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

Release: Thursday, May 15, 2014

Revision Summary

Date	Revision History	Revision Class	Comments
08/08/2013	1.0	New	Released new document.
11/14/2013	2.0	Major	Significantly changed the technical content.
02/13/2014	3.0	Major	Significantly changed the technical content.
05/15/2014	3.0	No change	No change to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This is a specification of the **Active Directory Federation Services and Proxy system** and the protocols that define the interaction behaviors between **Active Directory Federation Services (AD FS)** and the **Web Application Proxy**, or simply Proxy. It describes the intended functionality of the system and how the protocols in this system interact.

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]:

Active Directory Federation Services (AD FS)
Coordinated Universal Time (UTC)
extended key usage (EKU)
preauthentication
proxy
token
UTC (Coordinated Universal Time)

The following terms are specific to this document:

Active Directory Federation Services and Proxy system: A system of features and protocols whereby a client located outside the boundaries of a corporate network can access application services located inside those boundaries.

farm configuration: A collection of servers, each of which provide the same services, and to each of which a service request can be routed for load balancing.

internal network: The portion of the corporate network that is protected by a firewall.

non-claims-aware: A characteristic of a network device or application that makes it unable to participate in claims-based authentication.

perimeter network: The portion of the corporate network that is on the outside of the firewall and is exposed to external network traffic.

Web Application Proxy: A set of components that provide proxy services for clients that are requesting access to application services inside the boundaries of a corporate network.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described 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.

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.

[IETFDRAFT-JWS] Internet Engineering Task Force (IETF), "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature-10, April 2013, http://tools.ietf.org/html/draft-ietf-jose-json-web-signature-10

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

[MS-OFBA] Microsoft Corporation, "Office Forms Based Authentication Protocol".

[RFC1422] Kent, S., "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", RFC 1422, February 1993, http://www.rfc-editor.org/rfc/rfc1422.txt

[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

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

[RFC2478] Baize, E., and Pinkas, D., "The Simple and Protected GSS-API Negotiation Mechanism", RFC 2478, December 1998, http://www.ietf.org/rfc/rfc2478.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

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., et al., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.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

[RFC3339] Klyne, G., and Newman, C., "Date and Time on the Internet: Timestamps", RFC 3339, July 2002, http://www.ietf.org/rfc/rfc3339.txt

[RFC4158] Cooper, M., Dzambasow, Y., Hesse, P., et la., "Internet X.509 Public Key Infrastructure: Certification Path Building", RFC 4158, September 2005, http://rfc-editor.org/rfc/rfc4158.txt

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

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, http://www.ietf.org/rfc/rfc0793.txt

[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

[WSFederation1.2] Kaler, C., McIntosh, M., "Web Services Federation Language (WS-Federation)", Version 1.2, May 2009, http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html

1.2.2 Informative References

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

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

[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

1.3 Overview

The Active Directory Federation Services and Proxy system provides services for authentication, authorization, and access to application services located inside the boundaries of the corporate network for clients that are located outside that boundary. The system is composed of Active Directory Federation Services (AD FS) and the Proxy.

AD FS is located inside the boundaries of the corporate network and can run on one server or multiple servers (also known as a "**farm configuration**"). It is a collection of authentication and authorization services exposed to clients over the HTTP protocol [RFC2616]. AD FS implements a set of application authentication protocols including WS-Federation [WSFederation], SAML-P [SAMLCore2], and OAuth [RFC6749].

The Proxy is a service located at the "edge" of the corporate network. It provides **proxy** services for clients requesting access to application services inside the corporate network and orchestrates access traffic to these services.

The Proxy directs all authentication traffic to the AD FS in the internal network and provisions for certificate-based authentication in particular.

The Proxy publishes application services that are located inside the boundaries of the corporate network and makes them available for access to clients that are outside. It "gates" the access to the network by orchestrating the authentication to the edge through the AD FS before allowing the access to the application service (that is, **preauthentication**).

AD FS defines and implements a protocol that the Proxy supports and that allows the Proxy to orchestrate access to the network by authenticating requests to the edge.

The following diagram illustrates the various components of the system.

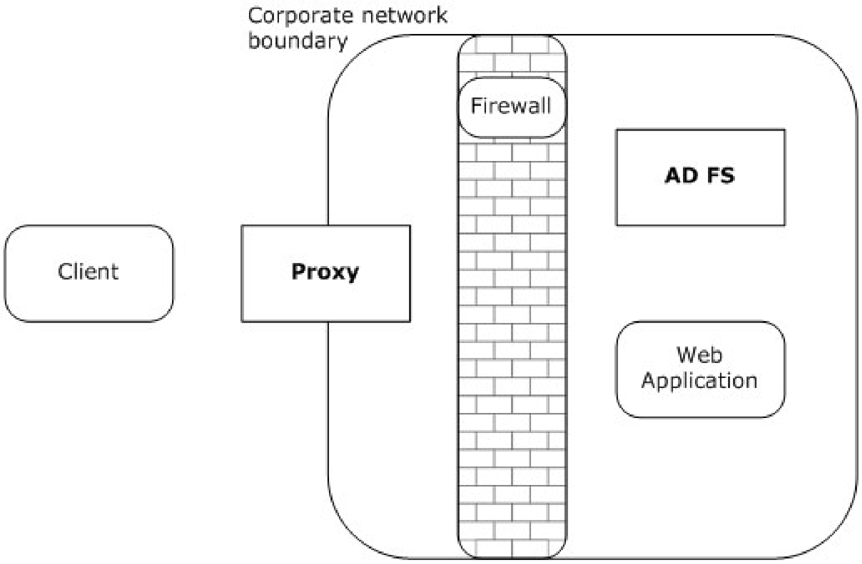

Figure 1: System components

The following components are part of the Active Directory Federation Services and Proxy system:

- **AD FS**: A federation services provider. In this specification this component will be referred to as the server.
- **Proxy**: Both an authentication and an application proxy. In this specification this component will be referred to as the client.

The following components interact with the Active Directory Federation Services and Proxy system:

- **Client**: These components refer to the type of client (for example, browser or rich client) in addition to the identity of the user and the device that is accessing a particular application service.
- **Firewall**: A component that filters traffic flowing between the **perimeter network** and the **internal network**. In the system described, web traffic is allowed between the Proxy and the AD FS and between the Proxy and the web application.
- **Web Application**: Any web service or application to which a client connects and that typically requires authentication for the user in the client.

This specification describes the distinct areas of interaction between the Proxy and the AD FS.

1.4 Relationship to Other Protocols

The following figure illustrates the relationship of this protocol to other protocols.

