims.

3.1.5.4.6 SAML Assertion Construction

Having assembled the required set of claims, the IP/STS constructs a security token according to the SAML 1.1 assertion syntax specified in section 2.2.4.2. The relying party's identity MUST be used to populate the Audience element of the required AuthenticationStatement. From the Authentication Context, AuthIdentity, AuthMethod, and AuthTime MUST be used to populate the Subject element, AuthenticationMethod attribute, and AuthenticationInstant, respectively, of the AuthenticationStatement. Additional claims required by the relying party MUST be placed in separate Attribute elements in the AttributeStatement.

As specified in section 2.2.4.1, the security token MUST be encoded as a wst:RequestSecurityTokenResponse and returned in the *wresult* parameter of the wsignin1.0 response message. The IP/STS MAY encrypt the security token as described in section 2.2.4.1.

3.1.5.4.7 Response Message Processing

All required parameters for a wsignin1.0 response message MUST be set correctly. The completed response message MUST be returned in an HTTP POST, as specified in section 2.1, to the URL designated for the relying party that sent the request.<54>

3.1.6 Timer Events

There are no protocol-specific timer events that MUST be serviced by an implementation. This protocol does not require timers beyond those that may be used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS. The protocol does not include provisions for time-based retry for sending protocol messages.

Security tokens and Authentication Contexts do have validity intervals, as specified in section 3.1.2. Implementations MAY<55> use a timer to indicate when the validity interval of a security token or an Authentication Context expires, but the protocol does not require the use of a timer.

3.1.7 Other Local Events

This protocol does not have dependencies on other protocols other than HTTP 1.1 and SSL/TLS to transport protocol messages. The Microsoft Web Browser Federated Sign-On Protocol relies on this

Copyright © 2013 Microsoft Corporation.

transport mechanism for the correct and timely delivery of protocol messages. The protocol does not take action in response to any changes or failure in machine state or network communications.

3.2 Requestor IP/STS Details

This section describes details of protocol processing that must be understood, in addition to the information in section 3.1, to implement a requestor IP/STS that can correctly perform its role in the protocol message exchange.

3.2.1 Abstract Data Model

A requestor IP/STS does not require additions to the abstract data model (section 3.1.1).

3.2.2 Timers

A requestor IP/STS does not depend on timers beyond those that may be used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS or those specified in section 3.1.2.

3.2.3 Initialization

Before any protocol messages can be exchanged, a requestor IP/STS MUST exchange metadata with relying parties and initialize federation partner records for them in local configuration data, as specified in section 3.1.1.2.

To service protocol messages, a requestor IP/STS MUST be listening for requests at the URL it has advertised to federation partners.

To service wsignin1.0 request messages, a requestor IP/STS MUST be able to contact user authentication and account services in its local security realm to obtain the authenticated user identity and optional attributes necessary for constructing a security token.

The protocol does not require specific initialization on receipt of a protocol message.

3.2.4 Higher-Layer Triggered Events

In addition to user-triggered events (see section 3.1.4), a requestor IP/STS is triggered on receipt of a protocol message to process that message and respond to the federation partner that sent it (see section 3.2.5).

3.2.5 Processing Events and Sequencing Rules

This section describes the logical steps performed by a requestor IP/STS when processing the following protocol messages:

- wsignin1.0
- wsignout1.0
- wsignoutcleanup1.0

Copyright © 2013 Microsoft Corporation.

3.2.5.1 Issuing a Security Token by Responding to a wsignin1.0 Request Message

This message is received by a requestor IP/STS from a relying party. The generic model for responding to a wsignin1.0 message is specified to consist of the following steps (from section 3.1.5.4):

- Protocol Activation (section 3.1.5.4.1)
- Message Validation (section 3.1.5.4.2)
- User Identification and Authentication (section 3.1.5.4.3)
- User Attribute Retrieval (section 3.1.5.4.4)
- Claim Mapping (section 3.1.5.4.5)
- SAML Assertion Construction (section 3.1.5.4.6)
- Response Message Processing (section 3.1.5.4.7)

A requestor IP/STS MUST process a wsignin1.0 request message, as described in the preceding sections, with the following exception:

- User identification and authentication

  It is not possible for a requestor IP/STS to authenticate the user by using this protocol with an IP/STS in another security realm because the user account is managed in the security realm of the requestor IP/STS.

3.2.5.2 Inbound wsignout1.0 Request Message Processing

This message is used to request that a requestor IP/STS initiate clean-up operations for cached session state, if any exists, for the user who triggered the request.

3.2.5.2.1 Protocol Activation

The protocol is triggered by receipt of a wsignout1.0 request message. The request MUST be validated by ensuring that all required parameters are present and contain the correct type of data (as specified in section 2.2.3).

3.2.5.2.2 Clean-Up Processing

A requestor IP/STS SHOULD delete any session state that has been locally cached for the web browser requestor that delivered the wsignout1.0 request message. A requestor IP/STS SHOULD send a wsignoutcleanup1.0 message to each relying party to which a security token has been issued for the web browser requestor that delivered the wsignout1.0 request message, as specified in section 3.2.5.3.<56>

3.2.5.2.3 Response Message Processing

If a *wreply* parameter was included in the request, as specified in section 2.2.5, the requestor IP/STS SHOULD redirect the web browser requestor to the specified URL when it has completed clean-up operations.<57>

Copyright © 2013 Microsoft Corporation.

3.2.5.3 Outbound wsignoutcleanup1.0 Request Message Processing

This message is used to request that a relying party initiate clean-up operations for cached session state, if any exists, for the user who triggered the request.

3.2.5.3.1 Protocol Activation

The protocol is triggered either when a user requests that the user's session be terminated, possibly by clicking a sign-out button, or when a wsignout1.0 request message is received from a relying party.

3.2.5.3.2 Relying Party Security Realm Discovery

As specified in section 3.1.1.5.1, a requestor IP/STS MAY store a session identifier for the web browser requestor in an HTTP session cookie (for more information, see [RFC2965]). It uses this identifier and its Outbound Sessions List to develop the list of relying parties to which it SHOULD<58> send wsignoutcleanup1.0 request messages.

3.2.5.3.3 Clean-Up Processing

Because the protocol does not guarantee a response to a wsignoutcleanup1.0 message, a requestor IP/STS SHOULD clean up the locally cached session state that it maintains before sending wsignoutcleanup1.0 messages. For example, as specified in section 3.1.1.5.1, the WebBrowserRequestorSession record SHOULD be deleted from the Outbound Sessions List. As each wsignoutcleanup1.0 message is sent, the requestor IP/STS SHOULD delete the corresponding Session Entry from the record. When the last Session Entry is deleted, the WebBrowserRequestorSession SHOULD be deleted from the Outbound Sessions List.<59>

3.2.5.3.4 Message Transmission

The requestor IP/STS SHOULD send a wsignoutcleanup1.0 request message to each relying party using an explicit HTTP GET method because the protocol does not support chaining wsignoutcleanup1.0 messages using the HTTP 1.1 redirection facilities. How these messages are sent is implementation-specific and not addressed in this protocol. The requestor IP/STS MAY walk the list of relying parties and issue the requests individually.<60>

3.2.6 Timer Events

A requestor IP/STS does not need to interact with any timers, or service any timer events, beyond those that may be used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS, or those specified in section 3.1.6.<61>

3.2.7 Other Local Events

A requestor IP/STS does not have dependencies on local events beyond what is specified in section 3.1.7.

3.3 Relying Party Details

This section describes details of protocol processing that must be understood, in addition to the information from section 3.1, to implement a relying party that can correctly perform its role in the protocol message exchange.

Copyright © 2013 Microsoft Corporation.

3.3.1 Abstract Data Model

A relying party performs two distinct functions. It processes protocol messages and the security tokens it receives, and it controls user access to protected resources based on the contents of those security tokens. The latter function is implementation-specific and not addressed in this protocol. A relying party MAY<62> be factored into separate components, a resource IP/STS, and a WS resource as follows. The resource IP/STS component MUST be able to perform all of the metadata and protocol message exchanges required to obtain a security token from a requestor IP/STS in another security realm. The WS resource component SHOULD use the Authentication Context derived from that security token to control user access.

The abstract data model for a relying party MAY<63> be extended to enable this componentization without requiring a requestor IP/STS to have interior knowledge of the relying party structure. This implementation-approach is supported by the following extensions of the abstract data model for a federation partner, as specified in section 3.1.1.3.

3.3.1.1 Resource IP/STS Abstract Data Model Extensions

The following is a potential representation for a resource IP/STS to organize the data that represents its federation partners. The federation partner record is used, as specified in section 3.1.1.3, with the following extensions and dependencies for the possible range of values for fields:

Identifier: For a requestor IP/STS from another security realm, it MUST be the *wtrealm* value, as specified in section 3.1.1.3. For a WS resource, it MUST be an identifier that is unique within the security realm such as a web server or web application URL or URI.

Role: There are three possible values for this field: requestor IP/STS, relying party, and WS resource. For a partner from another security realm, this field MAY contain requestor IP/STS or relying party or both values, as specified in section 3.1.1.3. For a partner within the security realm, this field SHOULD only contain WS resource.<64>

URL: If the **Role** field contains requestor IP/STS or relying party, this field MUST contain a URL. If the **Role** field contains WS resource, a URL MAY be present but is not required because this information can be reliably passed in using the *wreply* parameter.<65>

Certificate (optional): If the **Role** field contains requestor IP/STS, the **Certificate** field MUST contain a certificate, as specified in section 3.1.1.3. Otherwise, this field does not apply.

3.3.1.2 WS Resource Abstract Data Model Extensions

Following is a potential representation for a WS resource to organize the data that represents its federation partners. The federation partner record is used, as specified in section 3.1.1.3, with the following restrictions on the range of values for the **Role** field:

- **Role**: A string field that identifies the role (or roles) played by the partner and thus the types of protocol messages that it can send or receive. The only possible value for this field is resource IP/STS.

3.3.2 Timers

A relying party does not depend on timers beyond those that may be used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS, or those specified in section 3.1.2.<66>

Copyright © 2013 Microsoft Corporation.

3.3.3 Initialization

Before protocol messages can be exchanged, a relying party MUST exchange metadata with requestor IP/STSs and initialize federation partner records for them in local configuration data, as specified in section 3.1.1.2.

To service protocol messages, a relying party MUST be listening for requests at the URL it has advertised to federation partners.

To service wsignin1.0 response messages, a relying party SHOULD<67> have network access to the certificate revocation list (CRL) distribution point (CDP) contained in X.509 certificates obtained from federation partners for the purpose of validating security token signatures, as specified in section 3.1.1.2.

The protocol does not require specific initialization upon receipt of a protocol message.

3.3.4 Higher-Layer Triggered Events

In addition to the user-triggered events discussed in section 3.1.4, a relying party is triggered upon receipt of a protocol message to process that message and respond to the federation partner that sent it, as specified in section 3.3.5.

3.3.5 Processing Events and Sequencing Rules

This section describes the logical steps performed by a relying party when processing the following protocol messages:

- wsignin1.0
- wsignout1.0
- wsignoutcleanup1.0

Message processing by the relying party is described separately for the resource IP/STS and WS resource components that were introduced in the revised abstract data model (section 3.3.1).

3.3.5.1 Requesting a Security Token by Sending a wsignin1.0 Request Message

This message is sent by a relying party to a requestor IP/STS to request that a security token be issued for a specific user with the relying party as the audience. The generic model for sending a wsignin1.0 message is specified to consist of the following steps in Requesting a Security Token by Issuing a wsignin1.0 Request Message (section 3.1.5.3):

- Protocol Activation (section 3.1.5.3.1)
- Parameter Marshaling (section 3.1.5.3.2)
- Requestor IP/STS Security Realm Discovery (section 3.1.5.3.3)
- Message Transmission (section 3.1.5.3.4)

A relying party MUST process a wsignin1.0 request message (section 2.2.3), as specified in the preceding sections, with the following exceptions:

- Protocol Activation (section 3.3.5.1.1)
- Parameter Marshaling (Message Validation (section 3.3.5.2.2))

Copyright © 2013 Microsoft Corporation.

3.3.5.1.1 Protocol Activation

When a relying party is factored into resource IP/STS and WS resource components, the protocol is triggered differently for the components. For the WS resource component, the protocol is triggered when a web browser requestor attempts to access a WS resource that requires users to be authenticated and an Authentication Context does not exist for the user. For the resource IP/STS component, the protocol is triggered by receipt of a user authentication request from the WS resource component. If the components are located on separate servers, the WS resource MUST redirect the web browser requestor to the resource IP/STS to deliver the security token request. How user authentication requests are communicated between the components of a relying party is implementation-specific and not addressed in this protocol.<68>

3.3.5.1.2 Parameter Marshaling

A resource IP/STS that sends a wsignin1.0 request message (section 2.2.3) to a security token service in a different security realm MUST set the *wtrealm* parameter, as specified in Common Syntax for Request Messages (section 2.2.1). A WS resource that sends a wsignin1.0 request message to a security token service in the same security realm cannot use the *wtrealm* parameter. It MAY use the *wreply* parameter to distinguish itself from other WS resources in the security realm.<69>

3.3.5.2 Receiving a Security Token by Processing a wsignin1.0 Response Message

This message is received by a resource IP/STS from a requestor IP/STS. The purpose is to accept a security token that was issued for a specific user, with the resource IP/STS as the audience, in response to a wsignin1.0 request message. The generic model for relying party processing of a wsignin1.0 response message is specified to consist of the following steps, specified in the following sections:

- Protocol Activation (section 3.3.5.2.1)
- Message Validation (section 3.3.5.2.2)
- User Identification and Authentication (section 3.3.5.2.3)
- User Attribute Retrieval (section 3.3.5.2.4)
- Claim Mapping (section 3.3.5.2.5)
- Resource Access Control (section 3.3.5.2.6)

3.3.5.2.1 Protocol Activation

The protocol is triggered by receipt of a wsignin1.0 response messages.

3.3.5.2.2 Message Validation

The syntax MUST be validated by ensuring that all required parameters are present and contain the correct type of data. For further specifications, see section wsignin1.0 response message (section 2.2.4).<70>

3.3.5.2.3 User Identification and Authentication

The user's identity and related authentication data from the requestor IP/STS are passed in a security token. The resource IP/STS MUST create an Authentication Context record using the appropriate data from the AuthenticationStatement in the security token.

Copyright © 2013 Microsoft Corporation.

3.3.5.2.4 User Attribute Retrieval

A relying party MAY<71> maintain local identities for all users to control access to WS resources. If so, the requestor IP/STS (or the WS resource) MUST retrieve the local identity and use it to replace the AuthIdentity in the user's Authentication Context.

