[MS-ADFSOD]:
Active Directory Federation Services (AD FS) Protocols Overview

Intellectual Property Rights Notice for Open Specifications Documentation

This document provides an overview of the Active Directory Federation Services (AD FS) Protocols Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

This document provides an overview of the functionality and relationship of the Active Directory Federation Services (AD FS) protocols. AD FS provides a means for distributed identification, authentication, and authorization across organizational and platform boundaries. It extends the ability to use single sign-on functionality that is normally available only within a single security or enterprise boundary to Internet-facing applications, which gives customers, partners, and suppliers a streamlined user experience while accessing the web-based applications of an organization. The AD FS protocols that are described in this document provide for tasks such as sharing of authorization codes among groups of AD FS servers, obtaining security tokens for users from security token services, and integrating AD FS with preauthentication proxies. These protocols include [MS-ADFSOAL], [MS-ADFSPIP], [MS-ADFSPP], [MS-ADFSWAP], [MS-MWBE], [MS-MWBF], [MS-OAPX], and [MS-SAMLPR], along with various industry standards.

This document describes the intended functionality of the AD FS protocols and how the protocols interact with each other. It provides examples of some common use cases. It does not restate the processing rules and other details that are specific for each protocol. Those details are described in the specifications for the protocols and data structures that belong to this group.

Revision Summary

Date	Revision History	Revision Class	Comments
02/13/2014	1.0	New	Released new document.
05/15/2014	1.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document provides an overview of the protocols that support Active Directory Federated Services (AD FS).

1.1 Glossary

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

Active Directory Domain Services (AD DS)
preauthentication
Hypertext Transfer Protocol (HTTP)
realm (1)
security identifier (SID)
Secure Sockets Layer (SSL)
SOAP
Transport Layer Security (TLS)

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

claim
Web service (WS) resource

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

AD FS farm

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

federation
relying party
security token
security token service (STS)

The following terms are specific to this document:

ADFSOAL: The Active Directory Federation Services OAuth Authorization Code Lookup Protocol [MS-ADFSOAL.

ADFSPIP: The Active Directory Federation Services and Proxy Integration Protocol [MS-ADFSPIP].

ADFSPP: Active Directory Federation Service (AD FS) Proxy Protocol [MS-ADFSPP]

ADFSWAP: Active Directory Federation Service (AD FS) Web Agent Protocol [MS-ADFSWAP].

authorization code: An authorization code as defined in [RFC6749] section 1.3.1.

MWBE: Microsoft Web Browser Federated Sign-On Protocol Extensions [MS-MWBE].

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

OAPX: OAuth 2.0 Protocol Extensions [MS-OAPX].

OAuth: The OAuth 2.0 authorization framework [RFC6749].

SAML1: The Security Assertion Markup Language (SAML) 1.1 [SAMLCore].

SAML2: The Security Assertion Markup Language (SAML) 2.0 [SAMLCore2].

SAMLPR: Security Assertion Markup Language (SAML) Proxy Request Signing Protocol [MS-SAMLPR].

single sign-on: An authentication and authorization scheme in which a user needs only one set of credentials in order to access unrelated network resources.

WS-Federation: The Web Services Federation Language [WSFederation].

WSFedPRP: The WS-Federation: Passive Requestor Profile [WSFedPRP].

WS-Trust: The Web Services Trust Language [WSTrust].

1.2 References

References to Microsoft Open Specification documents 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.

[MS-ADFSOAL] Microsoft Corporation, "Active Directory Federation Services OAuth Authorization Code Lookup Protocol

[MS-ADFSPIP] Microsoft Corporation, "Active Directory Federation Services and Proxy Integration Protocol".

[MS-ADFSPP] Microsoft Corporation, "Active Directory Federation Service (AD FS) Proxy Protocol".

[MS-ADFSWAP] Microsoft Corporation, "Active Directory Federation Service (AD FS) Web Agent Protocol".

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

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

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

[MS-OAPX] Microsoft Corporation, "OAuth 2.0 Protocol Extensions".

[MS-SAMLPR] Microsoft Corporation, "Security Assertion Markup Language (SAML) Proxy Request Signing Protocol".

[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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

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

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012, http://www.rfc-editor.org/rfc/rfc6749.txt

[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

[SAMLCore2] Cantor, S., Kemp, J., Philpott, R., and Maler, E., Eds., "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[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

[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

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

1.3   Overview

Active Directory Federation Services (AD FS) provides a means for distributed identification, authentication, and authorization across organizational and platform boundaries. Put another way, AD FS, and federation in general, decouples the function of identity provider from the function of resource provider.

This decoupling, and its natural extension in **single sign-on** functionality, is normally available only within a single security or enterprise boundary; AD FS extends this ability to Internet-facing applications. This extension gives customers, partners, and suppliers a streamlined user experience while accessing the web-based applications of an organization.

In its most general use, AD FS is about relationships between organizations, although variations on that theme are becoming more common. In this most common scenario, high-level AD FS behavior appears to enable the sharing of identity, authentication, and authorization information between organizations. More accurately, AD FS allows the creation of trust between two organizations: the organization with resources or services that are being accessed, known as the *resource partner*, and the organization that holds the accounts of the clients that are accessing the resources, known as the *account partner*. Once that trust is established, the resource partner does not need to maintain identity information about external clients that are accessing internal resources. Rather, it can depend on the account partner to apply appropriate security policies to the client accounts being held by the account partner.

A major component of AD FS is the **security token service (STS)**. The STS generates and issues **security tokens**, which are used by the account partner to prove the authentication of a client and to express the client's authorization. When a client that is signed into the account partner requests access to a resource or service at the resource partner, the resource partner can expect the account partner to pass on the security token about the client that proves to the resource partner that the client is legitimate and should be granted access to the requested resource or service.

A more complete understanding of AD FS can be gained from the following resources:

- http://www.commoncriteriaportal.org/files/epfiles/ADFS_ST.pdf

- http://social.technet.microsoft.com/wiki/contents/articles/1011.active-directory-federation-services-ad-fs-overview.aspx

- http://technet.microsoft.com/en-in/library/hh831502.aspx

The AD FS protocols that are described in this document provide functionality to support AD FS in a variety of areas. These areas include obtaining **security tokens** for users from an STS, sharing of **authorization code** among groups of AD FS servers, and integrating AD FS with

preauthentication proxies. For more details on the AD FS protocols and how they support AD FS, see section 2.