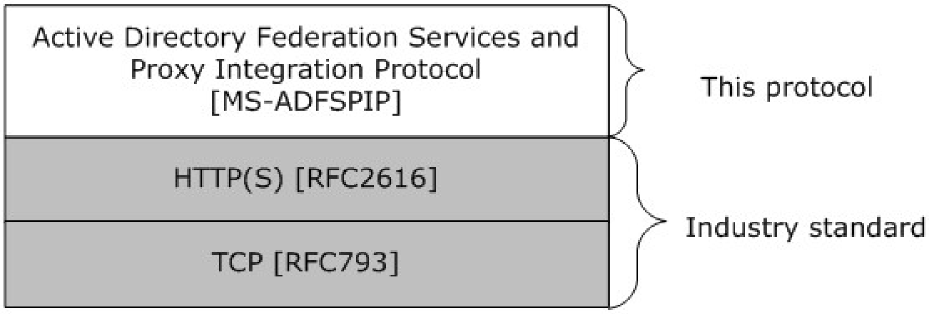
Active Directory Federation Services and Proxy Integration Protocol [MS-ADFSPIP]
HTTP(S) [RFC2616]
TCP [RFC793]
This protocol
Industry standard

Figure 2: Protocols related to the Active Directory Federation Services and Proxy Integration Protocol

This protocol uses TCP [RFC793] as its transport.

Where specified, this protocol uses base64url encoding ([RFC4648] section 5).

1.5 Prerequisites/Preconditions

No prerequisites or preconditions.

1.6 Applicability Statement

The protocols in the Active Directory Federation Services and Proxy system are applicable to any situation in which the following are important:

1. A proxy for AD FS.

2. Publishing of web applications or services behind-the-firewall to the Internet.

3. Preauthentication of clients accessing web applications or services behind a firewall.

1.7 Versioning and Capability Negotiation

This protocol does not provide any mechanism for capability negotiation.

1.8 Vendor-Extensible Fields

This protocol does not provide any vendor-extensible fields.

1.9 Standards Assignments

This protocol has not been assigned any standard parameters.

2 Messages

2.1 Transport

The protocol MUST be transported by HTTP/HTTPS [RFC2616]. The protocol requires HTTP/HTTPS ports as specified in section 2.2.2.4, attributes "HttpPort", "HttpsPort" and "HttpsPortForUserTlsAuth", obtained during Proxy server registration (section 3.4.5.1).

2.2 Common Data Types

This section defines the set of resource types that are consumed or produced by this protocol. Common element definitions are included in this section.

2.2.1 HTTP Headers

The following table summarizes the set of HTTP Headers defined by this specification.

Header	Description
X-MS-Endpoint-Absolute-Path	section 2.2.1.3
X-MS-Forwarded-Client-IP	section 2.2.1.2
X-MS-Proxy	section 2.2.1.1
X-MS-Target-Role	section 2.2.1.4

2.2.1.1 X-MS-Proxy

This header MUST contain the value of the server name of the proxy. This header is included when the proxy is processing client incoming requests as described in the runtime behaviors for the AD FS proxy server details in section 3.6.

```
String = *(%x20-7E)
X-MS-Proxy = String
```

2.2.1.2 X-MS-Forwarded-Client-IP

This header MUST contain the value of the IP address of the client sending the request. This header MUST be included when the proxy is processing incoming requests from clients trying to access the server.

```
String = *(%x20-7E)
X-MS-Forwarded-Client-IP = String
```

2.2.1.3 X-MS-Endpoint-Absolute-Path

This header MUST contain the full URL of the incoming request. This header MUST be included when the proxy is processing incoming requests from clients trying to access the server.

```
String = *(%x20-7E)
X-MS-Endpoint-Absolute-Path = String
```

2.2.1.4 X-MS-Target-Role

This header MUST contain the value "PrimaryComputer" to specify that a given HTTP GET request MUST perform the fetch on a server that has both read and write capabilities on the data.

```
String = *(%x20-7E)
X-MS-Target-Role = String
```

2.2.2 Complex Types

The following are the defined types used by the protocol details.

2.2.2.1 Proxy Trust

This is a JSON object containing a trust certificate. The format of the object is as follows:

```
{ "SerializedTrustCertificate" : "<certificate>" }
```

certificate: Base64 string encoded ([RFC4648] section 4) X509 certificate [RFC4158].

2.2.2.2 Proxy Trust Renewal

This is a JSON object containing a new trust certificate. The format of the object is as follows:

```
{ "SerializedReplacementCertificate" : "<certificate>" }
```

certificate: Base64 string encoded ([RFC4648] section 4) X509 certificate [RFC4158].

2.2.2.3 Proxy Relying Party Trust

This is a JSON object containing the identifier of the web application for the proxy. The format of the object is as follows:

```
{ "Identifier" : "<web-application-for-client-id>" }
```

web-application-for-client-id: URI of the web application representing the client. The server will issue tokens with this value as the audience as described in section 3.13.

2.2.2.4 Configuration

This is a JSON object containing information about the AD FS service. The format of the object is as follows:

```
{
  "ServiceConfiguration" :
    {
      "ServiceHostName" : "<service-host-name>",
      "HttpPort" : "<http-port-number>",
      "HttpsPort" : "<https-port-number >",
      "HttpsPortForUserTlsAuth" : "<user-TLS-port-number>",
      "DeviceCertificateIssuers" : [ "<device-certificate-issuer>", * ],
      "ProxyTrustCertificateLifetime" : "<trust-renewal-interval>"
    },
  "EndpointConfiguration" :
    [
      {
        "Path" : "<endpoint-uri>",
        "PortType" : "<port-type>",
        "AuthenticationSchemes" : "<credential-collection-scheme>",
        "ClientCertificateQueryMode" : "<tls-query-behavior>",
        "CertificateValidation" : "<certificate-validation>",
        "ServicePath" : "<service-endpoint-uri>",
        "ServicePortType" : "<service-port-type>"
      }, *
    ]
}
```

service-host-name: Host name of the AD FS service.

http-port-number: Port number for endpoints listening on HTTP.

https-port-number: Port number for endpoints listening on HTTPs.

user-tls-port-number: Port number for user TLS authentication endpoints.

device-certificate-issuer: Base64 string encoded ([RFC4648] section 4) X509 certificate [RFC4158].

trust-renewal-interval: Hint for proxy certificate lifetime.

endpoint-uri: URI of endpoint.

port-type: Port Type (section 2.2.2.12) for endpoint.

credential-collection-scheme: Credential Collection Scheme (section 2.2.2.13) for endpoint.

tls-query-behavior: TLS Query Behavior (section 2.2.2.14) for endpoint.

certificate-validation: Certificate Validation (section 2.2.2.15) for endpoint.

service-endpoint-uri: URI of endpoint on server. This URI is relative to service-host-name.

service-port-type: Port Type (section 2.2.2.12) for corresponding endpoint on server.

2.2.2.5 Relying Party Trust List

This is a JSON array of objects containing web application information. The format of the objects is as follows:

```
[ {
    "objectIdentifier" : "<object-identifier>",
    "name" : "<web-application-name>",
    "publishedThroughProxy" : "<is-web-application-published>",
    "nonClaimsAware" : "<is-a-non-claims-aware-web-application>",
    "enabled" : "<is-web-application-enabled>"
  }, + ]
```

object-identifier: The immutable object identifier for the web application on the server.

web-application-name: The name of the web application on the server, unique across web applications.

is-web-application-published: Boolean user configuration declaring this web application as being accessible from outside the internal network through a client.

is-a-non-claims-aware-web-application: Boolean value specifying if the web application is a **non-claims-aware** web application.

enabled: Boolean value specifying if the web application is enabled at the server.