3.3.5.2.5 Claim Mapping

If additional claims are required by a ClaimsIn entry for the requestor IP/STS, the resource IP/STS MUST retrieve them from the AttributeStatement in the security token and store them in the claims field of the user's Authentication Context.

3.3.5.2.6 Resource Access Control

The user's Authentication Context MUST be conveyed to the WS resource that the user originally tried to access. The location of the WS resource MUST be retrieved from the *wctx* parameter in the wsignin1.0 response message. How this Authentication Context is passed to the WS resource is implementation-specific and not addressed in this protocol. Commonly used techniques when the relying party components are located on separate servers include an HTTP cookie, a query string parameter, or a POST body.<72>

3.3.5.3 Outbound wsignout1.0 Request Message Processing

This message is used to request that a requestor IP/STS initiate clean-up operations for cached session state, if any exist, for the user who triggered the request.

3.3.5.3.1 Protocol Activation

The protocol is triggered when a user requests that the user's session be terminated, possibly by clicking a Sign-Out button. The WS resource MUST notify the resource IP/STS to issue a wsignout1.0 request. If the components are located on separate servers, this SHOULD be performed by sending a wsignout1.0 request message to the resource IP/STS using an HTTP 302 redirect.

3.3.5.3.2 Parameter Marshaling

For more information about parameter marshaling, see Appendix A: Windows Behavior.<73>

3.3.5.3.3 Requestor IP/STS Security Realm Discovery

As specified in Relying Party Web Browser Requestor Sessions List (section 3.1.1.5.2), resource IP/STS MAY store a session identifier for the web browser requestor in an HTTP session cookie (for more information, see [RFC2965]). It SHOULD use this identifier and its Inbound Sessions List to look up the correct requestor IP/STS.<74>

3.3.5.3.4 Message Transmission

The resource IP/STS sends a wsignout1.0 Request Message (section 2.2.5) by returning an HTTP 302 response to the web browser requestor with Location set to the URL of the requestor IP/STS. All the query string parameters required for the protocol MUST be set properly, as specified in wsignout1.0 Request Message (section 2.2.5).

Copyright © 2013 Microsoft Corporation.

3.3.5.4 Inbound wsignoutcleanup1.0 Request Message Processing

This message is used to request that a resource IP/STS initiate clean-up operations for cached session state, if any exist, for the user who triggered the request.

3.3.5.4.1 Protocol Activation

The protocol is triggered by receipt of a wsignoutcleanup1.0 message from a requestor IP/STS.

3.3.5.4.2 Clean-Up Processing

A resource IP/STS SHOULD send wsignoutcleanup1.0 messages to WS resources for which security tokens have been issued. Because the protocol does not guarantee a response to a wsignoutcleanup1.0 message, a resource IP/STS SHOULD delete any session state that has been locally cached for the web browser requestor before sending wsignoutcleanup1.0 messages to WS resources.<75>

3.3.5.4.3 Relying Party Security Realm Discovery

As specified in section Relying Party Web Browser Requestor Sessions List (section 3.1.1.5.2), a resource IP/STS MAY store a session identifier for the web browser requestor in an HTTP session cookie (for more information, see [RFC2965]). It SHOULD use this identifier and its Outbound Sessions List to develop the list of WS resources to which it SHOULD send wsignoutcleanup1.0 request messages.<76>

3.3.5.4.4 Message Transmission

The resource IP/STS MUST send a wsignoutcleanup1.0 request message to each WS resource using an explicit HTTP GET method because the protocol does not support chaining wsignoutcleanup1.0 messages using the HTTP 1.1 redirection facilities. How these messages are sent is implementation-specific and is not addressed in this protocol. The resource IP/STS MAY<77> walk the list of WS resources and issue the requests individually.

3.3.5.4.5 Response Message Processing

When clean-up processing is complete, the relying party SHOULD return any relying party specific data (such as a string indicating that clean up is complete) to the web browser requestor. If the *wreply* parameter, as specified in section 2.2.6, was included with the wsignoutcleanup1.0 message, the response SHOULD be sent to the URL specified by *wreply*.<78>

3.3.6 Timer Events

A requestor IP/STS does not need to interact with any timers, or service any timer events, beyond those that may be used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS, or those specified in Timer Events (section 3.1.6).<79>

3.3.7 Other Local Events

A relying party does not have dependencies on local events beyond what is specified in section 3.1.7.

3.4 Web Browser Requestor Details

This section discusses how the web browser requestor is used to transport protocol messages.

Copyright © 2013 Microsoft Corporation.

3.4.1 Abstract Data Model

A web browser requestor does is not required to understand any protocol-specific data for the correct operation of the protocol. It MUST be able to support HTTP query string and POST body parameterization. To provide the best end-user experience, it SHOULD<80> be able to support HTTP cookies (for more information, see [RFC2965]).

3.4.2 Timers

A web browser requestor does not depend on timers beyond those that are used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS, as specified in section 3.1.2.

A web browser requestor is not required to be aware of an implementation's use of timers to determine when the validity intervals of security tokens and Authentication Contexts expire.

3.4.3 Initialization

There is no protocol-specific initialization for a web browser requestor. It simply needs to be ready to perform the standard HTTP 1.1 methods required for accessing WS resources. Specifically, it MUST support HTTP GET and POST methods and properly respond to HTTP 1.1 redirection and error responses.

3.4.4 Higher-Layer Triggered Events

Protocol messages are exchanged between a requestor IP/STS and a relying party. The only function of the web browser requestor with respect to the protocol is to transport these messages. The web browser requestor may be triggered to begin protocol message exchange by receipt of an HTTP/1.1 302 message found that includes a location directive, or an HTTP/1.1 200 OK that includes a form with method set to POST.

3.4.5 Processing Events and Sequencing Rules

A web browser requestor plays a passive role in the operation of the protocol. Its only function is to transport protocol message requests and responses between a requestor IP/STS and one or more relying parties. It is not required to understand the types or content of these protocol messages.

The web browser requestor SHOULD transport all protocol message requests and responses between a requestor IP/STS and a relying party without changing the messages.<81>

3.4.6 Timer Events

A web browser requestor is not required to interact with any timers, or service any timer events, beyond those that are used by the underlying transport to transmit and receive messages over HTTP and SSL/TLS, or those specified in section 3.1.6.

3.4.7 Other Local Events

A web browser requestor does not have dependencies on local events beyond what is specified in section 3.1.7.

Copyright © 2013 Microsoft Corporation.

4 Protocol Examples

This section contains an example scenario for the protocol similar to the scenario specified in Protocol Overview (Synopsis) (section 1.3). This section also includes sample XML sections for the wsignin1.0 response parameters.

4.1 Message Flows

This section describes an example flow of messages among a web browser requestor, a WS resource, a resource IP/STS, and a requestor IP/STS, including example messages.

1. The web browser requestor sends a GET for the WS resource URL to the WS resource.
2. The WS resource returns a 302 Redirect to the resource IP/STS at the resource IP/STS URL.
    1. Query string parameters are appended to the IP/STS URL, that forms a wsignin1.0 request message, as follows:
        1. A query string parameter is added, which indicates that WS resource is the resource wanting authentication (*wreply*).
        2. The original requested WS resource URL is saved as a context parameter in the sign-in message (*wctx*).
    2. Example URL follows.

```
https://adfsresource1.treyresearch.net/adfs/ls/?wa=wsignin1.0&wreply=
https%3a%2f%2fadfsweb1.treyresearch.net%3a8081%2fclaimapp%2f&wct=
2006-07-11T03%3a26%3a39Z&wctx=https%3a%2f%2fadfsweb1.treyresearch.net
%3a8081%2fclaimapp%2fDefault.aspx
```

3. The web browser requestor sends a GET for the resource IP/STS URL.
4. The resource IP/STS returns a 302 Redirect to the requestor IP/STS URL:
    1. Query string parameters are appended to the requestor IP/STS URL, which form a wsignin1.0 request message (section 2.2.3), as follows:
        1. A query string parameter is added, which indicates that the resource IP/STS is the partner wanting authentication (*wtrealm*).
        2. Any state required by the resource IP/STS to continue processing the request is saved as a context parameter in the sign-in message (*wctx*).
    2. Example URL follows.

```
https://adfsaccount1.adatum.com/adfs/ls/auth/integrated/?wa=wsignin1.0
&wtrealm=urn%3afederation%3atreyCrazyResearch&wct=2006-07-11T03%3a28
%3a05Z&wctx=https%3a%2f%2fadfsweb1.treyresearch.net%3a8081%2fclaimapp
%2f%5chttps%3a%2f%2fadfsweb1.treyresearch.net%3a8081%2fclaimapp%2f
Default.aspx
```

5. The web browser requestor sends a GET for the requestor IP/STS URL.
    - Authentication occurs.

Copyright © 2013 Microsoft Corporation.

6. The requestor IP/STS sends a 200 response to the client with a POST Redirect to the resource IP/STS URL:

    1. The returned [HTML] contains a hidden form that contains a wsignin1.0 response and JavaScript, which causes the form to POST immediately (optionally the form may have a visible Submit button). The form's target is the resource IP/STS URL.

    2. The response contains a RequestSecurityTokenResponse message that includes a SAML token whose audience is the resource IP/STS. The token is signed by the requestor IP/STS X.509 certificate.

    3. Example form follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Working...
</title>
</head>
<body>
<form method="POST" action=
"https://adfsresource1.treyresearch.net/adfs/ls/" />
<input type="hidden" name="wa" value="wsignin1.0" />
<input type="hidden" name="wresult" value="
&lt;wst:RequestSecurityTokenResponse xmlns:wst=&quot;
http://schemas.xmlsoap.org/ws/2005/02/trust&quot;>
&lt;wst:RequestedSecurityToken>
&lt;saml:Assertion AssertionID=&quot;
_784067ac-af2c-40b1-993a-cbb376597b6a&quot;
IssueInstant=&quot;2006-07-11T03:15:40Z&quot;
Issuer=&quot;urn:federation:apieceodata&quot;
MajorVersion=&quot;1&quot; MinorVersion=&quot;
1&quot; xmlns:saml=&quot;urn:oasis:names:tc:SAML:1.0:
assertion&quot;>
&lt;saml:Conditions NotBefore=&quot;2006-07-11T03:15:40Z&quot;
 NotOnOrAfter=&quot;2006-07-11T04:15:40Z&quot;>
&lt;saml:AudienceRestrictionCondition>
&lt;saml:Audience>urn:federation:treyCrazyResearch
&lt;/saml:Audience>
&lt;/saml:AudienceRestrictionCondition>
&lt;/saml:Conditions>
&lt;saml:Advice>
&lt;adfs:CookieInfoHash xmlns:adfs=&quot;urn:microsoft:
federation&quot;>11AMDR+AihBUJMPNKS3N64ruuaY=
&lt;/adfs:CookieInfoHash>
&lt;/saml:Advice>
&lt;saml:AuthenticationStatement AuthenticationInstant=&
quot;2006-07-11T03:15:40Z&quot; AuthenticationMethod=
&quot;urn:federation:authentication:windows&quot;>
&lt;saml:Subject>
&lt;saml:NameIdentifier Format=&quot;
http://schemas.xmlsoap.org/claims/UPN&quot;>adamcar@adatum.com
&lt;/saml:NameIdentifier>
&lt;/saml:Subject>
&lt;/saml:AuthenticationStatement>
&lt;saml:AttributeStatement>
```


Copyright © 2013 Microsoft Corporation.

```
&lt;saml:Subject>
&lt;saml:NameIdentifier Format=&quot;
http://schemas.xmlsoap.org/claims/UPN&quot;>adamcar@adatum.com
&lt;/saml:NameIdentifier>
&lt;/saml:Subject>
&lt;saml:Attribute AttributeName=&quot;Group&quot;
AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>ClaimAppMapping
&lt;/saml:AttributeValue>
&lt;/saml:Attribute>
&lt;saml:Attribute AttributeName=&quot;Group&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>TokenAppMapping
&lt;/saml:AttributeValue>
&lt;/saml:Attribute>
&lt;saml:Attribute AttributeName=&quot;Group&quot;
AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>ResearchPlatinum
&lt;/saml:AttributeValue>
&lt;/saml:Attribute>
&lt;saml:Attribute AttributeName=&quot;Group&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>ResearchPurchaser
&lt;/saml:AttributeValue>
&lt;/saml:Attribute>
&lt;saml:Attribute AttributeName=&quot;ResearchFirstName&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>Adam
&lt;/saml:AttributeValue>
&lt;/saml:Attribute>
&lt;/saml:AttributeStatement>
&lt;Signature xmlns=&quot;http://www.w3.org/2000/09/xmldsig#&quot;>
&lt;SignedInfo>
&lt;CanonicalizationMethod Algorithm=&quot;
http://www.w3.org/2001/10/xml-exc-c14n#&quot; />
&lt;SignatureMethod Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#rsa-sha1&quot; />
&lt;Reference URI=&quot;#_784067ac-af2c-40b1-993a-cbb376597b6a&quot;>
&lt;Transforms>
&lt;Transform Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#enveloped-signature&quot; />
&lt;Transform Algorithm=&quot;
http://www.w3.org/2001/10/xml-exc-c14n#&quot; />
&lt;/Transforms>
&lt;DigestMethod Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#sha1&quot; />
&lt;DigestValue>Q4/7YEpc3fTVCzNPop6cyU+VAIE=
&lt;/DigestValue>
&lt;/Reference>
&lt;/SignedInfo>
&lt;SignatureValue>tHgfkoQPiPTXnMWS3K1N6xkL5FOaRzoJIcg0ZqV3CImj
gNtsAGhX8CKCJryYC3ARQdF6jCHHgthnPNyW0jPjA8/VPl1s+Nt5Pe9ODgwhJHx
3wk+gbQs8Ty0IYl+jftp4DM0WF6/LCvIxTmAZ4+GT40fCK9RxbgP4WbR4cj6lE6
C6wwX4odK+lqxVwtR5qx64SUyzPq0zbKG8YX0fSIuSgBJYKLJt+CUk+6YjCeY8m
xH89iL2HWXEBTeMuLh32QrFV2+PFg3jeDcCxCIC9VjwmAyU6VZr3elSpqp/RgtN
Dj8XckSZvwVdOrVztd7sEnJmJmmoeaLpbYNUJWmlbvGsvQ==
&lt;/SignatureValue>
&lt;KeyInfo>
&lt;X509Data>
```


Copyright © 2013 Microsoft Corporation.