1.4 Prerequisites/Preconditions

The protocols described in this document are relevant only if AD FS (or an equivalent implementation) is installed on the servers and the servers are configured to hold a federation role.

For authentication, AD FS requires the use of **Active Directory Domain Services** (hereafter called simply "Active Directory").

2 Functional Description

The AD FS protocols that are described in this document provide the following functionality to support AD FS:

- Obtain security tokens for users from a security token service (STS).

- Refine the standard handling of authorization codes and share authorization codes among groups of AD FS servers.

- Integrate AD FS with preauthentication proxies.

Some of the protocols are used between network clients and AD FS servers or between network clients and AD FS proxies. Other protocols are used between AD FS proxies and AD FS servers or among AD FS servers. Details for these protocols and the scenarios and use cases in which they are used are described below and in the sections that follow.

The following diagram shows the fundamental AD FS architecture and how the protocols described in this document fit into that architecture. Note that the diagram purposely shows only general placement of components and protocols, but does not show specific interaction and message flow. These details are included in subsequent sections.
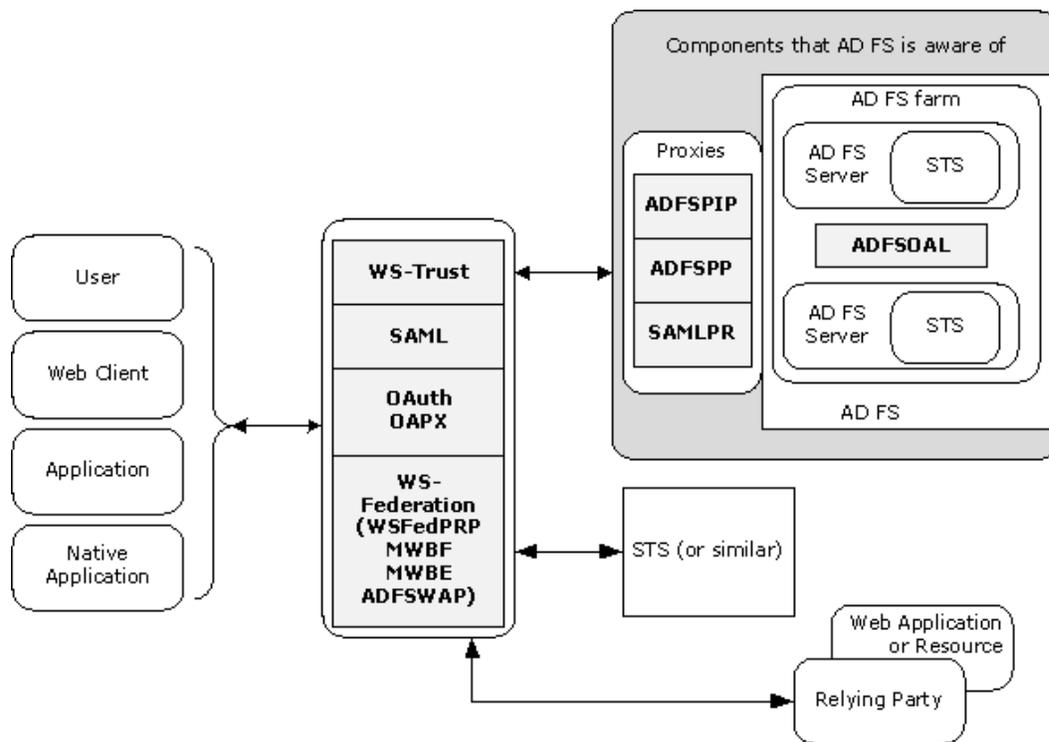
Figure 1: Overview of the AD FS architecture

The following blocks from the diagram are peripheral to AD FS. Note that the entities represented by these blocks are not necessarily even aware that they are interacting specifically with AD FS, but only that they are interacting with a standards compliant identity provider or resource provider.

User, Web Client, Application, Native Application: These are the entities that require use of the protected resource or service.

Web Application, Relying Party: The entity that owns (or at least grants access to) the protected resource or service and that relies on AD FS for proper handling of security requests.

STS (or similar): The entity that takes on the opposite role of AD FS in a security interaction, either identity provider or resource provider.

The following blocks from the diagram make up the relevant parts of AD FS.

AD FS Server: The server on which the AD FS service is running. As shown in the diagram, multiple AD FS servers can be configured to work in an **AD FS farm**. In this case, some type of load balancer is typically required and the AD FS farm as a whole becomes the identity provider.

STS: The security token service (STS) is responsible for the generation, issuance, and maintenance of the security tokens that are used to authenticate and authorize the web clients.

Note The diagram implies that an instantiation of AD FS is either the identity provider or the resource provider. There are network configurations, however, in which AD FS is both the identity provider and the resource provider for the same security exchange. In these cases, the apparent flow of messages and security data can appear to be different from cases where AD FS takes on only one of the roles. Details regarding these differences are detailed in later sections when appropriate.

The remaining blocks from the diagram represent the protocols that are described in this document. The blocks for **WS-Trust**, SAML, **OAuth**, and **WS-Federation** in particular symbolize a sort of separation between AD FS and the other entities in a security exchange, and how AD FS interacts with those entities.

Note The STS is at the heart of AD FS and, as such, is the component that is used as the central focus of the descriptions and explanations in this document. See section 2.2 for the details of the protocols described in this document in terms of the STS.

2.1   Summary of Protocols

The following table provides a list of protocols, algorithms, and structures,<1> grouped according to their primary purpose.