2.2.2.6 Relying Party Trust

This is a JSON object containing detailed web application information. The format of the object is as follows:

```
{
  "objectIdentifier" : "<object-identifier>",
  "name" : "<web-application-name>",
  "publishedThroughProxy" : "<is-web-application-published>",
  "nonClaimsAware" : "<is-a-non-claims-aware-web-application>",
  "enabled" : "<is-web-application-enabled>",
  "identifiers" : [ <web-application-identifier>, * ],
  "proxyTrustedEndpoints" : [ <web-application-at-proxy-endpoint-url>, *],
  "proxyEndpointMappings" :
    [ { "Key" = "<internal-url>", "Value" = "external-url" }, *]
}
```

object-identifier: The unique object identifier for the web application.

web-application-name: The name of the web application on the server, unique across web applications.

is-web-application-published: Boolean user configuration declaring this web application as accessible from outside the internal network through a client. This value MUST correspond to the value of (proxyTrustedEndpoints.Count > 0).

is-a-non-claims-aware-web-application: Boolean value specifying if the web application is a non-claims-aware web application.

enabled: Boolean value specifying if the web application is enabled at the server.

web-application-identifier: An identifier of the web application on the server.

web-application-at-proxy-endpoint-url: A URL representing an endpoint on the client for the web application where the server will issue tokens to.

internal-url: The internal URL corresponding to the internal-to-external mapping.

external-url: The external URL corresponding to the internal-to-external mapping.

2.2.2.7 Relying Party Trust Publishing Settings

This is a JSON object containing web application publishing information. The format of the object is as follows:

```
{
  "externalUrl" : "<external-url>",
  "internalUrl" : "<internal-url>",
  "proxyTrustedEndpointUrl" : "<web-application-at-proxy-url>"
}
```

external-url: The external URL to be associated with the web application external-to-internal mappings (section 2.2.2.6).

internal-url: The internal URL to be associated with the web application external-to-internal mappings (section 2.2.2.6).

web-application-at-proxy-url: The URL of the endpoint in the client where the server will issue tokens to.

2.2.2.8 Store Entry List

This is a JSON array of store entry objects, which are defined in section 2.2.2.9.

2.2.2.9 Store Entry

This is a JSON object containing store entry information. The format of the object is as follows:

```
{
  "key" : "<entry-key>",
  "version" : "<entry-version>",
  "value" : "<entry-value>"
}
```

entry-key: A string that contains the key of the data value for the store entry.

entry-version: A value that specifies the version of the key/value pair for the store entry.

entry-value: The value of the data-blob corresponding to the given key for the store entry.

2.2.2.10 Store Entry Key and Value

This is a JSON object containing the value of a store entry. The format of the object is as follows:

```
{
  "key" : "<entry-key>",
  "value" : "<entry-value>"
}
```

entry-key: A string containing the key of the data value for the store entry.

entry-value: The value of the data-blob corresponding to the given key for the store entry.

2.2.2.11 Serialized Request with Certificate

This is a JSON object containing a serialized request plus a serialized client certificate and its usage. The format of the object is as follows:

```
{
  "Request" :
    {
      "AcceptTypes" : "<accept-types>",
      "ContentEncoding" : "<content-encoding>",
      "ContentLength" : "<content-length>",
      "ContentType" : "<content-type>",
      "Cookies" :
        {
          "Name" : "<cookie-name>",
          "Value" : "<cookie-value>",
          "Path" : "<cookie-path>",
          "Domain" : "<cookie-domain>",
          "Expires" : "<cookie-expires>",
          "Version" : "<cookie-version>",
        },
      "Headers" :
        [ { "Name" : "<header-name>", "Value" : "<header-value>" }, * ],
      "HttpMethod" : "<http-method>",
      "RequestUri" : "<request-uri>",
      "QueryString" : "<query-string>",
      "UserAgent" : "<user-agent>",
      "UserHostAddress" : "<user-host-address>",
      "UserHostName" : "<user-host-name>",
      "UserLanguages" : "<user-languages>",
    },
  "SerializedClientCertificate" : "<serialized-client-certificate>",
  "CertificateUsage" : "<certificate-usage>",
}
```

accept-types: A string with a list of supported MIME accept types by the client. This corresponds to the values of the Accept header of the request.

content-encoding: Character set of the entity-body of the request.

content-length: Length in bytes of content sent in the request.

content-type: MIME content type of the request.

cookie-name: Name of the cookie.

cookie-value: Value of the cookie.

cookie-path: Virtual path transmitted with the cookie.

cookie-domain: Domain associated with the cookie.

cookie-expires: Expiration date and time of the cookie.

cookie-version: Version of the cookie.

header-name: Name of header.

header-value: Value of header.

http-method: HTTP data transfer method of the request, for example GET, POST, HEAD.

request-uri: URI of the request.

query-string: Query string included in the request.

user-agent: User agent presented in the request.

user-host-address: IP address and port number to which the request was directed.

user-host-name: DNS name and port number (if provided) specified in the request.

user-languages: Natural languages preferred for the response.

serialized-client-certificate: Client certificate obtained from TLS handshake base64 string encoded.

certificate-usage: Certificate Type (section 2.2.2.16) for certificate.

2.2.2.12 Port Type

This is an enumeration with the following values:

```
{
  "HttpPort"
  "HttpsPort"
  "HttpsPortForUserTlsAuth"
}
```

2.2.2.13 Credential Collection Scheme

This is an enumeration with the following integer values indicating the type of credential to collect from the client:

```
{
  8
  32768
}
```

8: Indicates basic authentication credentials.

32768: Indicates anonymous authentication.

2.2.2.14 TLS Query Behavior

This is an enumeration with the following values:

```
{
  "None"
  "QueryAndAccept"
  "QueryAndRequire"
}
```

2.2.2.15 Certificate Validation

This is an enumeration with the following values:

```
{
  "None"
  "User"
  "Device"
}
```

2.2.2.16 Certificate Type

This is an enumeration with the following values:

```
{
  "User"
  "Device"
}
```

2.2.2.17 Proxy Token

This is a JSON object representing the token issued to the client. The format of the object is defined in [IETFDRAFT-JWS] and is as follows:

```
{
  "ver" : "<version>",
  "aud" : "<audience>",
  "iat" : "<issued-at>",
  "exp" : "<expire>",
  "iss" : "<issuer>",
  "relyingpartytrustid" : "<rp-trust-id>",
  "deviceregid" : "<device-registration-id>",
  "authinstant" : "<auth-instant>",
  "authmethod" : "<auth-method>",
  "upn" : "<upn>"
}
```

version: Token version with a value of 1.0.

audience: Audience for this token. The proxy SHOULD verify that this value matches the value for [Proxy Service State].ProxyRelyingPartyTrustIdentifier.

issued-at: Issued at date and time. The proxy SHOULD verify that this value corresponds to a time in the past (before the current time). This is a JSON numeric value representing the number of seconds from 1970-01-01T0:0:0Z **Coordinated Universal Time (UTC)** until the specified **UTC** date/time. See [RFC3339] for details regarding date/times in general and UTC in particular.

expire: Expiration time of token. The proxy SHOULD verify that this value corresponds to a time in the future (after the current time). This is a JSON numeric value representing the number of seconds from 1970-01-01T0:0:0Z UTC until the specified UTC date/time. See [RFC3339] for details regarding date/times in general and UTC in particular.

issuer: Trusted issuer for this token. The proxy SHOULD verify that this value corresponds to the issuer URI that is published by the server issuing this token through its Federation Metadata [WSFederation1.2].

rp-trust-id: GUID representing application being accessed. The proxy MAY use this value to correlate requests and tokens when listening to multiple requests.

device-registration-id: Identity of the device attempting the access in the form of its certificate thumbprint. The proxy MAY use this value to correlate the client of the request with the client of the token.

auth-instant: Time of authentication. The proxy SHOULD verify that this value corresponds to an earlier time than the issued-at value.

auth-method: Authentication method. The proxy MAY use this value to perform richer authorization of access.

upn: User Principal Name (UPN) of user attempting the access.