```
&lt;X509Certificate&gt;MIIC0jCCAb6gAwIBAgIQKVyguUrraIFEOd4S4lAnOTAJB
gUrDgMCHQUAMCkxJzAlBgNVBAMTHkZlZGVyYXRpb24gU2VydmVyIEFERlNBY2NvdW
50MTAeFw0wNjAxMzEwMjU1MTZaFw0wNzAxMzEwODU1MTZaMCkxJzAlBgNVBAMTHkZ
lZGVyYXRpb24gU2VydmVyIEFERlNBY2NvdW50MTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAL93fgaIJs+E+Y+eYhCrAVHYECQtE43ioEvvljz/4c81eC6i2
Rpiwc+2N+AXWaHqae3vWoCxkTCXWYomm9DXM1JpqTjq0pEWPLjkH7kMa34LUgaKxR
bbmw6Bus9SwGjhCQKdtAwxO9wcp2gEW7FU+gkAtkcoQdv1bEOXwlCInWKXO5jbybT
/l8qGHsz+69fhJdDNoQtjwkIAxqQd9PNdp7r4POG4p5+6kQtkmX3xIWIB0rh0Bml0
9GBbWQUoYbvjx6GH7LB/XJhLIJ7RdIHTv4yS2sTHEgLnF3nx1hjW4cZlznDA7OywV
pZKerK81m7CrHzd78dOJeZYJs2taHUeHY0CAwEAATAJBgUrDgMCHQUAA4IBAQBx9D
BI1t+3/efeQCQVuAwn9yp1nC7CwWGevW8ZyJe5V+3Lx9dQHiO1MpuXpOX+5tEdoDQ
fqjNNvwmA6UM9v1b70CmjWtI/b77scaBAq6B0iQqWlEcyiHKNKjIvF3/ME/SYOL7p
XmY1zvX9EqnOQVtNpLBueeesUo86APAzOqWWWeH5qVwrz73sDoJCToGAtsrnN2b3c
9415u1KSYQNfVQVQl3vCdJXEljuTv0PcGcibgKKT4bEKzHcdkYO38cuuNPgt1a7d0
QnrZ4ZgpjpThLuBLVbzyMP3FiQqFC2hiZ0IKb0uYG5hZY7+wIRhbuYgyqLWsimRL/
aw4m7NdL0RTBO
&lt;/X509Certificate&gt;
&lt;/X509Data&gt;
&lt;/KeyInfo&gt;
&lt;/Signature&gt;
&lt;/saml:Assertion&gt;
&lt;/wst:RequestedSecurityToken&gt;
&lt;wsp:AppliesTo xmlns:wsp=&amp;quot;
http://schemas.xmlsoap.org/ws/2004/09/policy&amp;quot;&gt;
&lt;wsa:EndpointReference xmlns:wsa=&amp;quot;
http://schemas.xmlsoap.org/ws/2004/08/addressing&amp;quot;&gt;
&lt;wsa:Address&gt;urn:federation:treyCrazyResearch
&lt;/wsa:Address&gt;
&lt;/wsa:EndpointReference&gt;
&lt;/wsp:AppliesTo&gt;
&lt;/wst:RequestSecurityTokenResponse&gt;" />
<input type="hidden" name="wctx" value=
"https://adfsweb1.treyresearch.net:8081/claimapp/\
https://adfsweb1.treyresearch.net:8081/claimapp/Default.aspx" />
<noscript>
<p>Script is disabled.  Please click Submit to continue.
</p>
<input type="submit" value="Submit" />
</noscript>
</form>
<script language="javascript">
window.setTimeout('document.forms[0].submit()',0);
</script>
</body>
</html>
```

7. The web browser requestor sends a POST to the resource IP/STS URL.
8. The resource IP/STS sends a 200 response to the client with a POST Redirect to the WS resource URL:
    1. The returned HTML contains a hidden form that contains a wsignin1.0 response and JavaScript, which causes the form to POST immediately (optionally the form may have a visible Submit button). The form's target is the WS resource URL.
    2. The response contains a RequestSecurityTokenResponse message that includes a SAML token whose audience is the WS resource. The token is signed by the resource IP/STS X.509 certificate.

Copyright © 2013 Microsoft Corporation.

3. Full-example HTML form follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head><title>Working...</title></head>
<body>
<form method="POST" action=
"https://adfsweb1.treyresearch.net:8081/claimapp/" />
<input type="hidden" name="wa" value="wsignin1.0" />
<input type="hidden" name="wresult" value="
  &lt;wst:RequestSecurityTokenResponse xmlns:wst=&quot;
http://schemas.xmlsoap.org/ws/2005/02/trust&quot;>
    &lt;wst:RequestedSecurityToken>
      &lt;saml:Assertion AssertionID=&quot;
_f81faa32-fc47-4ddb-98a2-0bda61b7ead2&quot;
IssueInstant=&quot;2006-07-11T03:19:05Z&quot;
Issuer=&quot;urn:federation:treyCrazyResearch&quot;
MajorVersion=&quot;1&quot; MinorVersion=&quot;1&quot;
xmlns:saml=&quot;urn:oasis:names:tc:SAML:1.0:assertion&quot;>
        &lt;saml:Conditions NotBefore=&quot;
2006-07-11T03:19:05Z&quot;NotOnOrAfter=&quot;
2006-07-11T03:20:05Z&quot;>
          &lt;saml:AudienceRestrictionCondition>
            &lt;saml:Audience>
              https://adfsweb1.treyresearch.net:8081/claimapp/
            &lt;/saml:Audience>
          &lt;/saml:AudienceRestrictionCondition>
        &lt;/saml:Conditions>
        &lt;saml:Advice>
          &lt;adfs:ClaimSource xmlns:adfs=&quot;urn:
microsoft:federation&quot;>
            urn:federation:apieceodata
          &lt;/adfs:ClaimSource>
          &lt;adfs:CookieInfoHash xmlns:adfs=&quot;urn:
microsoft:federation&quot;>
            fA4h78BffP5tdaujuJ39y0b4qEo=
          &lt;/adfs:CookieInfoHash>
        &lt;/saml:Advice>
        &lt;saml:AuthenticationStatement AuthenticationInstant=&
quot;2006-07-11T03:15:40Z&quot; AuthenticationMethod=&quot;
urn:federation:authentication:windows&quot;>
          &lt;saml:Subject>
            &lt;saml:NameIdentifier Format=&quot;
http://schemas.xmlsoap.org/claims/UPN&quot;>
              adamcar@adatum.com
            &lt;/saml:NameIdentifier>
          &lt;/saml:Subject>
        &lt;/saml:AuthenticationStatement>
        &lt;saml:AttributeStatement>
          &lt;saml:Subject>
            &lt;saml:NameIdentifier Format=&quot;
http://schemas.xmlsoap.org/claims/UPN&quot;>
              adamcar@adatum.com
            &lt;/saml:NameIdentifier>
          &lt;/saml:Subject>
          &lt;saml:Attribute AttributeName=&quot;Group&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
```


Copyright © 2013 Microsoft Corporation.

```
            &lt;saml:AttributeValue&gt;
              Adatum TokenApp Claim
            &lt;/saml:AttributeValue&gt;
          &lt;/saml:Attribute&gt;
          &lt;saml:Attribute AttributeName=&quot;Group&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;&gt;
            &lt;saml:AttributeValue&gt;
              Adatum ClaimApp Claim
            &lt;/saml:AttributeValue&gt;
          &lt;/saml:Attribute&gt;
          &lt;saml:Attribute AttributeName=&quot;Group&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;&gt;
            &lt;saml:AttributeValue&gt;
              Purchaser
            &lt;/saml:AttributeValue&gt;
          &lt;/saml:Attribute&gt;
          &lt;saml:Attribute AttributeName=&quot;FirstName&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;&gt;
            &lt;saml:AttributeValue&gt;
              Adam
            &lt;/saml:AttributeValue&gt;
          &lt;/saml:Attribute&gt;
        &lt;/saml:AttributeStatement&gt;
        &lt;Signature xmlns=&quot;
http://www.w3.org/2000/09/xmldsig#&quot;&gt;
          &lt;SignedInfo&gt;
            &lt;CanonicalizationMethod Algorithm=&quot;
http://www.w3.org/2001/10/xml-exc-c14n#&quot; /&gt;
            &lt;SignatureMethod Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#rsa-sha1&quot; /&gt;
            &lt;Reference URI=&quot;
#_f81faa32-fc47-4ddb-98a2-0bda61b7ead2&quot;&gt;
              &lt;Transforms&gt;
                &lt;Transform Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#enveloped-signature&quot; /&gt;
                &lt;Transform Algorithm=&quot;
http://www.w3.org/2001/10/xml-exc-c14n#&quot; /&gt;
              &lt;/Transforms&gt;
              &lt;DigestMethod Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#sha1&quot; /&gt;
              &lt;DigestValue&gt;
                iLxgLs5ZLZZePFwiGqrGBddqtUI=
              &lt;/DigestValue&gt;
            &lt;/Reference&gt;
          &lt;/SignedInfo&gt;
          &lt;SignatureValue&gt;SeWQYd9ejm1KGmZoi3wWO3wrFGfvtUfBus
7KtdVUovYlha4ov7BVo3NO8lmou/Fd4+dEHbKmgAMWnEgmGygR2bXfNxJzHUvKf
YKCZoZu/T0tB1QK6mvRnMmcMPtLmmXY1Ckkd8Up2oVf7peFHDo1pPlJPUdloYtb
DwBn8Z2Z
              bjN/ktBH9bFRa7A17QM5RhC0/5HKU8n4fHQOZ3GhwXWtfiy
uTFYxjofG9nBm1ehgKXPD3jfTYrP/gCQf4QwCOtQHDyatBDOs/8gEhaTjO49oN8
E8MuaoyGg8kRV7/+K9H6jYD6NlN8e5mAoXW5x1irTFM4yGkLFr8UfVo8PT3pUBg
g==
          &lt;/SignatureValue&gt;
          &lt;KeyInfo&gt;
            &lt;X509Data&gt;
              &lt;X509Certificate&gt;MIIC1DCCAcCgAwIBAgIQ/EU1/PmUxKt
NHdsEKf/aODAJBgUrDgMCHQUAMCoxKDAmBgNVBAMTH0ZlZGVyYXRpb24gU2VydmVy
IEFERlNSZXNvdXJjZTEwHhcNMDYwMTMxMDI1OTAwWhcNMDcwMTMxMDg1O
```


Copyright © 2013 Microsoft Corporation.

```
TAwWjAqMSgwJgYDVQQDEx9GZWRlcmF0aW9uIFNlcnZlciBBREZTUmVzb3VyY2UxMI
IBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAspXHOLNdt4DP7EOT27sQgiR
ILGW9Bsk+HwcGwiTWnedGeG6IJatjragHdW7MYnLrMyiHY9Jc0e4NwzHPDt4qyXlU
C5c8XuHbr74lc32Jq6vlrRWYA/rxo+inRfxCrBkh5SOVRv4fiSyXya6jvbal1EhVc
sORV6g9iG3xW1Ja1GWOfwXUkfPqqeXCJynG/NHe2HqNqPTFCHOoqciHBNHYXZPEo7
W0C8XBPaVOEQ+1pVGaIubbcTsd3I15uuQGAr5Agce/FquUj4BCiQa8UkcTuuF7mmd
SOajnnNS784/5PK3Nc4CTNYiHPUyjg3r/O23z9BaOVIxVgWuTbLr0i/RmtQIDAQAB
MAkGBSsOAwIdBQADggEBAKLIcrdYRgYTYd/HGQCQDOoDrqAQFYDbU72hvPoc5Jknl
wZu3Jc1V8u5ZszRstBenrnJmNWNTBmJrwZ6xRl1eIYNfpCPY5o3dLK6JMSNgp/oT7
7pn6aYZ5/LpxhF3WGWWBg64/n4pOC5SSDteP1PsCm3LjW6Z9zYOL3mkqui3OLqMBK
cCo+JDMMnKgFDqcxutQ3YMPWDuYX+rvabhxMK2JZgWHhXyhWhn2qzi1z5cCFDA1hK
eMBcqDAMFlSsfbejL5tjapuUkiwAraa4BeTCJYz5/w1aeg2XGeTYOFAWPoJG8vvno
n9R37QIBz/Y8IKdDwZc6zAgTJjLZFhDreZbIvM=
              &lt;/X509Certificate>
            &lt;/X509Data>
          &lt;/KeyInfo>
        &lt;/Signature>
      &lt;/saml:Assertion>
    &lt;/wst:RequestedSecurityToken>
    &lt;wsp:AppliesTo xmlns:wsp=&quot;http://schemas.xmlsoap.org/
ws/2004/09/policy&quot;>
      &lt;wsa:EndpointReference xmlns:wsa=&quot;
http://schemas.xmlsoap.org/ws/2004/08/addressing&quot;>
        &lt;wsa:Address>
          https://adfsweb1.treyresearch.net:8081/claimapp/
        &lt;/wsa:Address>
      &lt;/wsa:EndpointReference>
    &lt;/wsp:AppliesTo>
  &lt;/wst:RequestSecurityTokenResponse>" />
<input type="hidden" name="wctx" value="
https://adfsweb1.treyresearch.net:8081/claimapp/Default.aspx" />
<noscript><p>Script is disabled.
Please click Submit to continue.</p><input type="submit"
value="Submit" /></noscript></form>
<script language="javascript">
window.setTimeout('document.forms[0].submit()',0);</script>
</body>
</html>
```

9. The web browser requestor sends a POST to the WS resource URL.

   - The WS resource validates the SAML token.

10. The WS resource returns a 200 response from the application to the client. This message is an internal implementation detail of the WS-Resource and is mentioned here for completeness only. This message is not necessary for interoperability. The WS-Resource is not restricted by the protocol once it has received the token.

    - The WS resource authorizes a user's request based on attributes from the SAML token.

11. The web browser requestor continues to browse the application at the WS resource, which results in additional traffic.

4.2 XML Examples

This section illustrates some XML examples of data found in the wsignin1.0 response *wresult* parameter.

Copyright © 2013 Microsoft Corporation.

4.2.1 Example RSTR

The following is an example of the shell of a properly constructed RSTR with the optional wsp:AppliesTo element containing sample content.

```
<wst:RequestSecurityTokenResponse>
      <wsp:AppliesTo>
            <wsa:EndpointReference>
                  <wsa:Address>
                                    https://TreyResearch.net/Ordering
                              </wsa:Address>
            </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:RequestedSecurityToken>
            <saml:Assertion> ... </saml:Assertion>
      </wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
```

4.2.2 Example SAML Attribute Element

The following is an example of the SAML Attribute element with sample data. Based on the content, this attribute contains a group claim of Purchasing Agent.

```
<saml:Attribute AttributeName="Group"
      AttributeNamespace="http://schemas.xmlsoap.org/claims">
            <saml:AttributeValue>
            Purchasing Agent
      </saml:AttributeValue>
</saml:Attribute>
```

4.2.3 Using the X509Certificate Element

The following is an example of the KeyInfo element, as specified in [XMLDSig]. This KeyInfo element is using the X509Certificate element to directly include the X.509 certificate, as recommended. For further specifications on the relevant message syntax, see Security Token Signature (section 2.2.4.2.2).

```
<KeyInfo>
    <X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509Certificate>MIIEJDCCA42gAw ... </X509Certificate>
    </X509Data>
</KeyInfo>
```

4.2.4 Using the X509SKI Element

The following is an example of the KeyInfo element, as specified in [XMLDSig]. This KeyInfo element is using the optional X509SKI element to reference the X.509 certificate without copying the entire certificate. For further specifications on the relevant message syntax, see Security Token Signature (section 2.2.4.2.2).

```
<KeyInfo>
    <X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509SKI> 31d97bd7   </X509SKI>
```


Copyright © 2013 Microsoft Corporation.

```
    </X509Data>
</KeyInfo>
```

4.3 Raw Message Examples

This section shows messages that represent sample HTTP encoded wsignin1.0 requests and responses involving a WS resource, resource IP/STS, and requestor IP/STS for one user accessing a WS resource.