Applicability	Name	Description	Reference
Security token generation	Microsoft Web Browser Federated Sign-On Protocol	Provides for the communication of a requestor's identity and attributes for the purpose of enabling access to a protected **Hypertext Transfer Protocol (HTTP)** [RFC2616] web application or its resources.	[MS-MWBF]
Security token generation	Microsoft Web Browser Federated Sign-On Protocol Extensions	Extends **MWBF** (introduced above) to support scripting for automated form submittal and to enable the passing of **security identifiers (SIDs)** in Security Assertion Markup Language (SAML) 1.1 assertions [SAMLCore].	[MS-MWBE]

Applicability	Name	Description	Reference
Security token generation	Security Assertion Markup Language (SAML) 1.1 and Security Assertion Markup Language (SAML) 2.0	A language that defines XML-encoded assertions (also known as security tokens) about authentication and authorization.	[SAMLCore] and [SAMLCore2]
Security token generation	The Web Services Trust Language	A language to help create security tokens and build trust relationships.	[WSTrust]
Security token generation	Active Directory Federation Service (AD FS) Proxy Protocol	Provides a means for a proxy located outside a protected network to communicate with an STS located inside the protected network.	[MS-ADFSPP]
Security token generation	Security Assertion Markup Language (SAML) Proxy Request Signing Protocol	Used by a proxy located outside a protected network to communicate with an STS located inside the protected network.	[MS-SAMLPR]
Security token generation	Active Directory Federation Service (AD FS) Web Agent Protocol	Enables a **web service (WS) resource** to obtain information about an STS.	[MS-ADFSWAP]
OAuth authorization code processing	The OAuth 2.0 Authorization Framework	Allows the owner of a protected resource or service to grant access to that resource or service.	[RFC6749]
OAuth authorization code processing	Active Directory Federation Services OAuth Authorization Code Lookup Protocol	Allows AD FS servers that are deployed in an AD FS farm configuration to share OAuth authorization codes.	[MS-ADFSOAL]
OAuth authorization code processing	OAuth 2.0 Protocol Extensions	Specifies mandatory extensions to [RFC6749] (the OAuth 2.0 Authorization Framework).	[MS-OAPX]
Integration of AD FS with Preauthentication proxies	Active Directory Federation Services and Proxy Integration Protocol	Implements a proxy that publishes application services that are located inside the boundaries of a corporate network and orchestrates authentication for clients that are outside that boundary (that is, preauthentication).	[MS-ADFSPIP]
single sign-on	The Web Services Federation Language	Mechanisms to allow federation of disparate security **realms** so that existing authentication and authorization information can be leveraged.	[WSFederation]
single sign-on	The WS-Federation: Passive Requestor Profile	A profile to describe how passive requestors such as web browsers can use WS-Federation mechanisms.	[WSFedPRP]

2.2   Components and Capabilities

The primary purpose of this overview document is to describe the protocols that support AD FS. These protocols are useful only within an AD FS environment, primarily (either directly or indirectly) one in which a security token service (STS) is running in order to handle security token concerns. Consequently, the AD FS protocols are best described in terms of an STS, which is done in the sections that follow.

2.2.1   STS Token Generation

The primary purpose of the security token service (STS) is to generate and issue security tokens. Because there are multiple ideas of federation and of what a security token is and how it is consumed, AD FS (and its core STS) must be aware of the most prevalent of those ideas so that it can handle the greatest number of request types that it encounters. As such, AD FS implements several federation frameworks and protocols, all of which are described in the sections that follow.

2.2.1.1   WS-Federation

A user will often need to use several resources or services that are available through the Internet, potentially in different security realms, in the course of a task or a day. One method to obtain access to these resources and services is for the user to sign in to each of the resource and service providers separately, but in doing so, the user is exposing himself or herself to increased security risks, to say nothing of fatigue and irritation. To help alleviate the potential security problems and to provide the user with a more convenient environment, security token services (STSs) can be implemented and deployed such that the user needs to sign in only once, after which all authentication and authorization is taken care of automatically.

- WS-Federation (Web Services Federation Language) and **WSFedPRP** (WS-Federation: Passive Requestor Profile)

  WS-Federation provides the general language and mechanism to connect users and resources across security boundaries, typically in disparate security realms, thereby providing for the creation of a federation of security realms.

  Whereas the specification at [WSFederation] provides for federation language and mechanisms in a broad variety of scenarios, the specification for WS-Federation: Passive Requestor Profile [WSFedPRP] provides more specific details for the scenario in which the requester of security services is passive; that is, the requester is not actively aware of the federation processes that occur. The passive requester is typically a web browser.

- MWBF (Microsoft Web Browser Federated Sign-On Protocol)

  The STS is described in [WSFederation] and [WSFedPRP], but considerations to make the STS more interoperable are implemented in the Microsoft Web Browser Federated Sign-On Protocol, which is described in [MS-MWBF] and its companion documents: [MS-MWBE] and [MS-ADFSWAP]. This increased interoperability is gained by restricting the protocol options and the variations of security tokens that would otherwise be allowed under WSFedPRP.

- **MWBE** (Microsoft Web Browser Federated Sign-On Protocol Extensions)

  A fundamental requirement for WS-Federation-style single sign-on that is defined in [WSFedPRP] and [MS-MWBF] is for the passive requestor, that is, the web browser, to be able to run scripts for form submittal during an HTTP POST (see section 3.1). However, not all web browsers support this scripting. The extensions defined in [MS-MWBE] accommodate older web browsers by providing a means to substitute a single HTTP POST message (as is used in MWBF) with a series of HTTP GET messages with query string parameters.

In addition to support for nonscripting web browsers, the extensions defined in [MS-MWBE] provide an extension to the SAML 1.1 language that allows security identifiers (SIDs) to be passed in SAML 1.1 assertions [SAMLCore]. By enabling this behavior, applications that require SIDs for full authorization can be accommodated.

▪ **ADFSWAP** (Active Directory Federation Service (AD FS) Web Agent Protocol)

The Microsoft Web Browser Federated Sign-On Protocol, given in [MS-MWBF], defines a highly-interoperable form of single sign-on based on WS-Federation. If order for **relying parties** to participate in the MWBF protocol, they need to have additional information about an STS such as URL endpoints, X.509 certificates for security token validation, identifiers for the STS, the security realms that an STS operates in, and the types of **claims** that an STS can emit. The Active Directory Federation Service (AD FS) Web Agent Protocol defined in [MS-ADFSWAP] allows relying parties to obtain this information.