2.2.2.18 Combined Token

This is a JSON object containing an access token for the client and an access token for the web application. The format of the object is as follows:

```
{
  "proxy_token" : <proxy-token>,
  "access_token" : <access-token>
}
```

proxy-token: [Proxy Token] (section 2.2.2.17).

access-token: Token issued by the server to the web application.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation of the client and server maintain to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.1.1.1 Server State

The following represents the data structure the server MUST hold in order to satisfy these protocol requirements<1>:

```
{
  "ProxyTrustedCertficates" : [ "<certificate-identifier>", * ],
  "ProxyRelyingPartyTrust" : "<web-application-for-proxy>",
  "Configuration" : "<configuration>",
  "RelyingPartyTrusts" : [ "<web-application>", * ],
  "ProxyStore" : [ "StoreEntry" : "<store-entry>", * ]
}
```

certificate-identifier: Data that MUST be used to validate the certificate when presented again.

web-application-for-proxy: Proxy Relying Party Trust (section 2.2.2.3) representing the web application for the client in the server.

configuration: Configuration (section 2.2.2.4) representing service and endpoint configuration.

web-application: Relying Party Trust (section 2.2.2.6) representing an available web application in the server.

store-entry: Store Entry (section 2.2.2.9) containing the triplet of key-version-value of data used by the client for its own consumption.

3.1.1.2 Client State

The following represents the data structure the proxy service MUST hold in order to satisfy these protocol requirements:

```
{
  "TrustCertificate" : "<certificate-with-private-key>",
  "ProxyRelyingPartyTrustIdentifier" : "<web-application-for-client-id>",
  "Configuration" : "<configuration>",
  "RelyingPartyTrusts" : [ "<web-application>", * ]
}
```

certificate-with-private-key: Points to a certificate. The proxy service MUST have a private key for the certificate.

web-application-for-client-id: Identifier of the web application representing the client on the server. This identifier MUST be used by the client when referring to itself on requests to the server.

configuration: Configuration (section 2.2.2.4) obtained from the server.

web-application: Relying Party Trust State (section 3.1.1.3) containing the configuration for a web application on the server.

3.1.1.3 Relying Party Trust State

The following represents the data structure the server MUST hold in order to satisfy these protocol requirements:

```
{
  "RelyingPartyTrust" : "<web-application>",
  "RedirectBasedPreauth" : "<redirect-based-preauth>"
}
```

web-application: Proxy Relying Party Trust (section 2.2.2.3) representing the web application that the server can issue tokens for.

pre-auth-required: Boolean denoting that access from outside the network needs preauthentication.

redirect-based-preauth: Boolean denoting that access from outside the network needs preauthentication based on HTTP redirects.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Proxy Registration Server Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

For the system to function properly, the client and the server MUST mutually authenticate each other using client TLS authentication [RFC2246]. For this, the client MUST have the appropriate local configuration to evaluate the trustworthiness of the server TLS certificate and MUST have a client TLS certificate for authenticating itself to the server.

The following resources are required to create and maintain a proper trust configuration between the client and the server.

Resource	Description
Proxy/EstablishTrust	Resource used to establish a trust with the server.
Proxy/RenewTrust	Resource used to renew the trust with the server.

The responses to all the operations can result in the following status codes.

Status code	Description
200	The operation has succeeded.
400	The request is not valid.
401	Unauthorized for specified user credentials or for client TLS certificate.
404	The object does not exist.
405	Invalid verb used in request (GET, DELETE, POST, PUT).
409	The object already exists.
500	Version is not specified where required or any other internal error.
501	Version specified (api-version) is invalid (only valid value is 1).

If the operation authenticates using Integrated Windows authentication [RFC2478], the server MUST validate that the authenticated principal is authorized to do the corresponding operation on the server.

3.2.5.1 Proxy/EstablishTrust

The client MUST first establish a trust with the server in order to act as a Proxy on the system.

3.2.5.1.1 POST

This operation creates a trust based on a Proxy Trust (section 2.2.2.1).

The operation is transported by a HTTP **POST** and can be invoked through the following URIs:

```
adfs/proxy/EstablishTrust
adfs/proxy/PrimaryWriter/EstablishTrust
```

If the operation is invoked through `adfs/proxy/EstablishTrust`, the request MUST authenticate using HTTP Basic authentication [RFC2617].

If the operation is invoked through `adfs/proxy/PrimaryWriter/EstablishTrust`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
500

3.2.5.1.1.1 Request Body

The request body MUST be a Proxy Trust (section 2.2.2.1).

3.2.5.1.1.2 Response Body

No response body is returned.

3.2.5.1.1.3 Processing Details

If the operation authenticates using HTTP Basic authentication [RFC2617], the server MUST validate that the authenticated principal is authorized to function as a proxy.

The server MUST validate that the [Proxy Trust].SerializedTrustCertificate has an **extended key usage (EKU)** for client authentication (1.3.6.1.5.5.7.3.2) ([RFC3280] section 4.2.1.13) and is within the validity period ([RFC1422] section 3.3). If validation fails, the server MUST return a HTTP error code of 400.

On successful authentication and authorization, the server MUST add [Proxy Trust].SerializedTrustCertificate to [Service State Data].ProxyTrustedCertificates for future validations.

3.2.5.2 Proxy/RenewTrust

The client MUST ensure that the trust with the server remains valid by renewing the trust certificate with the server.

3.2.5.2.1 POST

This operation renews a trust based on a Proxy Trust Renewal (section 2.2.2.2).

The operation is transported by a HTTP **POST** and can be invoked through the following URIs:

```
adfs/proxy/RenewTrust
adfs/proxy/PrimaryWriter/RenewTrust
```

If the operation is invoked through `adfs/proxy/RenewTrust`, the request MUST authenticate using client TLS authentication [RFC2246]. The server MUST validate that the certificate presented by the client during client TLS authentication [RFC2246] can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated, the server MUST return a HTTP error code of 400.

If the operation is invoked through `adfs/proxy/PrimaryWriter/RenewTrust`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
500

3.2.5.2.1.1 Request Body

The request body MUST be Proxy Trust Renewal (section 2.2.2.2).

3.2.5.2.1.2 Response Body

No response body is returned.

3.2.5.2.1.3 Processing Details

The server MUST validate that the [Proxy Trust].SerializedReplacementCertificate has an extended key usage (EKU) for client authentication (1.3.6.1.5.5.7.3.2) ([RFC3280] section 4.2.1.13) and is within the validity period ([RFC1422] section 3.3). If validation fails, the server MUST return a HTTP error code of 400.