4.3.1 Original GET to WS Resource

```
GET /claims/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-shockwave-flash, */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1;
.NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Host: treyws-test
Connection: Keep-Alive
```

4.3.2 HTTP Redirect to Resource IP/STS

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Length: 298
Content-Type: text/html; charset=utf-8
Location: https://treysts-7/adfs/ls/?wa=wsignin1.0&
wreply=https%3a%2f%2ftreyws-test%2fclaims%2f&
wct=2006-07-13T07%3a13%3a22Z&wctx=
https%3a%2f%2ftreyws-test%2fclaims%2fDefault.aspx
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Date: Thu, 13 Jul 2006 07:13:22 GMT

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="https://treysts-7/adfs/ls/?
wa=wsignin1.0&amp;wreply=https%3a%2f%2ftreyws-test%2fclaims%2f&amp;
wct=2006-07-13T07%3a13%3a22Z&amp;wctx=
https%3a%2f%2ftreyws-test%2fclaims%2fDefault.aspx">here</a>.</h2>
</body></html>
```

4.3.3 HTTP GET To Resource IP/STS

```
GET /adfs/ls/?wa=wsignin1.0&wreply=
https%3a%2f%2ftreyws-test%2fclaims%2f&wct=2006-07-13T07%3a13%3a22Z&
wctx=https%3a%2f%2ftreyws-test%2fclaims%2fDefault.aspx HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-shockwave-flash, */*
Accept-Language: en-us
```


Copyright © 2013 Microsoft Corporation.

```
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1;
.NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Connection: Keep-Alive
Host: treysts-7
```

4.3.4 HTTP Redirect to Requestor IP/STS

```
HTTP/1.1 302 Found
Date: Thu, 13 Jul 2006 07:13:22 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Location: https://adatumsts-7/adfs/ls/?wa=
wsignin1.0&wtrealm=urn%3afederation%3atrey+research&wct=
2006-07-13T07%3a13%3a22Z&wctx=https%3a%2f%2ftreyws-test%2fclaims%2f%
5chttps%3a%2f%2ftreyws-test%2fclai
ms%2fDefault.aspx
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 336

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="https://adatumsts-7/adfs/ls/?wa=
wsignin1.0&amp;wtrealm=urn%3afederation%3atrey+research&amp;
wct=2006-07-13T07%3a13%3a22Z&amp;
wctx=https%3a%2f%2ftreyws-test%2fclaims
%2f%5chttps%3a%2f%2ftreyws-test%2fclaims%2fDefault.aspx">
here</a>.</h2>
</body></html>
```

4.3.5 HTTP GET to Requestor IP/STS

```
GET /adfs/ls/?wa=wsignin1.0&wtrealm=urn%3afederation%3atrey+research&
wct=2006-07-13T07%3a13%3a22Z&wctx=https%3a%2f%2ftreyws-test%2fclaims
%2f%5chttps%3a%2f%2ftreyws-test%2fclaims%2fDefault.aspx HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-shockwave-flash, */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1;
 .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Connection: Keep-Alive
Host: adatumsts-7
```

4.3.6 Receive Security Token from Requestor IP/STS in HTML Form

```
HTTP/1.1 200 OK
Date: Thu, 13 Jul 2006 07:13:32 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
WWW-Authenticate: Negotiate
```


Copyright © 2013 Microsoft Corporation.

```
oYGgMIGdoAMKAQChCwYJKoZIgvcSAQICooGIBIGFYIGCBgkqhkiG9xIBAgICAG9zMHG
gAwIBBaEDAgEPomUwY6ADAgEXolwEWvlfVLnxDcKFUiUA+QeYjw1JgxS3Za+jWCdhQC
aUfclDBEozTSd4QWIpU8DI6mTEX/R6nT8z9dE3g6vYJGlam7jdYuYxc1slacV884faP
u7LunNUYOd6U
lp+rw==
X-AspNet-Version: 2.0.50727
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/html; charset=utf-8
Content-Length: 5660

<?xml version="1.0" encoding="utf-8" ?><!DOCTYPE html PUBLIC "-//W3C//
DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd"><html><head><title>Working...</title></head>
<body><form method="POST" action="https://treysts-7/adfs/ls/" />
<input type="hidden" name="wa" value="wsignin1.0" /><input type=
"hidden" name="wresult" value="&lt;wst:RequestSecurityTokenResponse
 xmlns:wst=&quot;http://schemas.xmlsoap.org/ws/2005/02/trust&quot;>
&lt;wst:RequestedSecurityToken>&lt;saml:Assertion AssertionID=&quot;
_efe97b43-7653-4914-ac6e-3058f9746200&quot; IssueInstant=&quot;
2006-07-13T07:13:32Z&quot; Issuer=&quot;urn:federation:adatum&quot;
 MajorVersion=&quot;1&quot; MinorVersion=&quot;1&quot; xmlns:saml=
&quot;urn:oasis:names:tc:SAML:1.0:assertion&quot;>&lt;saml:
Conditions NotBefore=&quot;2006-07-13T07:13:32Z&quot;
NotOnOrAfter=&quot;2006-07-13T08:13:32Z&quot;>&lt;saml:
AudienceRestrictionCondition>&lt;saml:Audience>urn:federation:
trey research&lt;/saml:Audience>&lt;/saml:
AudienceRestrictionCondition>&lt;/saml:Conditions>&lt;saml:
Advice>&lt;adfs:CookieInfoHash xmlns:adfs=&quot;urn:microsoft:
federation&quot;>zA1nNkoHU0Mug3rcyh8ScoWnsKE=&lt;/adfs:
CookieInfoHash>&lt;/saml:Advice>&lt;saml:AuthenticationStatement
 AuthenticationInstant=&quot;2006-07-13T07:13:32Z&quot;
AuthenticationMethod=&quot;urn:federation:authentication:
windows&quot;>&lt;saml:Subject>&lt;saml:NameIdentifier
Format=&quot;http://schemas.xmlsoap.org/claims/UPN&quot;>
Administrator@adatum.com&lt;/saml:NameIdentifier>&lt;/saml:
Subject>&lt;/saml:AuthenticationStatement>&lt;saml:
AttributeStatement>&lt;saml:Subject>&lt;saml:NameIdentifier
Format=&quot;http://schemas.xmlsoap.org/claims/UPN&quot;>
Administrator@adatum.com&lt;/saml:NameIdentifier>&lt;/saml:
Subject>&lt;saml:Attribute AttributeName=&quot;EmailAddress&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>administrator@adatum.com&lt;/saml:
AttributeValue>&lt;/saml:Attribute>&lt;saml:Attribute AttributeName=
&quot;CommonName&quot; AttributeNamespace=&quot;
http://schemas.xmlsoap.org/claims&quot;>&lt;saml:AttributeValue>
Mister Admin&lt;/saml:AttributeValue>&lt;/saml:Attribute>&lt;
saml:Attribute AttributeName=&quot;Group&quot; AttributeNamespace=
&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>ClaimSubmitter&lt;/saml:AttributeValue>
&lt;/saml:Attribute>&lt;saml:Attribute AttributeName=&quot;
Group&quot; AttributeNamespace=&quot;http://schemas.xmlsoap.org/
claims&quot;>&lt;saml:AttributeValue>ClaimApprover&lt;/saml:
AttributeValue>&lt;/saml:Attribute>&lt;/saml:
AttributeStatement>&lt;Signature xmlns=&quot;
http://www.w3.org/2000/09/xmldsig#&quot;>&lt;SignedInfo>
&lt;CanonicalizationMethod Algorithm=&quot;
http://www.w3.org/2001/10/xml-exc-c14n#&quot; />&lt;
```


Copyright © 2013 Microsoft Corporation.

```
SignatureMethod Algorithm=&amp;quot;
http://www.w3.org/2000/09/xmldsig#rsa-sha1&amp;quot; />&amp;lt;
Reference URI=&amp;quot;#_efe97b43-7653-4914-ac6e-3058f9746200&
quot;>&amp;lt;Transforms>&amp;lt;Transform Algorithm=&amp;quot;
http://www.w3.org/2000/09/xmldsig#enveloped-signature&amp;quot; />
&amp;lt;Transform Algorithm=&amp;quot;
http://www.w3.org/2001/10/xml-exc-c14n#&amp;quot; />&amp;lt;/Transforms>
&amp;lt;DigestMethod Algorithm=&amp;quot;
http://www.w3.org/2000/09/xmldsig#sha1&amp;quot; />&amp;lt;
DigestValue>u1ZzBz7qhliMv9/sZ9w2xTlE2UA=&amp;lt;/DigestValue>&amp;lt;
/Reference>&amp;lt;/SignedInfo>&amp;lt;SignatureValue>eG/BzLVoPm2XX9/
0vjWoDPj86NFCVb/BKkWBGkoCDaOAvBFdGJD4gD3kAJtE5lfm0ddvy0T0W0Kqnoid
MBxSQUC/z5a+UgdTqJ68pK/b+M8hrdXms/Tpdq/M+TMaL43sdIb9cGvrHLCCxqg/f
FvQKM1fbKrjjn2u6yZdilVxaDU=&amp;lt;/SignatureValue>&amp;lt;KeyInfo>&amp;lt;X5
09Data>&amp;lt;
X509Certificate>MIID1DCCArygAwIBAgIKG3gmFQAAAAAExTANBgkqhkiG9w0BA
QUFADAZMRcwFQYDVQQDEw5UaGlyZFBhcnR5Um9vdDAeFw0wNjA3MTIyMTAwMDNaFw
0wNzA3MTIyMTEwMDNaMBYxFDASBgNVBAMTC2FkYXR1bXN0cy03MIGfMA0GCSqGSIb
3DQEBAQUAA4GNADCBiQKBgQDSYqm/9eNOX4E72SSg0uq1JW3Mq9tuAKZ8aysPZhlz
MWvqK1951HDzDBtOUJaO+vBYADXJ/zFtT/4KxE3KiZl52PuAppzR8kK01KGLYhhdA
reFBC7mSbh9xSXyAg9ywJiYO+M8o3GZ02uBveAgimDHHJm6gCxcJ1qkR91z8AKFzw
IDAQABo4IBozCCAZ8wDgYDVR0PAQH/BAQDAgTwMEQGCSqGSIb3DQEJDwQ3MDUwDgY
IKoZIhvcNAwICAgCAMA4GCCqGSIb3DQMEAgIAgDAHBgUrDgMCBzAKBggqhkiG9w0D
BzAdBgNVHQ4EFgQUYcR0xltdk4eO1hJ/bYkb7z4b8hIwEwYDVR0lBAwwCgYIKwYBB
QUHAwIwHwYDVR0jBBgwFoAUF4n0HZrhD
VIa5STOYb4GSK5r3OUwVQYDVR0fBE4wTDBKoEigRoZEaHR0cDovL2NlcnRhdXRoLm
Rucy5jb3JwLm1pY3Jvc29mdC5jb20vQ2VydEVucm9sbC9UaGlyZFBhcnR5Um9vdC5
jcmwwgZoGCCsGAQUFBwEBBIGNMIGKMEIGCCsGAQUFBzAChjZodHRwOi8vY2VydGF1
dGgvQ2VydEVucm9sbC9jZXJ0YXV0aF9UaGlyZFBhcnR5Um9vdC5jcnQwRAYIKwYBB
QUHMAKGOGZpbGU6Ly9cXGNlcnRhdXRoXENlcnRfbnJvbGxcY2VydGF1dGhfVGhpcm
RQYXJ0eVJvb3QuY3J0MA0GCSqGSIb3DQEBBQUAA4IBAQB3qi+xNT/c6/mVxd3dCz1
ArkpxJ3bHIro/HAx7n7xGA6vqvTd49WdQt8fvKgEHfBcvI7maWetwFL4RcPm3UmVr
IV/+JkCiHdgf3eCPip36sZt94zVrGFfZFa98/jxo4w1o+WGFUtRtqE6S/gi0LREiM
LCKcRZmh+UL7JHeJWHey953K9cQLM2dGbZJZk/ku1a2CUI7GfPx4dKm1CGnboMcmk
3ifhCMmoRojeSiADcF+s89kbHtffYIITuFdb3SS9kwu2FsA8J3HHm2O/vM8YyDq5x
VnZmP5sREumUFno/nqO502X/GzR63hn6nqoWWr4UcVkkq9FCo8ygEGm6yjefu&amp;lt;
/X509Certificate>&amp;lt;/X509Data>&amp;lt;/KeyInfo>&amp;lt;/Signature>&amp;lt;
/saml:Assertion>&amp;lt;/wst:RequestedSecurityToken>&amp;lt;wsp:AppliesTo
 xmlns:wsp=&amp;quot;http://schemas.xmlsoap.org/ws/2004/09/policy&amp;quot;
>&amp;lt;wsa:EndpointReference xmlns:wsa=&amp;quot;
http://schemas.xmlsoap.org/ws/2004/08/addressing&amp;quot;>&amp;lt;
wsa:Address>urn:federation:trey research&amp;lt;/wsa:Address>&amp;lt;/
wsa:EndpointReference>&amp;lt;/wsp:AppliesTo>&amp;lt;/
wst:RequestSecurityTokenResponse>" /><input type="hidden"
name="wctx" value="https://treyws-test/claims/
\https://treyws-test/claims/Default.aspx" /><noscript>
<p>Script is disabled.  Please click Submit to continue.</p>
<input type="submit" value="Submit" /></noscript></form>
<script language="javascript">
window.setTimeout('document.forms[0].submit()',0);
</script></body></html>
```

4.3.7 HTTP POST Security Token to Resource IP/STS

```
POST /adfs/ls/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
```


Copyright © 2013 Microsoft Corporation.