2.2.1.2 WS-Trust

WS-Federation (section 2.2.1.1) operates directly on top of the HTTP protocol [RFC2616] and, specifically for AD FS, only with passive requesters, that is, web browsers. There is a class of applications that is not covered in the WS-Federation scenarios as implemented in AD FS, namely **SOAP**-based applications. These applications can also benefit from **federation** and the use of security tokens for purposes that are similar to those for WS-Federation.

The Web Services Trust Language [WSTrust] is available in AD FS to accommodate SOAP-based applications.

2.2.1.3 OAuth

Note When this overview document references OAuth, it is referring to the OAuth 2.0 Authorization Framework (as opposed to the OAuth 1.0 protocol, which has been deprecated and is not implemented in AD FS).

OAuth is a framework for authorization that is suited to a situation where a user, or *resource owner* in OAuth terms ([RFC6749] section 1.1), owns resources (or data) that are kept on a remote server, or *resource server*, and where the resource owner needs actions to be performed on that data by a client.

Note In OAuth terms, the client is an application or device that is performing actions on the data on behalf of the resource owner.

Traditionally in this type of situation, when the resource owner needed some action to be performed on data that resided on a remote server, the resource owner's credentials (for example, username and password) would need to be passed through the client that would be performing the action. This presented a potential security flaw in that the client now had access to the resource owner's credentials and would need to take certain steps to ensure the integrity and secrecy of those credentials. From the resource owner's perspective, this is a chancy situation. If the resource owner uses only a few clients, the problem of securing credentials and keeping them up-to-date can potentially be managed manually. This solution does not scale, however, in a more realistic scenario where the resource owner makes use of many different clients.

Using OAuth, the resource owner can authenticate with one authoritative source, an *authorization server* in OAuth terms ([RFC6749] section 1.1), and can then refer the client to the authorization server. The authorization server provides to the client the security materials that it needs in order to access the resource owner's data. In this way, the resource owner's credentials are known to only the authorization server, never to the clients. This is a more defensible position for resource owners and a simpler position for clients where they do not need to take on additional responsibility.

There are a number of ways in which OAuth can be implemented to provide the security advantages described above, as described in [RFC6749] section 1.3. AD FS supports the authorization code grant described in [RFC6749] section 1.3 and 4.1.

An STS in AD FS acts as an authorization server. See [RFC6749] section 1.1 for a definition of this role.

- **OAPX** (OAuth 2.0 Protocol Extensions)

  When an OAuth 2.0 client [RFC6749] requests authorization from a resource owner, it is advantageous to include a reference to the actual resource that is needed. In this way, authorization requests can be processed more efficiently. This additional information is specified in the OAuth 2.0 Protocol Extensions [MS-OAPX].

  The primary entities that are affected by these extensions are the OAuth 2.0 client that is requesting access to protected resources and the AD FS server.

- **ADFSOAL** (Active Directory Federation Services OAuth Authorization Code Lookup Protocol)

  An AD FS deployment can consist of just one AD FS server. In larger scenarios, however, it is more likely to have multiple AD FS servers, configured as an AD FS farm, that share the burden and that are placed behind some kind of load balancer.

  In these scenarios, it cannot be assumed that the AD FS server that issues an authorization code to an OAuth 2.0 client [RFC6749] will be the same AD FS server that redeems that code for an access token. As such, a mechanism to communicate information about authorization codes is required among the servers in the AD FS farm. This mechanism is defined in the Active Directory Federation Services OAuth Authorization Code Lookup Protocol [MS-ADFSOAL].

  The primary participants in this protocol are the AD FS servers that are part of an AD FS farm.

2.2.1.4 SAML

Security Assertion Markup Language (SAML) is an XML-based language that provides a means for security entities to communicate security information about a client that is requesting access to protected resources or services. In general, one of the security entities provides a login for the client, typically called the *identity provider* in SAML terms, and the other protects the resource or service, often called the *service provider*. When a request has been received for a resource or service at the service provider, the service provider and identity provider exchange requests for and provisions of security assertions. When the service provider has the security information about the client that it needs, it can make authentication and authorization decisions. See [SAMLCore] and [SAMLCore2].

In AD FS, the STS can take on either or both security-entity roles.

- **SAML1** (Security Assertion Markup Language version 1.1)

  SAML1 was the first SAML specification that was supported by AD FS. AD FS supported only SAML assertions ([SAMLCore] section 2), more generally known as security tokens in this overview document, but not the SAML protocol.

- **SAML2** (Security Assertion Markup Language version 2.0)

  SAML2 is supported more fully in AD FS than SAML1. AD FS supports both the SAML assertions and the SAML protocol of SAML2 ([SAMLCore2] sections 2 and 3).

2.2.2 STS Deployed on the Edge

For security and privacy reasons, a security token service (STS), which is a core part of AD FS, is often deployed behind a firewall. In this scenario, some type of proxy must be placed between the STS and the external network from which the requests for resources, services, or authentication will come. A proxy such as this is configured to interact securely with both the requestor (on the outside) and AD FS (on the inside).

- **ADFSPP** (Active Directory Federation Services Proxy Protocol)

  The Active Directory Federation Services (AD FS) Proxy Protocol [MS-ADFSPP] defines a means for AD FS to communicate with an STS proxy that is on the edge of a protected corporate network. Primarily, ADFSPP is used by the STS proxy to communicate the credentials of a user to the STS inside the protected network for the purpose of generating a security token when that user is participating in WS-Federation-style single sign-on activity, specifically using the MWBF Protocol.

  ADFSPP also provides a means for the STS proxy to obtain configuration data from the STS inside the protected network. This configuration data is useful for users when selecting an acceptable security realm from which to obtain a security token.

- **SAMLPR** (SAML Proxy Request Signing Protocol)

  The Security Assertion Markup Language (SAML) Proxy Request Signing Protocol provides a means for proxy servers to contact an STS server to request SAML signature operations upon messages that are being sent as well as related tasks. This protocol is used when the proxy server needs to perform operations that require knowledge of configured keys and other state information about federated sites known by the STS server.