The server MUST add [Proxy Trust].SerializedReplacementCertificate to [Service State Data].ProxyTrustedCertificates for future validations.

3.2.5.3 Proxy/WebApplicationProxy/Trust

The client MUST register with the server as a token recipient with the server before it can function as the Proxy on the system.

3.2.5.3.1 GET

This operation returns a Proxy Relying Party Trust (section 2.2.2.3) corresponding to the web application for the client in the server.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/proxy/WebApplicationProxy/trust?api-version=1
```

The request MUST authenticate using client TLS authentication [RFC2246]. The server MUST validate that the certificate presented by the client during client TLS authentication [RFC2246] can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated, the server MUST return a HTTP error code of 401.

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
500
501

3.2.5.3.1.1 Request Body

The server MUST ignore any request body.

3.2.5.3.1.2 Response Body

The response body MUST be a Proxy Relying Party Trust (section 2.2.2.3).

3.2.5.3.1.3 Processing Details

On successful authentication the server MUST return [Service State Data].ProxyRelyingPartyTrust (section 3.1.1.1).

3.2.5.3.2 POST

This operation creates the proxy relying party trust based on a Proxy Relying Party Trust (section 2.2.2.3).

The operation is transported by a HTTP **POST** and can be invoked through the following URIs:

```
adfs/proxy/WebApplicationProxy/trust?api-version=1
adfs/proxy/WebApplicationProxy/PrimaryWriter/trust?api-version=1
```

If the operation is invoked through adfs/proxy/WebApplicationProxy/trust?api-version=1, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through adfs/proxy/WebApplicationProxy/PrimaryWriter/trust?api-version=1, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
409
500
501

3.2.5.3.2.1 Request Body

The request body MUST be a Proxy Relying Party Trust (section 2.2.2.3).

3.2.5.3.2.2 Response Body

No response body is returned.

3.2.5.3.2.3 Processing Details

On successful authentication the server MUST verify that [Service State Data].ProxyRelyingPartyTrust is not set.

If it is set, the server MUST return a HTTP error code of 409.

If it is not set, the server MUST create the relying party trust for the proxy with an identifier of the received [Proxy Relying Party Trust].Identifier and set the [Service State Data].ProxyRelyingPartyTrust to the value of the received Proxy Relying Party Trust (section 2.2.2.3).

3.2.5.3.3 DELETE

This operation removes the proxy relying party trust.

The operation is transported by a HTTP **DELETE** and can be invoked through the following URIs:

```
adfs/proxy/WebApplicationProxy/trust?api-version=1
```

```
adfs/proxy/WebApplicationProxy/PrimaryWriter/trust?api-version=1
```

If the operation is invoked through `adfs/proxy/WebApplicationProxy/trust?api-version=1`, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through `adfs/proxy/WebApplicationProxy/PrimaryWriter/trust?api-version=1`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
500
501

3.2.5.3.3.1 Request Body

The server MUST ignore any request body.

3.2.5.3.3.2 Response Body

No response body is returned.

3.2.5.3.3.3 Processing Details

On successful authentication the server MUST verify that [Service State Data].ProxyRelyingPartyTrust is set.

If it is not set the server MUST return a HTTP error code of 404.

If it is set the server MUST remove the relying party trust for the proxy and clear the [Service State Data].ProxyRelyingPartyTrust value.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Proxy Registration Client Details

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

See corresponding section on Server Details.

In all operations where the server requires authenticating the proxy using client TLS authentication [RFC2246], the proxy MUST present the certificate on [Proxy Service State Data].TrustCertificate during client TLS authentication.

3.3.5.1 Proxy/EstablishTrust

See corresponding section on Server Details.

3.3.5.1.1 POST

See corresponding section on Server Details.

3.3.5.1.1.1 Request Body

See corresponding section on Server Details.

3.3.5.1.1.2 Response Body

See corresponding section on Server Details.

3.3.5.1.1.3 Processing Details

[Proxy Trust].SerializedTrustCertificate MUST have an EKU for client authentication (1.3.6.1.5.5.7.3.2) ([RFC3280] section 4.2.1.13) and MUST be within validity period ([RFC1422] section 3.3). The client MUST have the private key of this certificate.

If the server response is a HTTP status code of 200 the proxy MUST set [Proxy Service State].TrustCertificate to [Proxy Trust].SerializedTrustCertificate for future authentication to the server.

3.3.5.2 Proxy/RenewTrust

See corresponding section on Server Details.

3.3.5.2.1 POST

See corresponding section on Server Details.

3.3.5.2.1.1 Request Body

See corresponding section on Server Details.

3.3.5.2.1.2 Response Body

See corresponding section on Server Details.

3.3.5.2.1.3 Processing Details

[Proxy Trust].SerializedReplacementCertificate MUST have an EKU for client authentication (1.3.6.1.5.5.7.3.2) ([RFC3280] section 4.2.1.13) and MUST be within validity period ([RFC1422] section 3.3). The proxy MUST have the private key of this certificate.

If the server response is a HTTP status code of 200 the proxy MUST set [Proxy Service State].TrustCertificate to [Proxy Trust].SerializedReplacementCertificate for future authentication to the server<2>.

3.3.5.3 Proxy/WebApplicationProxy/Trust

See corresponding section on Server Details.

3.3.5.3.1 GET

See corresponding section on Server Details.

3.3.5.3.1.1 Request Body

See corresponding section on Server Details.

3.3.5.3.1.2 Response Body

See corresponding section on Server Details.

3.3.5.3.1.3 Processing Details

No processing details.

3.3.5.3.2 POST

See corresponding section on Server Details.

3.3.5.3.2.1 Request Body

See corresponding section on Server Details.

3.3.5.3.2.2 Response Body

See corresponding section on Server Details.

3.3.5.3.2.3 Processing Details

If the server response is a HTTP status code of 200 the proxy MUST set [Proxy Service State].ProxyRelyingPartyTrustIdentifier to [Proxy Relying Party Trust].Identifier.

3.3.5.3.3 DELETE

See corresponding section on Server Details.

3.3.5.3.3.1 Request Body

See corresponding section on Server Details.

3.3.5.3.3.2 Response Body

See corresponding section on Server Details.

3.3.5.3.3.3 Processing Details

If the server response is a HTTP status code of 200 the proxy MUST clear [Proxy Service State].ProxyRelyingPartyTrustIdentifier.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 Service Configuration Server Details

3.4.1 Abstract Data Model

None.

3.4.2 Timers

None.

3.4.3 Initialization

None.

3.4.4 High-Layer Triggered Events

None.

3.4.5 Message Processing Events and Sequencing Rules

For the proxy to function properly as a proxy component on the system, it MUST retrieve information from the server about the service configuration and the endpoints it listens to, and about the available relying party trusts.

The following resources are required to retrieve server configuration.

Resource	Description
Proxy/GetConfiguration	Resource used to retrieve service and endpoint configuration.
Proxy/RelyingPartyTrusts	Resource used to retrieve all relying party trusts.
Proxy/RelyingPartyTrusts/{Identity}	Resource used to retrieve a particular relying party trust.

The responses to all the operations can result in the following status codes.

Status code	Description
200	The operation has succeeded.
400	The request is not valid.
401	Unauthorized for specified user credentials or for client TLS certificate.
404	The object does not exist.
405	Invalid verb used in request (GET, DELETE, POST, PUT).
409	The object already exists.
500	Version is not specified where required or any other internal error.
501	Version specified (api-version) is invalid (only valid value is 1).

3.4.5.1 Proxy/GetConfiguration

The server MUST provide configuration for the client's run-time function.

3.4.5.1.1 GET

This operation returns a Configuration (section 2.2.2.4) containing service and end-point configuration.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/proxy/GetConfiguration
```

The request MUST authenticate using client TLS authentication [RFC2246]. The server MUST validate that the certificate presented by the client during client TLS authentication [RFC2246] can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated the server MUST return a HTTP error code of 400.