```
application/msword, application/x-shockwave-flash, */*
Referer: https://adatumsts-7/adfs/ls/auth/integrated/?
wa=wsignin1.0&wtrealm=urn%3afederation%3atrey+research&
wct=2006-07-13T07%3a13%3a22Z&wctx=https%3a%2f%2ftreyws-test%
2fclaims%2f%5chttps%3a%2f%2ftreyws-test%2fclaims%2fDefault.aspx
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1;
.NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Host: treysts-7
Content-Length: 5512
Connection: Keep-Alive
Cache-Control: no-cache

wa=wsignin1.0&wresult=%3Cwst%3ARequestSecurityTokenResponse+xmlns
%3Awst%3D%22http%3A%2F%2Fschemas.xmlsoap.org%2Fws%2F2005%2F02%2
Ftrust%22%3E%3Cwst%3ARequestedSecurityToken%3E%3Csaml%3AAssertion+
AssertionID%3D%22_efe97b43-7653-4914-ac6e-3058f9746200%22+
IssueInstant%3D%222006-07-13T07%3A13%3A32Z%22+Issuer%3D%22urn%3A
federation%3Aadatum%22+MajorVersion%3D%221%22+MinorVersion%3D%
221%22+xmlns%3Asaml%3D%22urn%3Aoasis%3Anames%3Atc%3ASAML%3A1.0%3A
assertion%22%3E%3Csaml%3AConditions+NotBefore%3D%222006-07-13T07
%3A13%3A32Z%22+NotOnOrAfter%3D%222006-07-13T08%3A13%3A32Z%22%3E
%3Csaml%3AAudienceRestrictionCondition%3E%3Csaml%3AAudience%3Eurn
%3Afederation%3Atrey+research%3C%2Fsaml%3AAudience%3E%3C%2Fsaml%3A
AudienceRestrictionCondition%3E%3C%2Fsaml%3AConditions%3E%3Csaml%3A
Advice%3E%3Cadfs%3ACookieInfoHash+xmlns%3Aadfs%3D%22urn%3Amicrosoft
%3Afederation%22%3EzA1nNkoHU0Mug3rcyh8ScoWnsKE%3D%3C%2Fadfs
%3ACookieInfoHash%3E%3C%2Fsaml%3AAdvice%3E%3Csaml
%3AAAuthenticationStatement+AuthenticationInstant%3D%222006-07-13T07
%3A13%3A32Z%22+AuthenticationMethod%3D%22urn%3Afederation
%3Aauthentication%3Awindows%22%3E%3Csaml%3ASubject%3E%3Csaml
%3ANameIdentifier+Format%3D%22http%3A%2F%2Fschemas.xmlsoap.org
%2Fclaims%2FUPN%22%3EAdministrator@adatum.com%3C%2Fsaml
%3ANameIdentifier%3E%3C%2Fsaml%3ASubject%3E%3C%2Fsaml
%3AAAuthenticationStatement%3E%3Csaml%3AAttributeStatement
%3E%3Csaml%3ASubject%3E%3Csaml%3ANameIdentifier+Format%3D%2
2http%3A%2F%2Fschemas.xmlsoap.org%2Fclaims%2FUPN%22%3E
Administrator@adatum.com%3C%2Fsaml%3ANameIdentifier%3E%3C%2F
saml%3ASubject%3E%3Csaml%3AAttribute+AttributeName
%3D%22EmailAddress%22+AttributeNamespace%3D%22
http%3A%2F%2Fschemas.xmlsoap.org%2Fclaims%22%3E%3Csaml%3A
AttributeValue%3Eadministrator@adatum.com%3C%2Fsaml%3AAttributeValue
%3E%3C%2Fsaml%3AAttribute%3E%3Csaml%3AAttribute+AttributeName%3D%22
CommonName%22+AttributeNamespace%3D%22http%3A%2F%2Fschemas.xmlsoap.org
%2Fclaims%22%3E%3Csaml%3AAttributeValue%3EMister+Admin%3C%2Fsaml%3A
AttributeValue%3E%3C%2Fsaml%3AAttribute%3E%3Csaml%3AAttribute+
AttributeName%3D%22Group%22+AttributeNamespace%3D%22http%3A%2F%2F
schemas.xmlsoap.org%2Fclaims%22%3E%3Csaml%3AAttributeValue%3E
ClaimSubmitter%3C%2Fsaml%3AAttributeValue%3E%3C%2Fsaml%3AAttribute
%3E%3Csaml%3AAttribute+AttributeName%3D%22Group%22+AttributeNamespace
%3D%22http%3A%2F%2Fschemas.xmlsoap.org%2Fclaims%22%3E%3Csaml
%3AAttributeValue%3EClaimApprover%3C%2Fsaml%3AAttributeValue%3E
%3C%2Fsaml%3AAttribute%3E%3C%2Fsaml%3AAttributeStatement%3E
%3CSignature+xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2F09
%2Fxmldsig%23%22%3E%3CSignedInfo%3E%3CCanonicalizationMethod+
Algorithm%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2F10%2Fxml-exc-
```


Copyright © 2013 Microsoft Corporation.

```
c14n%23%22+%2F%3E%3CSignatureMethod+Algorithm%3D%22http%3A%2F
%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1%22+%2F%3E%3C
Reference+URI%3D%22%23_efe97b43-7653-4914-ac6e-3058f9746200
%22%3E%3CTransforms%3E%3CTransform+Algorithm%3D%22http%3A%2F
%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23enveloped-signature%22+
%2F%3E%3CTransform+Algorithm%3D%22http%3A%2F%2Fwww.w3.org
%2F2001%2F10%2Fxml-exc-c14n%23%22+%2F%3E%3C%2FTransforms
%3E%3CDigestMethod+Algorithm%3D%22http%3A%2F%2Fwww.w3.org%2
F2000%2F09%2Fxmldsig%23sha1%22+%2F%3E%3CDigestValue%3Eu1ZzB
z7qhliMv9%2FsZ9w2xTlE2UA%3D%3C%2FDigestValue%3E%3C%2FReference%
3E%3C%2FSignedInfo%3E%3CSignatureValue%3EeG%2FBzLVoPm2XX9%2F0vj
WoDPj86NFCVb%2FBKkWBGkoCDaOAvBFdGJD4gD3kAJtE5lfm0ddvy0T0W0Kqnoi
dMBxSQUC%2Fz5a%2BUgdTqJ68pK%2Fb%2BM8hrdXms%2FTpdq%2FM%2BTMaL43s
dIb9cGvrHLCCxqg%2FfFvQKM1fbKrjjn2u6yZdilVxaDU%3D%3C%2F
SignatureValue%3E%3CKeyInfo%3E%3CX509Data%3E%3CX509Certificate
%3EMIID1DCCArygAwIBAgIKG3gmFQAAAAAExTANBgkqhkiG9w0BAQUFADAZMRcw
FQYDVQQDEw5UaGlyZFBhcnR5Um9vdDAeFw0wNjA3MTIyMTAwMDNaFw0wNzA3MTI
yMTEwMDNaMBYxFDASBgNVBAMTC2FkYXR1bXN0cy03MIGfMA0GCSqGSIb3DQEBAQ
UAA4GNADCBiQKBgQDSYqm%2F9eNOX4E72SSg0uq1JW3Mq9tuAKZ8aysPZhlzMWv
qK1951HDzDBtOUJaO%2BvBYADXJ%2FzFtT%2F4KxE3KiZl52PuAppzR8kK01KGL
YhhdAreFBC7mSbh9xSXyAg9ywJiYO%2BM8o3GZ02uBveAgimDHHJm6gCxcJ1qkR
91z8AKFzwIDAQABo4IBozCCAZ8wDgYDVR0PAQH%2FBAQDAgTwMEQGCSqGSIb3DQ
EJDwQ3MDUwDgYIKoZIhvcNAwICAgCAMA4GCCqGSIb3DQMEAgIAgDAHBgUrDgMCB
zAKBggqhkiG9w0DBzAdBgNVHQ4EFgQUYcR0xltdk4eO1hJ%2FbYkb7z4b8hIwEw
YDVR0lBAwwCgYIKwYBBQUHAwIwHwYDVR0jBBgwFoAUF4n0H
ZrhDVIa5STOYb4GSK5r3OUwVQYDVR0fBE4wTDBKoEigRoZEaHR0cDovL2NlcnRh
dXRoLmRucy5jb3JwLm1pY3Jvc29mdC5jb20vQ2VydEVucm9sbC9UaGlyZFBhcnR
5Um9vdC5jcmwwgZoGCCsGAQUFBwEBBIGNMIGKMEIGCCsGAQUFBzAChjZodHRwOi
8vY2VydGF1dGgvQ2VydEVucm9sbC9jZXJ0YXV0aF9UaGlyZFBhcnR5Um9vdC5jc
nQwRAYIKwYBBQUHMAKGOGZpbGU6Ly9cXGNlcnRhdXRoXENlcnRFbnJvbGxcY2Vy
dGF1dGhfVGhpcmRQYXJ0eVJvb3QuY3J0MA0GCSqGSIb3DQEBBQUAA4IBAQB3qi%
2BxNT%2Fc6%2FmVxd3dCz1ArkpxJ3bHIro%2FHAx7n7xGA6vqvTd49WdQt8fvKg
EHfBcvI7maWetwFL4RcPm3UmVrIV%2F%2BJkCiHdgf3eCPip36sZt94zVrGFfZF
a98%2Fjxo4w1o%2BWGFUtRtqE6S%2Fgi0LREiMLCKcRZmh%2BUL7JHeJWHey953
K9cQLM2dGbZJZk%2Fku1a2CUI7GfPx4dKm1CGnboMcmk3ifhCMmoRojeSiADcF%
2Bs89kbHtffYIITuFdb3SS9kwu2FsA8J3HHm2O%2FvM8YyDq5xVnZmP5sREumUF
no%2FnqO502X%2FGzR63hn6nqoWWr4UcVkkq9FCo8ygEGm6yjefu%3C%2FX
509Certificate%3E%3C%2FX509Data%3E%3C%2FKeyInfo%3E%3C%2F
Signature%3E%3C%2Fsaml%3AAssertion%3E%3C%2Fwst%3A
RequestedSecurityToken%3E%3Cwsp%3AAppliesTo+xmlns%3Awsp%3D%22
http%3A%2F%2Fschemas.xmlsoap.org%2Fws%2F2004%2F09%2Fpolicy%22%
3E%3Cwsa%3AEndpointReference+xmlns%3Awsa%3D%22http%3A%2F%2F
schemas.xmlsoap.org%2Fws%2F2004%2F08%2Faddressing%22%3E%3C
wsa%3AAddress%3Eurn%3Afederation%3Atrey+research%3C%2F
wsa%3AAddress%3E%3C%2Fwsa%3AEndpointReference%3E%3C%2F
wsp%3AAppliesTo%3E%3C%2Fwst%3ARequestSecurityTokenResponse
%3E&wctx=https%3A%2F%2Ftreyws-test%2Fclaims%2F%5Chttps
%3A%2F%2Ftreyws-test%2Fclaims%2FDefault.aspx
```

4.3.8 Receive Security Token from Resource IP/STS in HTML Form

```
HTTP/1.1 200 OK
Date: Thu, 13 Jul 2006 07:13:35 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private
```


Copyright © 2013 Microsoft Corporation.

```
Content-Type: text/html; charset=utf-8
Content-Length: 5750

<?xml version="1.0" encoding="utf-8" ?><!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd"><html><head><title>Working...</title>
</head><body><form method="POST" action="https://treyws-test/claims/"
 /><input type="hidden" name="wa" value="wsignin1.0" /><input type=
"hidden" name="wresult" value="&lt;wst:RequestSecurityTokenResponse
xmlns:wst=&quot;http://schemas.xmlsoap.org/ws/2005/02/trust&quot;>
&lt;wst:RequestedSecurityToken>&lt;saml:Assertion AssertionID=&quot;
_cbeb6de4-8ddf-4365-8852-92fff3169910&quot; IssueInstant=&quot;
2006-07-13T07:13:35Z&quot; Issuer=&quot;urn:federation:trey research&
quot; MajorVersion=&quot;1&quot; MinorVersion=&quot;1&quot; xmlns:
saml=&quot;urn:oasis:names:tc:SAML:1.0:assertion&quot;>&lt;saml:
Conditions NotBefore=&quot;2006-07-13T07:13:35Z&quot; NotOnOrAfter=&
quot;2006-07-13T08:13:35Z&quot;>&lt;saml:AudienceRestrictionCondition>
&lt;saml:Audience>https://treyws-test/claims/&lt;/saml:Audience>&lt;/
saml:AudienceRestrictionCondition>&lt;/saml:
Conditions>&lt;saml:Advice>
&lt;adfs:ClaimSource xmlns:adfs=&quot;urn:microsoft:federation&quot;>
urn:federation:adatum&lt;/adfs:ClaimSource>&lt;
adfs:CookieInfoHash xmlns:
adfs=&quot;urn:microsoft:federation&quot;>RcVgPMKduWJ0ISSZqegkFU+cl80=
&lt;/adfs:CookieInfoHash>&lt;/saml:Advice>&lt;saml:
AuthenticationStatement AuthenticationInstant=&quot;2006-07-13T07:
13:32Z&quot; AuthenticationMethod=&quot;urn:federation:authentication:
windows&quot;>&lt;saml:Subject>&lt;saml:NameIdentifier Format=&quot;
http://schemas.xmlsoap.org/claims/UPN&quot;>Administrator@adatum.com
&lt;/saml:NameIdentifier>&lt;/saml:Subject>&lt;/saml:
AuthenticationStatement>&lt;saml:AttributeStatement>&lt;saml:
Subject>&lt;saml:NameIdentifier Format=&quot;
http://schemas.xmlsoap.org/claims/UPN&quot;>Administrator@adatum.com
&lt;/saml:NameIdentifier>&lt;/saml:Subject>&lt;saml:Attribute
AttributeName=&quot;EmailAddress&quot; AttributeNamespace=&quot;
http://schemas.xmlsoap.org/claims&quot;>&lt;saml:AttributeValue>
administrator@adatum.com&lt;/saml:AttributeValue>&lt;/saml:Attribute>
&lt;saml:Attribute AttributeName=&quot;CommonName&quot;
AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>Mister Admin&lt;/saml:AttributeValue>&lt;/
saml:Attribute>&lt;saml:Attribute AttributeName=&quot;Group&quot;
AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>Form Approver&lt;/saml:AttributeValue>&lt;
/saml:Attribute>&lt;saml:Attribute AttributeName=&quot;Group&quot;
 AttributeNamespace=&quot;http://schemas.xmlsoap.org/claims&quot;>
&lt;saml:AttributeValue>Form Submitter&lt;/saml:AttributeValue>&lt;
/saml:Attribute>&lt;/saml:AttributeStatement>&lt;Signature xmlns=
&quot;http://www.w3.org/2000/09/xmldsig#&quot;>&lt;SignedInfo>&lt;
CanonicalizationMethod Algorithm=&quot;
http://www.w3.org/2001/10/xml-exc-c14n#&quot; />&lt;SignatureMethod
 Algorithm=&quot;http://www.w3.org/2000/09/xmldsig#rsa-sha1&quot; />
&lt;Reference URI=&quot;#_cbeb6de4-8ddf-4365-8852-92fff3169910&quot;>
&lt;Transforms>&lt;Transform Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#enveloped-signature&quot; />&lt;
Transform Algorithm=&quot;http://www.w3.org/2001/10/xml-exc-c14n#&
quot; />&lt;/Transforms>&lt;DigestMethod Algorithm=&quot;
http://www.w3.org/2000/09/xmldsig#sha1&quot; />&lt;DigestValue>/
9ui0fsa61HZVJ15iJLstm48sJM=&lt;/DigestValue>&lt;/Reference>&lt;
/SignedInfo>&lt;SignatureValue>dJmjC1SMfVqTtEafP/lljNiKHInSjZByWvI
```


Copyright © 2013 Microsoft Corporation.