- **ADFSPIP** (Active Directory Federation Services and Proxy Integration Protocol)

  Rather than allowing a web client to have access into the network behind a firewall to discover services and authenticate for their use, a proxy can be placed outside the firewall. This proxy can then be used to publish the services that are available behind the firewall and also to forward authentication requests to the AD FS server, that is, the STS, that is also behind the firewall. To be successful, the proxy must have a trust relationship with AD FS and the services behind the firewall, and have access to information about what services are available. The Active Directory Federation Services and Proxy Integration Protocol [MS-ADFSPIP] establishes this trust and access.

  The primary participants in this protocol are the web application proxy (WAP) that is outside the firewall and the AD FS server or servers that are behind the firewall.

2.3 Protocol Relationships

The following diagram shows the dependencies and relationships of the AD FS protocols to each other and to industry standards.
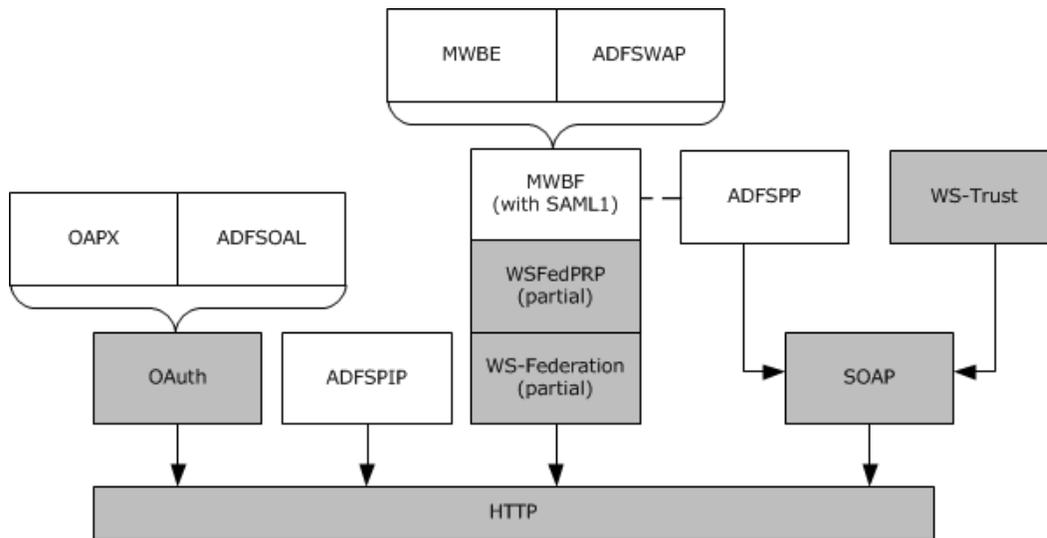
Figure 2: Protocol relationships

2.4 Coherency Requirements

This group of protocols has no special coherency requirements.
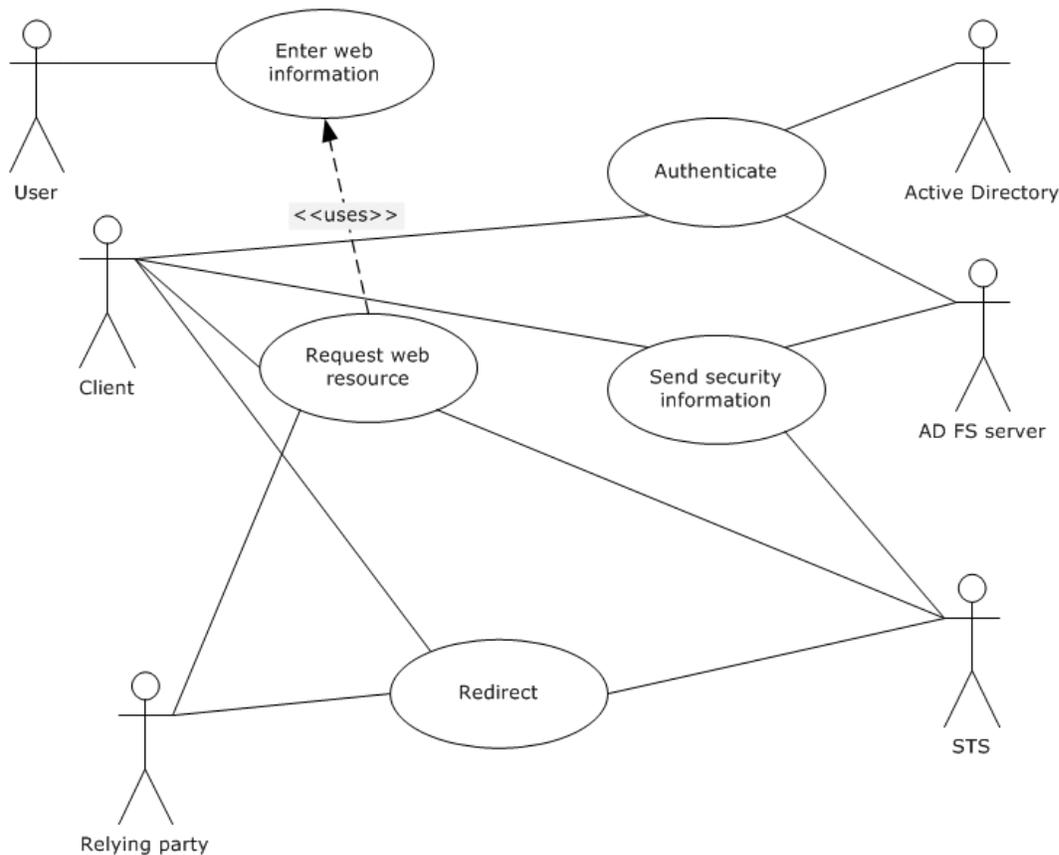
2.5 Security

This group of protocols has no security considerations outside the security considerations of the individual protocols and extensions.

2.6 Additional Considerations

There are no additional considerations.

3 Use Cases

3.1 Single Sign-on Using a Security Token Service and WS-Federation

Figure 3: Use case diagram for single sign-on with WS-Federation

Goal

To obtain a security token for use when signing on to web resources without requiring user intervention.

Context of Use

A user wishes to use a number of resources that are accessible through the Internet, but does not wish to log on to each resource separately. Alternatively, the user wishes to use one resource multiple times, but does not wish to log on to the resource each time it is used. See [WSFederation], [WSFedPRP], and, more specifically, the Microsoft Web Browser Federated Sign-On Protocol [MS-MWBF] for reference.