The response message for this operation can result in the following status codes.

Status code
200
400
405
500

3.4.5.1.1.1 Request Body

The server MUST ignore any request body.

3.4.5.1.1.2 Response Body

The response body MUST be a Configuration (section 2.2.2.4).

3.4.5.1.1.3 Processing Details

On successful authentication the server MUST return a [Service State Data].Configuration (section 3.1.1.1).

3.4.5.2 Proxy/RelyingPartyTrusts

The proxy MUST retrieve information about relying party trusts to obtain relying party trust object identifiers that the proxy MUST use when identifying relying party trusts on requests to the server.

3.4.5.2.1 GET

This operation returns a Relying Party Trust List (section 2.2.2.5) containing all available relying party trusts.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/proxy/RelyingPartyTrusts?api-version=1
```

The request MUST authenticate using client TLS authentication [RFC2246]. The server MUST validate that the certificate presented by the client during client TLS authentication [RFC2246] can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated the server MUST return a HTTP error code of 401.

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
500

Status code
501

3.4.5.2.1.1 Request Body

The server MUST ignore any request body.

3.4.5.2.1.2 Response Body

The response body MUST be a Relying Party Trust List (section 2.2.2.5).

3.4.5.2.1.3 Processing Details

On successful authentication the server MUST return a [Service State Data].RelyingPartyTrusts (section 3.1.1).

3.4.5.3 Proxy/RelyingPartyTrusts/

This resource is available for the client to access data about a specific web application identified by {Identifier}.

3.4.5.3.1 GET

This operation returns a Relying Party Trust (section 2.2.2.6) containing information specific to a relying party trust.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/proxy/RelyingPartyTrusts/{Identifier}?api-version=1
```

The request MUST authenticate using client TLS authentication [RFC2246]. The server MUST validate that the certificate presented by the client during client TLS authentication [RFC2246] can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated the server MUST return a HTTP error code of 401.

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
500
501

3.4.5.3.1.1 Request Body

The server MUST ignore any request body.

3.4.5.3.1.2 Response Body

The response body MUST be a Relying Party Trust (section 2.2.2.6).

3.4.5.3.1.3 Processing Details

On successful authentication the server MUST return a [Service State Data].RelyingPartyTrusts for the relying party trust with [Relying Party Trust].ObjectIdentifier equals to the URI {Identifier} value (section 3.1.1).

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

3.5 Service Configuration Client Details

3.5.1 Abstract Data Model

None.

3.5.2 Timers

None.

3.5.3 Initialization

None.

3.5.4 High-Layer Triggered Events

None.

3.5.5 Message Processing Events and Sequencing Rules

See corresponding section on Server Details.

In all operations where the server requires authenticating the client using client TLS authentication [RFC2246], the client MUST do client TLS authentication [RFC2246] using the certificate in [Proxy Service State Data].TrustCertificate.

3.5.5.1 Proxy/GetConfiguration

See corresponding section on Server Details.

3.5.5.1.1 GET

See corresponding section on Server Details.

3.5.5.1.1.1 Request Body

See corresponding section on Server Details.

3.5.5.1.1.2 Response Body

See corresponding section on Server Details.

3.5.5.1.1.3 Processing Details

If the server response is a HTTP status code of 200 the proxy MUST set [Proxy Service State].Configuration to Configuration obtained in the response.

3.5.5.2 Proxy/RelyingPartyTrusts

See corresponding section on Server Details.

3.5.5.2.1 GET

See corresponding section on Server Details.

3.5.5.2.1.1 Request Body

See corresponding section on Server Details.

3.5.5.2.1.2 Response Body

See corresponding section on Server Details.

3.5.5.2.1.3 Processing Details

None.

3.5.5.3 Proxy/RelyingPartyTrusts/

See corresponding section on Server Details.

3.5.5.3.1 GET

See corresponding section on Server Details.

3.5.5.3.1.1 Request Body

See corresponding section on Server Details.

3.5.5.3.1.2 Response Body

See corresponding section on Server Details.

3.5.5.3.1.3 Processing Details

None.

3.5.6 Timer Events

None.

3.5.7 Other Local Events

None.

3.6 Proxy Configuration Server Details

3.6.1 Abstract Data Model

None.

3.6.2 Timers

None.

3.6.3 Initialization

None.

3.6.4 High-Layer Triggered Events

None.

3.6.5 Message Processing Events and Sequencing Rules

The proxy MAY use the server store to save and retrieve information about the proxy service or about applications published through the proxy. The server provides resources to set and retrieve information based on a key/value pair entry model.

The following resources are available to store custom proxy configuration on the server.

Resource	Description
Proxy/WebApplicationProxy/Store	Resource used to retrieve all entries in the store.
Proxy/WebApplicationProxy/Store/{Key}	Resource used to add, retrieve, remove, or modify an entry in the store.

The responses to all the operations can result in the following status codes.

Status code	Description
200	The operation has succeeded.
400	The request is not valid.
401	Unauthorized for the specified user credentials or for the client TLS certificate.

Status code	Description
404	The object does not exist.
405	Invalid verb used in request (GET, DELETE, POST, PUT).
409	The object already exists.
412	A precondition failed.
500	Version is not specified where required or any other internal error.
501	Version specified (api-version) is invalid (only valid value is 1).

In all operations where the server requires authenticating the proxy using client TLS authentication [RFC2246], the server MUST validate that the certificate presented by the proxy during client TLS authentication can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated, the server MUST return a HTTP error code of 401.

3.6.5.1 Proxy/WebApplicationProxy/Store

The proxy MAY retrieve entries from the store by means of this resource.

3.6.5.1.1 GET

This operation returns a Store Entry List (section 2.2.2.8) containing all entries in the store.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/proxy/WebApplicationProxy/Store?api-version=1
```

The request MUST authenticate using client TLS authentication [RFC2246].

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
409
500
501

3.6.5.1.1.1 Request Body

The server MUST ignore any request body.

3.6.5.1.1.2 Response Body

The response body MUST be a Store Entry List (section 2.2.2.8).

3.6.5.1.1.3 Processing Details

None.

3.6.5.2 Proxy/WebApplicationProxy/Store/

The client MAY use the store to retrieve, add, remove or modify a particular entry from the store by making requests of this resource.

3.6.5.2.1 GET

This operation returns a Store Entry (section 2.2.2.9) containing its version and value.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/proxy/WebApplicationProxy/Store/{Key}?api-version=1
```

The request MUST authenticate using client TLS authentication [RFC2246].