```
buJX7ruBPlmKIS2D7Y1qSCPErlP3J+tjtpxW00PRz+1EoYTycK694OjtBZIccaOMWX
duX/ymd0dGZPC2FMHRqKGCkTA6JarxbGmW0+YnL4ZhMvXDzisRO/BApP6rA3Kyt91S
SoMo=&lt;/SignatureValue>&lt;KeyInfo>&lt;X509Data>&lt;
X509Certificate>MIID0jCCArqgAwIBAgIKG4FJ3gAAAAAExzANBgkqhkiG9w0BAQ
UFADAZMRcwFQYDVQQDEw5UaGlyZFBhcnR5Um9vdDAeFw0wNjA3MTIyMTEwMDJaFw0w
NzA3MTIyMTIwMDJaMBQxEjAQBgNVBAMTCXRyZXlzdHMtNzCBnzANBgkqhkiG9w0BAQ
EFAAOBjQAwgYkCgYEAurI6nUCAigHhQ+yPwBPkCeYTFUAsO/F+IyntOz/1HAqT+Nua
G8v4oMn8ryMLTIZ9KUevLMFVO8azyl+tBeiOKMcWLVOJXHOt/nwWcjX5PFncdHQnWJ
+2HhMrIbfRuUuzfb0uLTQKqveXYdfLJpKAay+xQqE6b5foxcVXu6NBJvUCAwEAAaOC
AaMwggGfMA4GA1UdDwEB/wQEAwIE8DBEBgkqhkiG9w0BCQ8ENzA1MA4GCCqGSIb3DQ
MCAgIAgDAOBggqhkiG9w0DBAICAIAwBwYFKw4DAgcwCgYIKoZIhvcNAwcwHQYDVR0O
BBYEFNHRtdtbZ+f6FcOfMpyeQ817ty3cMBMGA1UdJQQMMAoGCCsGAQUFBwMCMB8GA1
UdIwQYMBaAFBeJ9B2a4Q1SGuUkzmG+Bkiua9zlMFUGA1UdHwROMEwwSqBIoEaGRGh0
dHA6Ly9jZXJ0YXV0aC5kbnMuY29yc
C5taWNyb3NvZnQuY29tL0NlcnRFbnJvbGwvVGhpcmRQYXJ0eVJvb3QuY3JsMIGaBgg
rBgEFBQcBAQSBjTCBijBCBggrBgEFBQcwAoY2aHR0cDovL2NlcnRhdXRoL0NlcnRFb
nJvbGwvY2VydGF1dGhfVGhpcmRQYXJ0eVJvb3QuY3J0MEQGCCsGAQUFBzAChjhmaWx
lOi8vXFxjZXJ0YXV0aFxDZXJ0RW5yb2xsXGNlcnRhdXRoX1RoaXJkUGFydHlSb290L
mNydDANBgkqhkiG9w0BAQUFAAOCAQEAXtLcnh5lHJ7XUNUFkR9BNaPy/qArolMojfX
JRXdGuoFCmnRiV/TYVFzSFCV0D9ChPnT4b0MPzXJLaCAelXLhItiWt8gN4gQB1PhGn
J97lJRmUNgxLlueNM24qHJ3bRQBdKIXcaABnL0ICf76CEXUYBXg7mTiQz7rdOUKi/D
fweCeWoeFd7u+tJjrFX0W4fzdiASW9ymQp8jV+JPANCykOROyWxCCvgGmBKRJ+aV/J
9zweaoo18BR11U7TDNfs9Acoi+TejlxFnZwf+4rTXpxkXnIJWnMAcE6kcspqYHDTJP
Yc3p5RhCCeys9clyhkQznN62d7QnuHgTk9RwaPbpj1g==&lt;/X509Certificate>
&lt;/X509Data>&lt;/KeyInfo>&lt;/Signature>&lt;/saml:Assertion>&lt;
/wst:RequestedSecurityToken>&lt;wsp:AppliesTo xmlns:wsp=&quot;
http://schemas.xmlsoap.org/ws/2004/09/policy&quot;>&lt;
wsa:EndpointReference xmlns:wsa=&quot;
http://schemas.xmlsoap.org/ws/2004/08/addressing&quot;>&lt;
wsa:Address>https://treyws-test/claims/&lt;/wsa:Address>&lt;/
wsa:EndpointReference>&lt;/wsp:AppliesTo>&lt;/
wst:RequestSecurityTokenResponse>" /><input type="hidden"
name="wctx" value="https://treyws-test/claims/Default.aspx" />
<noscript><p>Script is disabled.  Please click Submit to continue.
</p><input type="submit" value="Submit" /></noscript></form>
<script language="javascript">
window.setTimeout('document.forms[0].submit()',0);</script>
</body></html>
```

4.3.9 HTTP POST Security Token to WS Resource

```
POST /claims/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-shockwave-flash, */*
Referer: https://treysts-7/adfs/ls/
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1;
.NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Host: treyws-test
Content-Length: 5609
Connection: Keep-Alive
Cache-Control: no-cache
```


Copyright © 2013 Microsoft Corporation.

```
wa=wsignin1.0&wresult=%3Cwst%3ARequestSecurityTokenResponse+xmlns%3A
wst%3D%22http%3A%2F%2Fschemas.xmlsoap.org%2Fws%2F2005%2F02%2Ftrust
%22%3E%3Cwst%3ARequestedSecurityToken%3E%3Csaml%3AAssertion+
AssertionID%3D%22_cbeb6de4-8ddf-4365-8852-92fff3169910%22+
IssueInstant%3D%222006-07-13T07%3A13%3A35Z%22+Issuer%3D%22urn%3A
federation%3Atrey+research%22+MajorVersion%3D%221%22+MinorVersion
%3D%221%22+xmlns%3Asaml%3D%22urn%3Aoasis%3Anames%3Atc%3ASAML%3A
1.0%3Aassertion%22%3E%3Csaml%3AConditions+NotBefore%3D%222006-07
-13T07%3A13%3A35Z%22+NotOnOrAfter%3D%222006-07-13T08%3A13%3A35Z
%22%3E%3Csaml%3AAudienceRestrictionCondition%3E%3Csaml%3AAudience
%3Ehttps%3A%2F%2Ftreyws-test%2Fclaims%2F%3C%2Fsaml%3AAudience%3E%
3C%2Fsaml%3AAudienceRestrictionCondition%3E%3C%2Fsaml%3AConditions%
3E%3Csaml%3AAdvice%3E%3Cadfs%3AClaimSource+xmlns%3Aadfs%3D%22urn%3A
microsoft%3Afederation%22%3Eurn%3Afederation%3Aadatum%3C%2Fadfs%3A
ClaimSource%3E%3Cadfs%3ACookieInfoHash+xmlns%3Aadfs%3D%22urn%3A
microsoft%3Afederation%22%3ERcVgPMKduWJ0ISSZqegkFU%2Bcl80%3D%3C%2F
adfs%3ACookieInfoHash%3E%3C%2Fsaml%3AAdvice%3E%3Csaml%3A
AuthenticationStatement+AuthenticationInstant%3D%
222006-07-13T07%3A13%3A32Z%22+AuthenticationMethod%3D%22urn%3A
federation%3Aauthentication%3Awindows%22%3E%3Csaml%3ASubject%3E
%3Csaml%3ANameIdentifier+Format%3D%22http%3A%2F%2F
schemas.xmlsoap.org%2Fclaims%2FUPN%22%3EAdministrator@adatum.com
%3C%2Fsaml%3ANameIdentifier%3E%3C%2Fsaml%3ASubject%3E%3C%2Fsaml
%3AAuthenticationStatement%3E%3Csaml%3AAttributeStatement%3E%3C
saml%3ASubject%3E%3Csaml%3ANameIdentifier+Format%3D%22http%3A%2F
%2Fschemas.xmlsoap.org%2Fclaims%2FUPN%22%3E
Administrator@adatum.com%3C%2Fsaml%3ANameIdentifier%3E%3C%2F
saml%3ASubject%3E%3Csaml%3AAttribute+AttributeName%3D%22
EmailAddress%22+AttributeNamespace%3D%22http%3A%2F%2F
schemas.xmlsoap.org%2Fclaims%22%3E%3Csaml%3AAttributeValue
%3Eadministrator@adatum.com%3C%2Fsaml%3AAttributeValue
%3E%3C%2Fsaml%3AAttribute%3E%3Csaml%3AAttribute+AttributeName
%3D%22CommonName%22+AttributeNamespace%3D%22http%3A%2F%2F
schemas.xmlsoap.org%2Fclaims%22%3E%3Csaml%3AAttributeValue%3E
Mister+Admin%3C%2Fsaml%3AAttributeValue%3E%3C%2Fsaml%3A
Attribute%3E%3Csaml%3AAttribute+AttributeName%3D%22Group
%22+AttributeNamespace%3D%22http%3A%2F%2Fschemas.xmlsoap.org
%2Fclaims%22%3E%3Csaml%3AAttributeValue%3EForm+Approver%3C%2F
saml%3AAttributeValue%3E%3C%2Fsaml%3AAttribute%3E%3Csaml%3A
Attribute+AttributeName%3D%22Group%22+AttributeNamespace%3D%
22http%3A%2F%2Fschemas.xmlsoap.org%2Fclaims%22%3E%3Csaml%3A
AttributeValue%3EForm+Submitter%3C%2Fsaml%3AAttributeValue
%3E%3C%2Fsaml%3AAttribute%3E%3C%2Fsaml%3AAttributeStatement
%3E%3CSignature+xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2F
09%2Fxmldsig%23%22%3E%3CSignedInfo%3E%3CCanonicalizationMethod
+Algorithm%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2F10%2Fxml-exc-
c14n%23%22+%2F%3E%3CSignatureMethod+Algorithm%3D%22http%3A%2F%
2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1%22+%2F%3E%3C
Reference+URI%3D%22%23_cbeb6de4-8ddf-4365-8852-92fff3169910%
22%3E%3CTransforms%3E%3CTransform+Algorithm%3D%22http%3A%2F%2
Fwww.w3.org%2F2000%2F09%2Fxmldsig%23enveloped-signature%22+
%2F%3E%3CTransform+Algorithm%3D%22http%3A%2F%2Fwww.w3.org%2F
2001%2F10%2Fxml-exc-c14n%23%22+%2F%3E%3C%2FTransforms%3E%3C
DigestMethod+Algorithm%3D%22http%3A%2F%2Fwww.w3.org%2F2000
%2F09%2Fxmldsig%23sha1%22+%2F%3E%3CDigestValue%3E%2F9ui0fsa
61HZVJ15iJLstm48sJM%3D%3C%2FDigestValue%3E%3C%2FReference
%3E%3C%2FSignedInfo%3E%3CSignatureValue%3EdJmjC1SMfVqTtEaf
P%2FlljNiKHInSjZByWvIbuJX7ruBPlmKIS2D7Y1qSCPErlP3J%2BtjtpxW
00PRz%2B1EoYTycK694OjtBZIccaOMWXduX%2Fymd0dGZPC2FMHRqKGCkTA
```


Copyright © 2013 Microsoft Corporation.

```
6JarxbGmW0%2BYnL4ZhMvXDzisRO%2FBApP6rA3Kyt91SSoMo%3D%3C%2F
SignatureValue%3E%3CKeyInfo%3E%3CX509Data%3E%3CX509
Certificate%3EMIID0jCCArqgAwIBAgIKG4FJ3gAAAAAExzANBgkqhkiG9
w0BAQUFADAZMRcwFQYDVQQDEw5UaGlyZFBhcnR5Um9vdDAeFw0wNjA3MTIy
MTEwMDJaFw0wNzA3MTIyMTIwMDJaMBQxEjAQBgNVBAMTCXRyZXlzdHMtNzC
BnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAurI6nUCAigHhQ%2ByPwBP
kCeYTFUAsO%2FF%2BIyntOz%2F1HAqT%2BNuaG8v4oMn8ryMLTIZ9KUevLM
FVO8azyl%2BtBeiOKMcWLVOJXHOt%2FnwWcjX5PFncdHQnWJ%2B2HhMrIbf
RuUuzfb0uLTQKqveXYdfLJpKAay%2BxQqE6b5foxcVXu6NBJvUCAwEAAaOC
AaMwggGfMA4GA1UdDwEB%2FwQEAwIE8DBEBgkqhkiG9w0BCQ8ENzA1MA4GC
CqGSIb3DQMCAgIAgDAOBggqhkiG9w0DBAICAIAwBwYFKw4DAgcwCgYIKoZI
hvcNAwcwHQYDVR0OBBYEFNHRtdtbZ%2Bf6FcOfMpyeQ817ty3cMBMGA1UdJ
QQMMAoGCCsGAQUFBwMCMB8GA1UdIwQYMBaAFBeJ9B2a4Q1SGuUkzmG%2BBk
iua9zlMFUGA1UdHwROMEwwSqBIoEaGRGh0dHA6Ly9jZXJ0YXV0aC5kb
nMuY29ycC5taWNyb3NvZnQuY29tL0NlcnRfbnJvbGwvVGhpcmRQYXJ0eVJv
b3QuY3JsMIGaBggrBgEFBQcBAQSBjTCBijBCBggrBgEFBQcwAoY2aHR0cDo
vL2NlcnRhdXRoL0NlcnRfbnJvbGwvY2VydGF1dGhfVGhpcmRQYXJ0eVJvb3
QuY3J0MEQGCCsGAQUFBzAChjhmaWxlOi8vXFxjZXJ0YXV0aFxDZXJ0RW5yb
2xsXGNlcnRhdXRoX1RoaXJkUGFydHlSb290LmNydDANBgkqhkiG9w0BAQUF
AAOCAQEAXtLcnh5lHJ7XUNUFkR9BNaPy%2FqArolMojfXJRXdGuoFCmnRiV
%2FTYVFzSFCV0D9ChPnT4b0MPzXJLaCAelXLhItiWt8gN4gQB1PhGnJ97lJ
RmUNgxLlueNM24qHJ3bRQBdKIXcaABnL0ICf76CEXUYBXg7mTiQz7rdOUKi
%2FDfweCeWoeFd7u%2BtJjrFX0W4fzdiASW9ymQp8jV%2BJPANCykOROyWx
CCvgGmBKRJ%2BaV%2FJ9zweaoo18BR11U7TDNfs9Acoi%2BTejlxFnZwf%2
B4rTXpxkXnIJWnMAcE6kcspqYHDTJPYc3p5RhCCeys9clyhkQznN62d7Qnu
HgTk9RwaPbpj1g%3D%3D%3C%2FX509Certificate%3E%3C%2FX509Data%
3E%3C%2FKeyInfo%3E%3C%2FSignature%3E%3C%2Fsaml%3AAssertion%
3E%3C%2Fwst%3ARequestedSecurityToken%3E%3Cwsp%3AAppliesTo+
xmlns%3Awsp%3D%22http%3A%2F%2Fschemas.xmlsoap.org%2Fws%2F
2004%2F09%2Fpolicy%22%3E%3Cwsa%3AEndpointReference+xmlns
%3Awsa%3D%22http%3A%2F%2Fschemas.xmlsoap.org%2Fws%2F2004
%2F08%2Faddressing%22%3E%3Cwsa%3AAddress%3Ehttps%3A%2F
%2Ftreyws-test%2Fclaims%2F%3C%2Fwsa%3AAddress%3E%3C%2F
wsa%3AEndpointReference%3E%3C%2Fwsp%3AAppliesTo%3E%3C
%2Fwst%3ARequestSecurityTokenResponse%3E&wctx=https
%3A%2F%2Ftreyws-test%2Fclaims%2FDefault.aspx
```

4.3.10 Final HTTP 200 OK Response from WS Resource

This message is an internal implementation detail of the WS-Resource and is shown here for completeness only. This message is not necessary for interoperability. The WS-Resource is not restricted by the protocol once it has received the token.

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 10211
Content-Type: text/html; charset=utf-8
Expires: -1
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Date: Thu, 13 Jul 2006 07:13:35 GMT

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
```