Actors

- **User**: The person who wishes to use a web resource, which is controlled by the relying party (defined below).

- **Client**: The network entity that the user uses to request the web resource. In the case of WS-Federation, specifically [MS-MWBF], this is a web browser.

- **Relying Party**: The relying part is the network entity that controls access to the web resource, and is typically some type of resource server. It relies on AD FS for security tokens that it can use to make security decisions about the user or client.

- **STS**: For the purpose of this use case, the security token service (STS) handles authentication for the relying party.

- **AD FS Server**: The AD FS server authenticates the user and provides security tokens to the relying party (possibly through the STS that authenticates for the relying party) so that the relying party can make security decisions about the user or client.

  Note The AD FS server is itself an STS, and in some cases, the AD FS server provides authentication for the relying party and also authenticates the user.

- **Active Directory**: Active Directory is the identity store that the AD FS server contacts for authentication information about the user.

Preconditions

A web resource exists that is needed by a user. The resource is protected by an entity, the relying party, that relies on AD FS (directly or indirectly) for authentication of the user.

Main success scenario

1. **Trigger**: The user interacts with the client (that is, a web browser) to connect with a web resource.

2. The client sends a request for the web resource to the appropriate address.

3. The relying party redirects the client to the STS that it uses for authentication.

   Note In this scenario, the user and the relying party are in different security realms. The STSs in the security realms have established a federation partnership.

4. The STS that authenticates for the relying party redirects the client to the STS that authenticates the user, that is, the AD FS server.

5. The AD FS server authenticates the user, either for the first time by querying the user, or for subsequent times by using information that is stored on the client and at the AD FS server. Active Directory supports the AD FS server for this task as needed.

6. The AD FS server creates the security information needed by the STS that authenticates for the relying party and sends the information to the client.

7. The client sends the security information on to the STS that authenticates for the relying party.

8. The STS that authenticates for the relying party creates the security information needed by the relying party, for example, the security token, and sends the information to the client.

9. The client reconnects with the web resource, and includes the security information that was created by the STS that authenticates for the relying party.

Post condition

The user has obtained appropriate access to the web resource.

Variation: Single security realm

In this variation, the user and relying party are in the same security realm.

1-3. Same as the main success scenario except that the user and relying party are in the same security realm; the STS that authenticates for the relying party is also the AD FS server.

4. The AD FS server authenticates the user, either for the first time by querying the user, or for subsequent times by using information that is stored on the client and at the AD FS server. Active Directory supports the AD FS server for this task as needed.

5. The AD FS server creates the security information needed by the relying party, for example, the security token, and sends the information to the client.

6. The client reconnects with the web resource, and includes the security information that was created by the AD FS server.

3.1.1 Success Cases

3.1.1.1 User and Relying Party in Different Realms Example

For this example of single sign-on using a security token service (STS) and WS-Federation, the environment is configured as for the main success scenario in section 3.1 and the user has not yet accessed the web resource in question. The following diagram shows the network entities involved and the sequence of messages that flow between those entities.
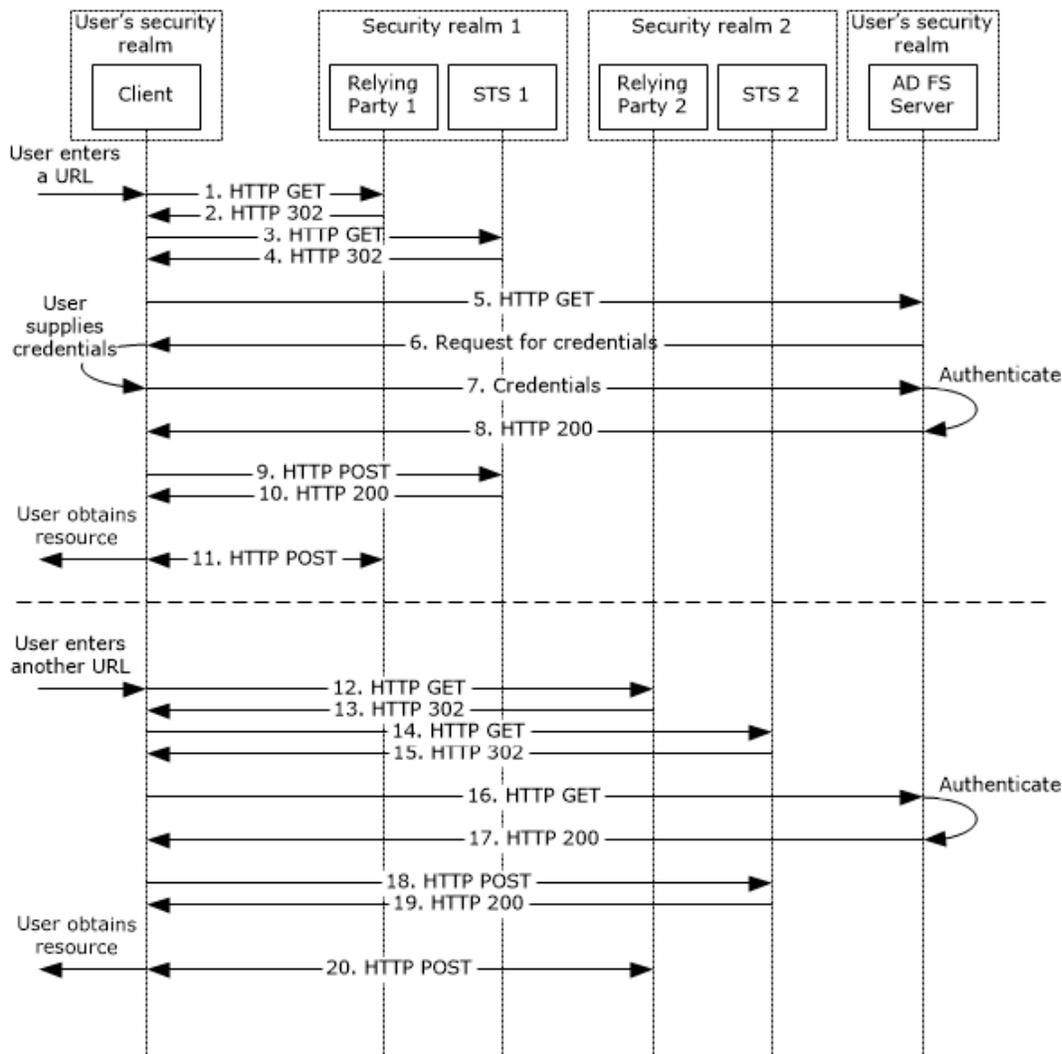
Figure 4: Sequence diagram for user and relying party in different realms

1. When the user requests, via a web browser (see [WSFedPRP] and [MS-MWBF]), access to a web resource, the client sends an HTTP GET to the address of the web resource.