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
500
501

3.6.5.2.1.1 Request Body

The server MUST ignore any request body.

3.6.5.2.1.2 Response Body

The response body MUST be a Store Entry (section 2.2.2.9).

3.6.5.2.1.3 Processing Details

Upon successful authentication the server MUST return the Store Entry (section 2.2.2.9) represented by the object in [Server State].ProxyStore that has a key value with the same string value as {Key}.

If after successful authentication a Store Entry with the same string value as {Key} is not present in [Server State].ProxyStore, the server MUST return an HTTP error code of 404.

3.6.5.2.2 POST

This operation adds a new entry to the store.

The operation is transported by a HTTP **POST** and can be invoked through the following URIs:

```
adfs/proxy/WebApplicationProxy/Store/{Key}?api-version=1
adfs/proxy/PrimaryWriter/WebApplicationProxy/Store/{Key}?api-version=1
```

If the operation is invoked through `adfs/WebApplicationProxy/Store/{Key}`, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through `adfs/proxy/PrimaryWriter/WebApplicationProxy/Store/{Key}`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
409
500
501

3.6.5.2.2.1 Request Body

The request body is a Store Entry Key and Value (section 2.2.2.10).

3.6.5.2.2.2 Response Body

No response body is returned.

3.6.5.2.2.3 Processing Details

On successful authentication the server MUST validate that the URI value of {Key} is the same as the value of [Store Entry Key and Value].key from the request body.

If it is not the same the server MUST return a HTTP error code of 400.

If it is the same the server MUST add the entry to the store by adding Store Entry Key and Value with a version of 1 to [Server State].ProxyStore.

If there is an existing value for the key specified then the server MUST return a HTTP error code of 409.

3.6.5.2.3 PUT

This operation modifies the value of an existing entry in the store.

The operation is transported by a HTTP **PUT** and can be invoked through the following URIs:

```
adfs/proxy/WebApplicationProxy/Store/{Key}?api-version=1
adfs/proxy/PrimaryWriter/WebApplicationProxy/Store/{Key}?api-version=1
```

If the operation is invoked through `adfs/WebApplicationProxy/Store/{Key}`, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through `adfs/proxy/PrimaryWriter/WebApplicationProxy/Store/{Key}`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
404
412
500
501

3.6.5.2.3.1 Request Body

The request body is a Store Entry (section 2.2.2.9).

3.6.5.2.3.2 Response Body

No response body is returned.

3.6.5.2.3.3 Processing Details

On successful authentication the server MUST validate that the URI value of {Key} is the same as the value of [Store Entry].key from the request body.

If it is not the same the server MUST return a HTTP error code of 400.

If it is the same the server MUST find a corresponding Store Entry on [Server State].ProxyStore for the corresponding key.

If it is not found the server MUST return a HTTP error code of 404.

If it is found the server MUST validate that the value [Store Entry].version of the entry found is the same as the value of [Store Entry].version from the request body.

If it is not the same the server MUST return a HTTP error code of 412.

If it is the same the server MUST set the value of [Store Entry].value of the corresponding Store Entry on [Server State].ProxyStore to the [Store Entry].value and MUST increment by 1 its value of [Store Entry].version.

3.6.5.2.4 DELETE

This operation modifies the value of an existing entry in the store.

The operation is transported by a HTTP **PUT** and can be invoked through the following URIs:

```
adfs/proxy/WebApplicationProxy/Store/{Key}?api-version=1
adfs/proxy/PrimaryWriter/WebApplicationProxy/Store/{Key}?api-version=1
```

If the operation is invoked through `adfs/WebApplicationProxy/Store/{Key}`, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through `adfs/proxy/PrimaryWriter/WebApplicationProxy/Store/{Key}`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
401
404
500
501

3.6.5.2.4.1 Request Body

The server MUST ignore any request body.

3.6.5.2.4.2 Response Body

No response body is returned.

3.6.5.2.4.3 Processing Details

On successful authentication the server MUST find a corresponding Store Entry on [Server State].ProxyStore for {Key}.

If it is not found the server MUST return a HTTP error code of 404.

If it is found the server MUST remove the Store Entry from [Server State].ProxyStore.

3.6.6 Timer Events

None.

3.6.7 Other Local Events

None.

3.7 Proxy Configuration Client Details

3.7.1 Abstract Data Model

None.

3.7.2 Timers

None.

3.7.3 Initialization

None.

3.7.4 High-Layer Triggered Events

None.

3.7.5 Message Processing Events and Sequencing Rules

See corresponding section on Server Details.

3.7.5.1 Proxy/WebApplicationProxy/Store

See corresponding section on Server Details.

3.7.5.1.1 GET

See corresponding section on Server Details.

3.7.5.1.1.1 Response Body

See corresponding section on Server Details.

3.7.5.1.1.2 Request Body

See corresponding section on Server Details.

3.7.5.1.1.3 Processing Details

None.

3.7.5.2 Proxy/WebApplicationProxy/Store/

See corresponding section on Server Details.

3.7.5.2.1 GET

See corresponding section on Server Details.

3.7.5.2.1.1 Request Body

See corresponding section on Server Details.

3.7.5.2.1.2 Response Body

See corresponding section on Server Details.

3.7.5.2.1.3 Processing Details

None.

3.7.5.2.2 POST

See corresponding section on Server Details.

3.7.5.2.2.1 Request Body

See corresponding section on Server Details.

3.7.5.2.2.2 Response Body

See corresponding section on Server Details.

3.7.5.2.2.3 Processing Details

None.

3.7.5.2.3 PUT

See corresponding section on Server Details.

3.7.5.2.3.1 Request Body

See corresponding section on Server Details.

3.7.5.2.3.2 Response Body

See corresponding section on Server Details.

3.7.5.2.3.3 Processing Details

None.

3.7.5.2.4 DELETE

See corresponding section on Server Details.

3.7.5.2.4.1 Request Body

See corresponding section on Server Details.

3.7.5.2.4.2 Response Body

See corresponding section on Server Details.

3.7.5.2.4.3 Processing Details

None.

3.7.6 Timer Events

None.

3.7.7 Other Local Events

None.

3.8 Application Publishing Server Details

3.8.1 Abstract Data Model

None.

3.8.2 Timers

None.

3.8.3 Initialization

None.

3.8.4 High-Layer Triggered Events

None.

3.8.5 Message Processing Events and Sequencing Rules

The following resources are available to set the publishing settings to web applications.

Resource	Description
Proxy/RelyingPartyTrusts/{Identity}/PublishingSettings	Resource used to publish a particular web application through the client.

The responses to all the operations can result in the following status codes.

Status code	Description
200	The operation has succeeded.
400	The request is not valid.

Status code	Description
401	Unauthorized for the specified user credentials or for the client TLS certificate.
404	The object does not exist.
405	Invalid verb used in request (GET, DELETE, POST, PUT).
409	The object already exists.
500	Version is not specified where required or any other internal error.
501	Version specified (api-version) is invalid (only valid value is 1).