Copyright © 2013 Microsoft Corporation.

```
...
</html>
```


Copyright © 2013 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

The security considerations specified in [WSFedPRP] section 8 also apply to the Microsoft Web Browser Federated Sign-On Protocol. Specific concerns are highlighted in this section.

5.1.1 Security Token Integrity

The integrity of a security token is compromised when the security token is modified. A digital signature on the **digest** of a security token enables the recipient of the security token to determine whether or not the security token has been modified since it was signed. The Microsoft Web Browser Federated Sign-On Protocol uses digital signatures to secure the integrity of the security token contained in the wsiginin1.0 response message (section 2.2.4) while in transit. The IP/STS signs the security token that it issues, allowing the relying party to check for changes that might have occurred in transit. The strength of the digital signature depends on the signature algorithm used and the key sizes involved.<82>

5.1.2 Certificate Validation

When X.509 [X509] certificates are used, relying parties SHOULD<83> validate the X.509 certificate that corresponds to the key used to sign the security token. X.509 certificates may expire, may be revoked, or may not be issued by a trusted source. The steps required to validate X.509 certificates and to check the revocation status of an X.509 certificate are specified in [RFC3280]. Implementers should pay special attention to [RFC3280] section 9 for security considerations involving usage of X.509 certificates.

5.1.3 Confidentiality

Security tokens contain information about users and may contain sensitive data. The Microsoft Web Browser Federated Sign-On Protocol requires the use of SSL/TLS, as specified in Transport (section 2.1). The use of SSL/TLS prevents the exposure of user information outside the services participating in the protocol, as well as helping to prevent replay attack.<84>

5.1.4 Replay Attack

Specification [WSFedPRP] section 8 specifies that security tokens may be replayed. SSL/TLS is the primary defense against replay, but implementers should also understand that appropriate settings for the validity period of the token help to constrain the time that a security token may be replayed.<85>

5.1.5 Privacy

Privacy is a concern whenever the transmission of user information occurs. Specification [WSFedPRP] section 8 addresses some of the privacy issues, such as obtaining user permission for transmission of user data. The confidentiality measures specified in Confidentiality (section 5.1.3) also address some privacy concerns. Another privacy issue relevant to **single sign on** across security realms is the correlation of user information by relying parties using a common identifier. Because this protocol does not require the same identifier to be issued to every relying party, implementers have the option of implementing configurable behavior for the transmission of user identifiers to relying parties.<86>

Copyright © 2013 Microsoft Corporation.

5.1.6 Identifiers

Claim (section 3.1.1.4) specifies the use of UPN and EmailAddress identifiers for users. The relying party will depend on the identifier being unique so collisions should be avoided. Collisions may be avoided by configuring a relying party to only accept a specific set of suffix **domain naming service (DNS) names** used in the UPN or EmailAddress claim of the security token issued by a security realm's IP/STS. This prevents a malicious IP/STS from enabling its users to impersonate users from another IP/STS.<87>

5.1.7 Cookies

For more information on the HTTP cookie state management mechanism, see [RFC2965]. Cookies may be used to store state information about a user in the user's web browser requestor and to optimize the user experience. Because cookies may be used to identify a user's session and to store user information, special care should be taken with the use of cookies. As specified in [WSFedPRP] section 8, all cookies SHOULD<88> be set as secure.

5.2 Index of Security Parameters

Security parameter	Section
wauth	wsignin1.0 request message
wresult	wsignin1.0 response message

Copyright © 2013 Microsoft Corporation.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 R2 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.6: All Windows behavior documented in this specification applies only to Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

<2> Section 1.8: Windows uses the protocol extensions specified in [MS-MWBE]. For further specifications on these extensions, see [MS-MWBE].

<3> Section 1.8: Windows uses the existing extensibility points specified in [WSFedPRP] as protocol extensions, as specified in [MS-MWBE]. For details on these extensions, see [MS-MWBE].

<4> Section 2.1: Except in the cases specified in [MS-MWBE] section 3.2.5.1.1, Windows uses POST methods to transmit wsignin1.0 responses to relying parties.

<5> Section 2.1: Windows requires use of the HTTPS URL scheme to identify each IP/STS and WS resource participating in the protocol.

<6> Section 2.1: Requests for wsignout1.0 and wsignoutcleanup1.0 are not authenticated by Windows.

Copyright © 2013 Microsoft Corporation.

<7> Section 2.2: These messages are not supported. For the protocol behavior of Windows for such messages, see section 3.1.5.1.

<8> Section 2.2: Parameters not specified in this document are not supported. For the protocol behavior of Windows on nonspecified parameters, see section 3.1.5.1.

<9> Section 2.2.1: These parameters are not supported. For the protocol behavior of Windows on these parameters, see section 3.1.5.1.

<10> Section 2.2.2: This parameter is supported. For the protocol behavior of Windows on this parameter, see sections 3.1.5.4.7 and 3.3.5.1.2.

<11> Section 2.2.2: This parameter is not supported. For the protocol behavior of Windows for this parameter, see section 3.1.5.1.

<12> Section 2.2.3: This parameter is supported by Windows. For the protocol behavior of Windows on this parameter, see sections 3.1.5.4.7 and 3.3.5.1.2.

<13> Section 2.2.3: This parameter is supported by Windows. For the protocol behavior of Windows on this parameter, see section 3.3.5.1.2.

<14> Section 2.2.3: The values supported for this parameter are specified in the following table. For the protocol behavior of Windows on this parameter, see sections 3.1.5.4.2 and 3.3.5.1.2

Method of authentication wanted	wauth URI
User name/password authentication	urn:oasis:names:tc:SAML:1.0:am:password
SSL client authentication	urn:ietf:rfc:2246
Windows integrated authentication	urn:federation:authentication:windows
Multiple factor authentication	http://schemas.microsoft.com/claims/multipleauthn
If the HTTP GET of the wsignin1.0 request message (section 2.2.3) contains an X-MS-Proxy HTTP header, then Windows integrated authentication; otherwise, multiple factor authentication. The X-MS-Proxy HTTP header is defined in [MS-ADFSPIP] section 2.2.1.1.	http://schemas.microsoft.com/claims/wiaormultiauthn

<15> Section 2.2.3: This parameter is conditionally supported. For the supported conditions and processing semantics of Windows for this parameter, see sections 3.1.5.4.2 and 3.3.5.1.2.

<16> Section 2.2.3: The *ClientRequestID* parameter is supported only on Windows Server 2012 R2.

<17> Section 2.2.3: The *login_hint* parameter is supported only on Windows Server 2012 R2.

<18> Section 2.2.3: The *username* parameter is supported only on Windows Server 2012 R2.

<19> Section 2.2.4: This parameter is supported. For the protocol behavior of Windows on this parameter, see section 3.1.5.4.7.

<20> Section 2.2.4.1: The AppliesTo element is supported. For the protocol behavior of Windows for the AppliesTo element, see sections 3.1.5.4.7 and 3.3.5.2.2.

<21> Section 2.2.4.1: Optional elements and attributes are not supported. For the protocol behavior of Windows for optional elements and attributes in the RSTR, see 3.1.5.4.7 and 3.3.5.2.2.

Copyright © 2013 Microsoft Corporation.

<22> Section 2.2.4.2: The Advice element is used by Windows to extend the protocol. These extensions are as specified in [MS-MWBE].

<23> Section 2.2.4.2.1.2: The AttributeStatement is conditionally supported. For more information on the protocol behavior of Windows for AttributeStatements, see sections 3.1.5.4.7 and 3.3.5.2.2.

<24> Section 2.2.4.2.1.2: This namespace is supported. For the protocol behavior of Windows for AttributeNamespace values, see sections 3.1.5.4.7 and 3.3.5.2.2.

<25> Section 2.2.4.2.1.3: The SubjectConfirmation element is not supported. For the protocol behavior of Windows for the SubjectConfirmation element, see sections 3.1.5.4.7 and 3.3.5.2.2.

<26> Section 2.2.4.2.2: Both direct inclusion and SKI are supported for X.509 certificates. For the protocol behavior of Windows for referencing X.509 certificates, see sections 3.1.5.4.7 and 3.3.5.2.2.

<27> Section 2.2.5: The *wreply* parameter is not supported. For the protocol behavior of Windows for this parameter, see sections 3.1.5.4.7 and 3.3.5.1.2.

<28> Section 2.2.6: The *wreply* parameter is not supported. For the protocol behavior of Windows for this parameter, see section 3.3.5.4.2.

<29> Section 3.1.1.2: The requestor IP/STS and relying parties use internal data structures similar to Authentication Context and support all of the fields defined for the abstract data model.

<30> Section 3.1.1.3: This information is exchanged using files that can be generated, or consumed, using export or import capabilities. Also, the information can be typed into a management console.

<31> Section 3.1.1.3: Two separate records are used for a federation partner that is capable of playing both roles.

<32> Section 3.1.1.3: Both configurations are supported. If a requestor IP/STS is configured as a farm of servers, each server can use a different private key, or the same private key can be installed on all servers.

<33> Section 3.1.1.3: Metadata exchanged between federation partners includes claims requested by a relying party. A requestor IP/STS stores this information in local configuration data using the ClaimsOut concept.

<34> Section 3.1.1.3: Metadata exchanged between federation partners includes claims requested by a relying party. A relying party stores this information in local configuration data using the **ClaimsIn** concepts.

<35> Section 3.1.1.3: A requestor IP/STS includes as many claims as possible based on the attributes available for the user that is the subject of the security token. It sends the security token even if not all of the requested claims can be generated.

<36> Section 3.1.1.4: A requestor IP/STS includes optional claims requested by a relying party in the AttributeStatement of the security token issued.

<37> Section 3.1.1.4: Federation partners, performing either the requestor IP/STS or the relying party role, support the EmailAddress, UPN, CommonName, and Group claims, as defined in section 3.1.1.4. Based on local configuration, relying parties interpret Group claims as strings or map them to specific group objects defined in Active Directory.

Copyright © 2013 Microsoft Corporation.

<38> Section 3.1.1.4: Federation partners, performing either the requestor IP/STS or the relying party role, support the Custom claim as defined. Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 will reject any request containing an **AttributeNamespace** value other than http://schemas.xmlsoap.org/claims with an HTTP 1.1 status code 500 server error.

<39> Section 3.1.1.5: HTTP session cookies, as specified in [RFC2965], are used to identify individual web browser requestors.

<40> Section 3.1.1.5.1: An HTTP session cookie, as specified in [RFC2965], is issued to a web browser requestor to manage the list of relying parties that should be sent wsignoutcleanup1.0 messages when the user requests to sign out. The cookie is rewritten for every new security token that is issued for the web browser requestor session.

<41> Section 3.1.1.5.2: An HTTP session cookie, as specified in [RFC2965], is issued to a web browser requestor to manage the list of requestor IP/STSs that should be sent wsignout1.0 messages when the user requests to sign out. The cookie is rewritten for every new security token that is received for the web browser requestor session.

<42> Section 3.1.2: A web browser requestor access request, such as an HTTP GET or POST, for a particular URL managed by a WS resource is rejected if the validity interval of the Authentication Context for the user has expired. Once an access request has been accepted, it will be processed even if the validity interval expires before the response is returned to the web browser requestor.

<43> Section 3.1.2: Timers are not used to determine when validity intervals expire. The NotBefore and NotOnOrAfter values obtained from security tokens and recoded in Authentication Contexts are checked explicitly.

<44> Section 3.1.5.1: Federation partners do not emit query string parameters when sending a protocol message using an HTTP POST. If any query string parameters are received with an HTTP POST, they are ignored.

<45> Section 3.1.5.1: The xml-attribute-request and the xml-pseudonym-request messages described in section 2.2 are not emitted by Windows. Windows Server 2003 returns an HTTP 403 error message. Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 return an HTTP 500 error message if they receive such messages. Any parameters that are not specified in section 2.2 or in the protocol extension specified in [MS-MWBE] are ignored by Windows.

<46> Section 3.1.5.2: An unsupported or unrecognized parameter received with a protocol message is ignored, and the message is processed as if it were not present.

<47> Section 3.1.5.2: SOAP faults are not used. An HTTP 500 error response is returned to the web browser requestor.

<48> Section 3.1.5.3.2: A WS resource stores the application URL in the *wctx* parameter. A resource IP/STS stores the WS resource URL in the *wctx* parameter by prepending it to the incoming value.

<49> Section 3.1.5.3.3: A web browser requestor can send an HTTP GET with a *whr* parameter to WS resource URLs directly. A relying party uses the *whr* parameter for security realm discovery when present. If this parameter is not present, a resource IP/STS displays an HTML form to collect the information from the user.