2. The resource server, that is, the relying party, upon ascertaining that the requested resource requires appropriate authentication and authorization, responds to the client with an HTTP 302 response (redirect). The redirect response contains the address of the STS that the relying party relies on for secure access to the web resource, along with other information that is needed by the STS for a wsignin1.0 message.

   See [MS-MWBF] section 4.1, step 2, and [MS-MWBF] section 4.3.2 for examples of the redirect response.

3. The client appends the information from the relying party to an HTTP GET as a query string. The client looks into its cookie cache [RFC2965] to determine whether it has any cookies from this particular STS; for this stage in this example, the client does not have any cookies from this particular STS. The client then sends the GET to the STS given by the relying party.

4. The STS that authenticates for the relying party responds to the client with another HTTP 302 response (redirect). The redirect response contains the address of the STS that authenticates the user, in the case of this example, an AD FS server, along with other information that is needed by the AD FS server for a wsignin1.0 message.

   See [MS-MWBF] section 4.1, step 4, and [MS-MWBF] section 4.3.4 for examples of the redirect response.

5. The client appends the information from the STS to an HTTP GET as a query string. The client looks into its cookie cache [RFC2965] to determine whether it has any cookies from the AD FS server; for this stage in this example, the client does not have any cookies from the AD FS server. The client then sends the GET to the AD FS server.

6. The AD FS server receives the HTTP GET from the client with the query string from the STS that authenticates for the relying party.

   The AD FS server sends messages to the client to interact with the user or the client in order to obtain (either manually or automatically) credential information from the user. In the case of this example, the server sends a forms page to the client so that the user can enter user credentials.

   Note The details of this interaction between the server and client are outside the scope of this document, but typically involve standard GET and POST HTTP communication [RFC2616] and HTTP over **Secure Sockets Layer (SSL)** and **Transport Layer Security (TLS)** [RFC2818], as well as others. In addition, there are other types of interaction that might occur between the AD FS server and the client at this point in the message flow instead of what is described above; for example, a Kerberos request to automatically obtain user credentials might be sent to the client instead of a forms page.

7. The user supplies the required credential information and the client sends the response to the AD FS server.

8. The AD FS server authenticates the user, in the case of this example, by contacting Active Directory. The AD FS server then generates the security material that must be passed on to the STS that authenticates for the relying party and also constructs a cookie that contains information about the user authentication that has occurred. This data is sent to the client along with hidden JavaScript code in an HTTP 200 response.

   See [MS-MWBF] section 4.1, step 6, and [MS-MWBF] section 4.3.6 for examples of the HTTP 200 response.

9. The client executes the hidden JavaScript code, which causes it to send an HTTP POST to the STS that authenticates for the relying party. The POST includes the security material generated by the AD FS server. In addition, the client stores the cookie in its cookie cache.

10. The STS receives the HTTP POST and creates the security token needed by the relying party. This is sent to the client along with hidden JavaScript code in an HTTP 200 response.

    See [MS-MWBF] section 4.1, step 8, and [MS-MWBF] section 4.3.8 for examples of the HTTP 200 response.

11. The client executes the hidden JavaScript code, which causes it to send an HTTP POST to the web resource. The POST includes the security token given by the STS. At this time, the user is able to access the web resource.

12. Sometime in the future, the user sends an HTTP GET, via the client, to another web resource. The relying party for this other web resource relies on a different STS for secure access to its web resource.

13. As in step 2 above, a redirect is sent from this new relying party to the client with the address of the STS that it relies on for authentication.

14. As in step 3 above, the client sends an HTTP POST to this new STS.

15. Step 4 from above is repeated (with data that is relevant for this new STS).

16. Step 5 from above is repeated with the appropriate data. This time, however, when the client sends the HTTP GET to the AD FS server with the query string from the new STS, it also includes the cookie that the AD FS server sent to it when sending the security token in step 8.

17. Having received the HTTP GET, the AD FS server has also received the cookie information that it stored in the client earlier. Having this cookie information, the server is not required to query for user credentials again, but can immediately begin to create the security material as in step 8 above. The generated security information is sent to the client as hidden JavaScript, as described previously.

18. (through 20) The rest of the message flow happens exactly as it did in steps 9 through 11 above, but with the appropriate data for the new STS and relying party. At this time, the user is able to access the second web resource without having to log on a second time.