<50> Section 3.1.5.3.3: A resource IP/STS writes a persistent cookie (for more information, see [RFC2965]) to the web browser requestor to preserve the value of the security realm identifier.

Copyright © 2013 Microsoft Corporation.

<51> Section 3.1.5.4.2: If the *wauth* parameter, which is described in section 2.2.3, is present and is not set to one of the values from the table in that section, Windows returns an HTTP 1.1 status code 500 server error.

<52> Section 3.1.5.4.3: An IP/STS can authenticate a user by means of the Windows implementation of Kerberos (for more information, see [RFC4120]) by collecting the user's Windows identifier and password in an HTML form (using HTTPS) and validating the credentials with Active Directory, or by client SSL authentication using an X.509 certificate issued to the web browser requestor. An IP/STS can also authenticate the user by initiating the Microsoft Web Browser Federated Sign-On Protocol with another STS.

<53> Section 3.1.5.4.4: Attributes are retrieved from either Active Directory or Active Directory Application Mode using authenticated LDAP binds.

<54> Section 3.1.5.4.7: The following Windows behaviors apply for response message processing:

- The *wctx* parameter specified in section 2.2.2 is always returned by Windows with the response if the parameter is present in the request.
- When emitting wsignin1.0 responses, Windows includes the wsp:AppliesTo element.
- When emitting wsignin1.0 response messages, Windows does not include any other optional elements or attributes in the RSTR.
- The presence of the SAML AttributeStatement in the SAML assertion in a wsignin1.0 response message emitted by Windows depends on configuration of the product. Windows supports emitting both no SAML AttributeStatement and one SAML AttributeStatement. Windows includes a SAML AttributeStatement when more than one claim is included in the token. For further specifications, see sections 3.1.1.4 and 3.1.5.
- When emitting wsignin1.0 response messages, Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 use the namespace http://schemas.xmlsoap.org/claims.
- Windows does not specify the SubjectConfirmation element when emitting wsignin1.0 response messages.
- When emitting wsignin1.0 response messages, Windows includes the X.509 certificate directly in the KeyInfo element by default. Windows does not reference X.509 certificates using the X509SKI element by default when emitting wsignin1.0 response messages. Windows can be configured to use the X509SKI element when emitting wsignin1.0 response messages.

<55> Section 3.1.6: Timers are not used to determine when validity intervals expire. The NotBefore and NotOnOrAfter values obtained from security tokens and recoded in Authentication Contexts are checked explicitly.

<56> Section 3.2.5.2.2: An STS, regardless of role, sends wsignoutcleanup1.0 messages to relying parties. An HTTP session cookie (for more information, see [RFC2965]) is issued to a web browser requestor to manage the list of relying parties that should be sent wsignoutcleanup1.0 messages when the user requests to sign out. A requestor IP/STS does not cache any user or web browser requestor session data on a server.

<57> Section 3.2.5.2.3: The *wreply* parameter is not supported for wsignout1.0 request messages, so the web browser requestor is not redirected after wsignoutcleanup1.0 messages are sent.

<58> Section 3.2.5.3.2: An HTTP session cookie (for more information, see [RFC2965]) is issued to a web browser requestor to manage the list of relying parties. A requestor IP/STS sends

Copyright © 2013 Microsoft Corporation.

wsignoutcleanup1.0 messages to all relying parties identified in the session cookie when a user requests to sign-out.

<59> Section 3.2.5.3.3: All of the web browser requestor session state that needs to be deleted is maintained in an HTTP session cookie (for more information, see [RFC2965]). This cookie is deleted as part of the process of sending wsignoutcleanup1.0 messages.

<60> Section 3.2.5.3.4: When a wsignout1.0 message is received, Windows sends a wsignoutcleanup1.0 message to all relying parties identified by the Outbound Sessions List maintained for the web browser requestor. An HTML page that contains a set of IFrames, one for each WS resource, is constructed. In the process of returning this HTML page to the web browser requestor, the HTTP session cookie used to maintain the Outbound Sessions List is deleted. Processing of the iframes in this HTML page by the web browser requestor causes the wsignoutcleanup1.0 messages to be sent in parallel. Windows does not use the *wreply* parameter when emitting wsignoutcleanup1.0 messages.

<61> Section 3.2.6: Timers are not used to determine when validity intervals expire. The NotBefore and NotOnOrAfter values obtained from security tokens (see section 2.2.4.2) and recoded in Authentication Contexts are checked explicitly.

<62> Section 3.3.1: A relying party is implemented as separate resource IP/STS and WS resource components. This factorization is used whether or not the components are deployed on the same server or on different servers.

<63> Section 3.3.1: A relying party is factored into separate resource IP/STS and WS resource components even if the components are located on the same server. Interactions between the resource IP/STS and WS resource are not exposed to requestor IP/STSs in other security realms.

<64> Section 3.3.1.1: The roles a federation partner may perform are indicated in configuration data, as described in section 3.3.1.1. This information is obtained using out-of-band metadata exchange. Files can be exchanged using export or import capabilities, or the data may be typed into a management console.

<65> Section 3.3.1.1: Federation partner URLs are maintained in configuration data. This configuration data is always treated as authoritative versus the value of a **wreply** parameter received in a wsignin1.0 message.

<66> Section 3.3.2: Timers are not used to determine when validity intervals expire. The NotBefore and NotOnOrAfter values obtained from security tokens, as specified in section 2.2.4.2, and recoded in Authentication Contexts are checked explicitly. A resource IP/STS or a WS resource will reject a security token for which the validity interval has expired. A WS resource maintains a user's Authentication Context in an HTTP session cookie (for more information, see [RFC2965]). The cookie is signed to prevent undetected manipulation. The WS resource will return an HTTP 500 error for any request that is accompanied by a session cookie for which the validity interval has expired.

<67> Section 3.3.3: By default, relying parties support CRL checking by means of a CDP. However, if the CDP is not accessible, CRL checking can be disabled to prevent otherwise acceptable security tokens from being rejected.

<68> Section 3.3.5.1.1: A dialect of the Microsoft Web Browser Federated Sign-On Protocol is used, with the WS resource acting as a relying party and the resource IP/STS acting as a requestor IP/STS. The WS resource sends a wsignin1.0 request message to the resource IP/STS to request a security token for the user.

<69> Section 3.3.5.1.2: The following Windows behaviors apply when marshaling parameters for a request message:

Copyright © 2013 Microsoft Corporation.

- When a WS resource sends a wsignin1.0 request message, the *wtrealm* parameter described in Common Syntax for Request Messages (section 2.2.1) cannot be used because the WS resource and the resource IP/STS are located in the same security realm. Because the *wtrealm* parameter identifies a security realm, the *wtrealm* parameter cannot be used to distinguish different federation partners in the same security realms. The *wreply* parameter is used instead, and it is set to the URL of the WS resource.
- Windows also includes the *wctx* parameter specified in Common Syntax for Response Messages (section 2.2.2) and wsignin1.0 request message (section 2.2.3) to preserve any *URL* parameters of the original HTTP request to the relying party.
- Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 always include the *wct* parameter specified in wsignin1.0 request message (section 2.2.3) in the request message.
- By default, Windows does not emit the *wauth* parameter specified in wsignin1.0 request message (section 2.2.3). Windows may be configured to issue the *wauth* parameter and will issue a *wauth* parameter with one of the URIs specified in wsignin1.0 request message (section 2.2.3).
- The *whr* parameter specified in wsignin1.0 request message (section 2.2.3) is not emitted by a relying party unless the *whr* parameter is received on the original request to the Windows relying party and the destination IP/STS is in the same security realm. If a Windows resource IP/STS acting as a relying party receives the *whr* parameter specified in wsignin1.0 request message (section 2.2.3), the wsignin1.0 request message is forwarded to the requestor IP/STS corresponding to the URI value. The *whr* parameter is not included in the request forwarded to the Windows requestor IP/STS. If no requestor IP/STS corresponds to the URI value of the *whr* parameter, Windows ignores the parameter.

<70> Section 3.3.5.2.2: The following Windows behaviors apply when validating wsignin1.0 response messages:

- Windows processes wsp:AppliesTo messages according to the message syntax requirements specified in section 2.2.4.1 when receiving wsignin1.0 responses.
- Windows ignores any other optional elements included in the RSTR when processing received wsignin1.0 response messages.
- Windows supports processing both no SAML AttributeStatement and one SAML AttributeStatement when receiving wsignin1.0 responses.
- Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 reject the security token with an HTTP 1.1 500 error code if a wsignin1.0 response is received with an Attribute element that has an AttributeNamespace value other than http://schemas.xmlsoap.org/claims.
- When processing received wsignin1.0 messages, Windows ignores the SubjectConfirmation element, if present.
- When receiving wsignin1.0 responses, Windows uses the certificate included directly in the X509Certificate element, if present. When receiving wsignin1.0 responses, Windows uses the SKI, if present, to look up the corresponding certificate and continue to process the message. A failed look-up will result in the wsignin1.0 response messages being rejected with an HTTP 1.1 500 server error.
- When receiving wsignin1.0 responses, Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 reject any messages with the **EncryptedData** element described in section 2.2.4.1 with an HTTP 1.1 500 server error.

Copyright © 2013 Microsoft Corporation.

<71> Section 3.3.5.2.4: A user's identity from the Subject element of the security token is mapped to a local Windows security principal identity in Active Directory based on local configuration. In terms of the Authentication Context abstract data model, the local Windows security principal identity replaces the AuthIdentity value derived from the Subject of a security token.

<72> Section 3.3.5.2.6: The Microsoft Web Browser Federated Sign-On Protocol is used with the WS resource acting as a relying party and the resource IP/STS acting as a requestor IP/STS. The resource IP/STS sends a wsignin1.0 response message to the WS resource to issue a security token for the user. There is no change to the protocol message processing, as specified in section Issuing a Security Token by responding to a wsignin1.0 request message (section 3.1.5.4).

<73> Section 3.3.5.3.2: The *wreply* parameter described in section wsignout1.0 Request Message (section 2.2.5) is not supported when emitting wsignout1.0 Request Messages.

<74> Section 3.3.5.3.3: Relying party components, both resource IP/STSs and WS resources, send wsignout1.0 messages to security token services. An HTTP session cookie (for more information, see [RFC2965]) is issued to a web browser requestor to manage the list of security token services that should be sent wsignout1.0 messages when the user requests to sign-out.

<75> Section 3.3.5.4.2: A security token service, regardless of role, sends wsignoutcleanup1.0 messages to relying parties. An HTTP session cookie (for more information, see [RFC2965]) is issued to a web browser requestor to manage the list of relying parties that should be sent wsignoutcleanup1.0 messages when the user requests to sign-out. A resource IP/STS does not cache any user or web browser requestor session data on a server.

<76> Section 3.3.5.4.3: A security token service, regardless of role, sends wsignoutcleanup1.0 request messages to relying parties. An HTTP session cookie (for more information, see [RFC2965]) is issued to a web browser requestor to manage the list of relying parties that should be sent wsignoutcleanup1.0 messages when the user requests to sign-out.

<77> Section 3.3.5.4.4: An HTML page that contains a set of iframes (as specified in [HTML] section 16.5), one for each WS resource, is returned to the web browser requestor. Processing of the iframes by the web browser requestor causes the wsignoutcleanup1.0 messages to be sent in parallel.

<78> Section 3.3.5.4.5: When receiving a wsignoutcleanup1.0 request message (section 2.2.6), the web browser requestor is not redirected to the *wreply* URL after sign-out processing is complete. Windows completes by returning a string indicating that clean up is complete for the security realm, regardless of the presence of *wreply*.

<79> Section 3.3.6: Timers are not used to determine when validity intervals expire. The NotBefore and NotOnOrAfter values obtained from security tokens, as specified in section 2.2.4.2, and recoded in Authentication Contexts are checked explicitly.

<80> Section 3.4.1: Internet Explorer supports the use of session and persistent HTTP cookies (for more information, see [RFC2965]). The Microsoft implementation of this protocol requires that web browser requestors support at least session cookies. It uses persistent cookies to preserve security realm identifiers if they are supported by the web browser requestor.

<81> Section 3.4.5: Internet Explorer in Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2 always passes protocol messages through unaltered. The RMS 2.0 client in Windows Vista SP1, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2 adds a *whr* parameter to the wsignin 1.0 Request Message (section 2.2.3) if the wsignin 1.0 Request Message does not already contain a *whr* parameter.

Copyright © 2013 Microsoft Corporation.

<82> Section 5.1.1: The Windows IP/STS only supports the RSA algorithm (for more information, see [RFC3447]) for signatures, and supports the SHA-1 algorithm (for more information, see [FIPS180]) for calculating digests. The default key length of the Windows RSA keys used for signing security tokens is 2,048 bits.

<83> Section 5.1.2: The certificate validation behavior of Windows is configurable. By default, for certificates used to sign security tokens, Windows ensures that the current time is in the certificate's validity interval, that the certificate has a valid signature, that the certificate is issued by a trusted authority, and that the certificate serial number is not present in the CRL of the issuing authority.

<84> Section 5.1.3: The use of SSL/TLS is required for each Windows IP/STS and WS resource.

<85> Section 5.1.4: The validity period of security tokens issued by the Windows IP/STS is limited to 8 hours by default.

<86> Section 5.1.5: The Windows requestor IP/STS may be configured to issue a different identifier to each resource IP/STS for each user to prevent correlation of user information across multiple relying parties. This behavior is turned off by default.

<87> Section 5.1.6: The Windows resource IP/STS requires that messages from each requestor IP/STS be restricted to only using a specific set of suffix DNS names when UPN or EmailAddress is used as the unique identifier for the user by that requestor IP/STS.

<88> Section 5.1.7: Windows sets cookies as secure.

Copyright © 2013 Microsoft Corporation.

7 Change Tracking

This section identifies changes that were made to the [MS-MWBF] protocol document between the August 2013 and November 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

Copyright © 2013 Microsoft Corporation.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2.1 Normative References	69539 Added [RFC4648] to the list of references.	Y	Content updated.
2.2.2 Common Syntax for Response Messages	69538 Updated reference to [WSTrust] on encoding security tokens as RequestSecurityTokenResponse elements.	Y	Content updated.
2.2.4.1 High-Level Format of wresult Parameter	69538 Updated reference to [WSTrust] on encoding security tokens as RequestSecurityTokenResponse elements.	Y	Content updated.
2.2.4.2.2 Security Token Signature	69539 Added reference to [RFC4648] on the base64-encoded plain value.	Y	Content updated.
3.1.2 Timers	69489 Updated information on the use of the NotBefore and NotOnOrAfter attributes.	Y	Content updated.

Copyright © 2013 Microsoft Corporation.

8 Index

Copyright © 2013 Microsoft Corporation.

L

M

N

O

P

R

Copyright © 2013 Microsoft Corporation.

S

T

U

V

W

X

Copyright © 2013 Microsoft Corporation.