3.1.1.2 User and Relying Party in Same Realm Example

For this example of single sign-on using a security token service (STS) and WS-Federation, the environment is configured as for the variation scenario in section 3.1 and the user has not yet accessed the web resource in question. The following diagram shows the network entities involved and the sequence of messages that flow between those entities.
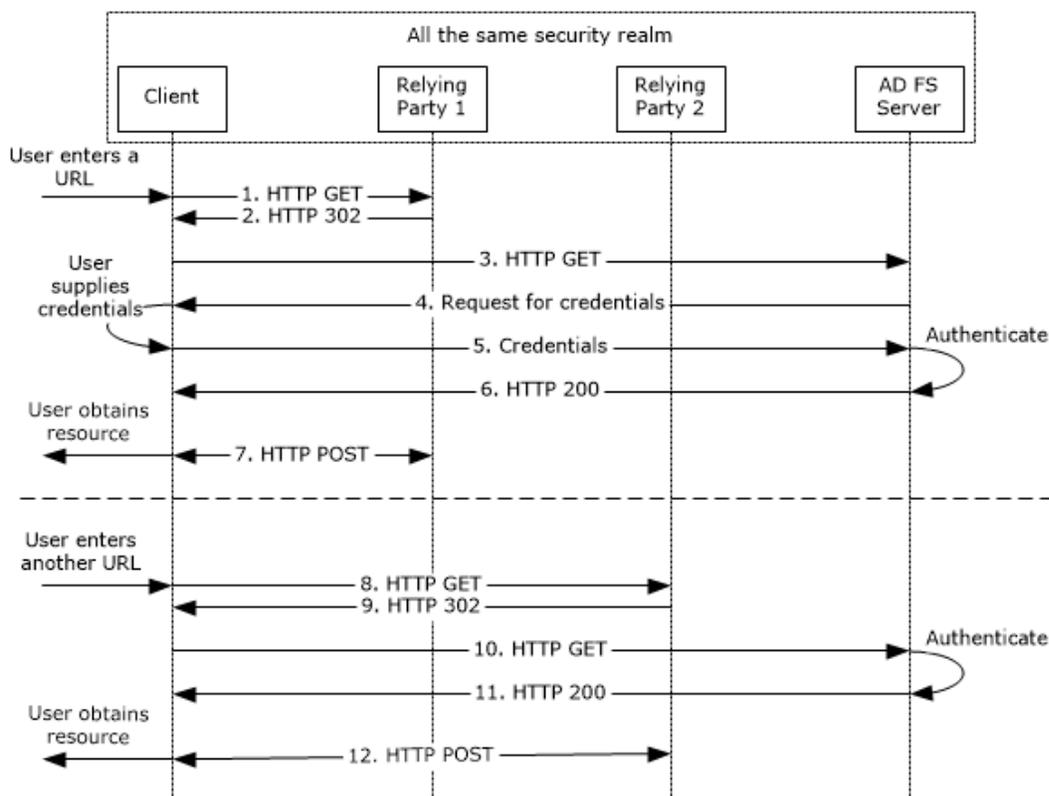
Figure 5: Sequence diagram for user and relying party in same realm

1. When the user requests, via a web browser (see [WSFedPRP] and [MS-MWBF]), access to a Web resource, the client sends an HTTP GET to the address of the web resource.

2. The resource server, that is, the relying party, upon ascertaining that the requested resource requires appropriate authentication and authorization, responds to the client with an HTTP 302 response (redirect). The redirect response contains the address of the STS that the relying party relies on for secure access to the Web resource, along with other information that is needed by the STS for a wsignin1.0 message. In the case of this example, the aforementioned STS is also the AD FS server that authenticates the user.

   See [MS-MWBF] section 4.1, step 2, and [MS-MWBF] section 4.3.2 for examples of the redirect response.

3. The client appends the information from the relying party to an HTTP GET as a query string. The client looks into its cookie cache [RFC2965] to determine whether it has any cookies from the AD FS server; for this stage in this example, the client does not have any cookies from the AD FS server. The client then sends the GET to the AD FS server.

4. The AD FS server receives the HTTP GET from the client with the query string from the relying party.

   The AD FS server sends messages to the client to interact with the user or the client in order to obtain (either manually or automatically) credential information from the user. In the case of this example, the server sends a forms page to the client so that the user can enter user credentials.

   Note The details of this interaction between the server and client are outside the scope of this document, but typically involve standard GET and POST HTTP communication [RFC2616] and HTTP over Secure Sockets Layer (SSL) and Transport Layer Security (TLS) [RFC2818], as well as others. In addition, there are other types of interaction that might occur between the AD FS server and the client at this point in the message flow instead of what is described above; for example, a Kerberos request to automatically obtain user credentials might be sent to the client instead of a forms page.

5. The user supplies the required credential information and the client sends the response to the AD FS server.

6. The AD FS server authenticates the user, in the case of this example, by contacting Active Directory. The AD FS server then generates the security token needed by the relying party, and also constructs a cookie that contains information about the user authentication that has occurred. This data is sent to the client along with hidden JavaScript code in an HTTP 200 response.

7. The client executes the hidden JavaScript code, which causes it to send an HTTP POST to the web resource. The POST includes the security token given by the AD FS server. In addition, the client stores the cookie in its cookie cache.

   At this time, the user is able to access the web resource.

8. Sometime in the future, the user sends an HTTP GET, via the client, to another web resource. The relying party for this other web resource relies on the same STS, that is, the AD FS server, for secure access to its web resource.

9. As in step 2 above, a redirect is sent from this new relying party to the client with the address of the AD FS server.

10. Step 3 from above is repeated with the appropriate data. This time, however, when the client sends the HTTP GET to the AD FS server with the query string from the new relying party, it also includes the cookie that the AD FS server sent to it when sending the security token in step 6.

11. Having received the HTTP GET, the AD FS server has also received the cookie information that it stored in the client earlier. Having this cookie information, the server is not required to request user credentials again, but can immediately begin to create the security material as in step 6 above. The generated security information is sent to the client as hidden JavaScript, as described previously.

12. The client executes the hidden JavaScript code as it did in step 7 above. At this time, the user is able to access the second web resource without having to log on a second time.

4   Appendix A: Product Behavior

The information in this document is applicable to the following versions of products:

- 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

<1> Section 2.1: The following table shows the availability of each of the AD FS protocols in various versions of Windows, including when the protocol was first included and, if applicable, when the protocol was last available. If the table does not show a Windows version in which it was last available, the protocol continues to be available in the latest version of Windows according to the applicability list at the beginning of this section.

Protocol	First included in:	Last available in:
ADFSOAL	Windows Server 2012 R2	--
ADFSPIP	Windows Server 2012 R2	--
ADFSPP	Windows Server 2003 R2	Windows Server 2012
ADFSWAP	Windows Server 2003 R2	--
MWBE	Windows 2000	--
MWBF	Windows 2000	--
OAPX	Windows Server 2012 R2	--
OAuth	Windows Server 2012 R2	--
SAML1	Windows 2000	--
SAML2	Windows Server 2008 R2	--
SAMLPR	Windows Server 2003 R2	Windows Server 2012
WS-Federation and WSFedPRP	Windows 2000	--
WS-Trust	Windows 2000	--

Release: Thursday, May 15, 2014

5 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

6 Index