In all operations where the server requires authenticating the proxy using client TLS authentication [RFC2246], the server MUST validate that the certificate presented by the proxy during client TLS authentication can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated the server MUST return a HTTP error code of 401.

If the operation authenticates using Integrated Windows authentication [RFC2478], the server MUST validate that the authenticated principal is authorized to do the corresponding operation on the server.

3.8.5.1 Proxy/RelyingPartyTrusts/{Identifier}/PublishingSettings

3.8.5.1.1 POST

This operation creates a new set of publishing settings on a relying party trust.

The operation is transported by a HTTP **POST** and can be invoked through the following URIs:

```
adfs/proxy/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1
adfs/proxy/PrimaryWriter/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1
```

If the operation is invoked through `adfs/proxy/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1`, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through `adfs/proxy/PrimaryWriter/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400
401
404

Status code
409
500
501

3.8.5.1.1.1 Request Body

The request body MUST be a Relying Party Trust Publishing Settings (section 2.2.2.7).

3.8.5.1.1.2 Response Body

No response body is returned.

3.8.5.1.1.3 Processing Details

If the publishing settings specified in Relying Party Trust Publishing Settings have been set previously the server MUST return a HTTP error code of 409.

If they have not been set the server MUST add the Relying Party Trust Publishing Settings for the relying party trust identifier with {Identifier}. The server MUST add a new URL to [Service State Data].RelyingPartyTrusts[objectIdentifier:={Identifier}].proxyTrustedEndpoints with the value of [Relying Party Trust Publishing Settings].proxyTrustedEndpointUrl and add a new mapping to [Service State Data].RelyingPartyTrusts[objectIdentifier:={Identifier}].proxyEndpointMappings with the value of [Relying Party Trust Publishing Settings].internalURL to Key and [Relying Party Trust Publishing Settings].externalURL to Value.

3.8.5.1.2 DELETE

This operation removes the publishing settings for a relying party trust.

The operation is transported by a HTTP **DELETE** and can be invoked through the following URIs:

```
adfs/proxy/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1
adfs/proxy/PrimaryWriter/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1
```

If the operation is invoked through `adfs/proxy/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1`, the request MUST authenticate using client TLS authentication [RFC2246].

If the operation is invoked through `adfs/proxy/PrimaryWriter/RelyingPartyTrusts/{Identifier}/PublishingSettings?api-version=1`, the request MUST authenticate using Integrated Windows authentication [RFC2478].

The response message for this operation can result in the following status codes.

Status code
200
400

Status code
401
404
500
501

3.8.5.1.2.1 Request Body

The request body MUST be a Relying Party Trust Publishing Settings (section 2.2.2.7).

3.8.5.1.2.2 Response Body

No response body is returned.

3.8.5.1.2.3 Processing Details

If the publishing settings specified in Relying Party Trust Publishing Settings have not been set previously the server MUST return a HTTP error code of 404.

If they have been set then use the following algorithm for processing this request:

1. If the [Relying Party Trust Publishing Settings].proxyTrustedEndpointUrl is missing or the [Relying Party Trust Publishing Settings].internalUrl is present in the request body, the server MUST return an HTTP error code of 400.

2. If the Relying Party Trust (section 2.2.2.6) with objectIdentifier with the same string value as {Identifier} in [Server State].RelyingPartyTrusts is not found, or if [Service State Data].RelyingPartyTrusts[objectIdentifier:={Identifier}].proxyTrustedEndpoints with the value of [Relying Party Trust Publishing Settings].proxyTrustedEndpointUrl is not found, or if [Relying Party Trust Publishing Settings].externalUrl is specified and an entry with the matching externalUrl is not found in [Service State Data].RelyingPartyTrusts[objectIdentifier:={Identifier}].proxyEndpointMappings, the server MUST return an HTTP error code of 404.

3. On the Relying Party Trust (section 2.2.2.6) [Service State Data].RelyingPartyTrusts[objectIdentifier:={Identifier}], remove the entry from Relying Party Trust Publishing Settings (section 2.2.2.7) that contains in proxyTrustedEndpointUrl the value of [Relying Party Trust Publishing Settings].proxyTrustedEndpointUrl from the request body.

4. If the value of [Relying Party Trust Publishing Settings].externalUrl is present in the request body, remove the entry from [Service State Data].RelyingPartyTrusts[objectIdentifier:={Identifier}].proxyEndpointMappings that has an enternalUrl matching of [Relying Party Trust Publishing Settings].externalUrl from the request body.

3.8.6 Timer Events

None.

3.8.7 Other Local Events

None.

3.9 Application Publishing Client Details

3.9.1 Abstract Data Model

None.

3.9.2 Timers

None.

3.9.3 Initialization

None.

3.9.4 High-Layer Triggered Events

None.

3.9.5 Message Processing Events and Sequencing Rules

See corresponding section on Server Details.

In all operations where the server requires authenticating the client using client TLS authentication [RFC2246], the client MUST use the certificate represented by [Proxy Service State Data].TrustCertificate during client TLS authentication.

3.9.5.1 Proxy/RelyingPartyTrusts/{Identifier}/PublishingSettings

See corresponding section on Server Details.

3.9.5.1.1 POST

See corresponding section on Server Details.

3.9.5.1.1.1 Request Body

See corresponding section on Server Details.

3.9.5.1.1.2 Response Body

See corresponding section on Server Details.

3.9.5.1.1.3 Processing Details

If the server response is a HTTP status code of 200 the proxy MUST add a new identifier object to [Client State].RelyingPartyTrusts with the RelyingPartyTrust.Identifier set to {Identifier}.

3.9.5.1.2 DELETE

See corresponding section on Server Details.

3.9.5.1.2.1 Request Body

See corresponding section on Server Details.

3.9.5.1.2.2 Response Body

See corresponding section on Server Details.

3.9.5.1.2.3 Processing Details

If the server response is a HTTP status code of 200 the proxy MUST remove from [Client State].RelyingPartyTrusts the object with RelyingPartyTrust.Identifier with the same string value as {Identifier}.

3.9.6 Timer Events

None.

3.9.7 Other Local Events

None.

3.10 Proxy Runtime Behaviors Server Details

3.10.1 Abstract Data Model

None.

3.10.2 Timers

None.

3.10.3 Initialization

None.

3.10.4 High-Layer Triggered Events

None.

3.10.5 Message Processing Events and Sequencing Rules

The following resource is available to send a request along with the certificate to the server.

Resource	Description
BackEndProxyTLS	Resource used to obtain a request along with the certificate used for client TLS authentication [RFC2246].

The responses to all the operations can result in the following status codes.

Status code	Description
200	The operation has succeeded.

Status code	Description
400	The request is not valid.
401	Unauthorized for client TLS certificate.
500	Internal error.

In all operations where the server requires authenticating the proxy using client TLS authentication [RFC2246], the server MUST validate that the certificate presented by the client during client TLS authentication can be validated by one of the values of [Service State Data].ProxyTrustCertificates. If the certificate cannot be validated the server MUST return a HTTP error code of 401.

3.10.5.1 BackEndProxyTLS

The proxy MUST support client TLS authentication [RFC2246] on behalf of the server by obtaining the certificate and forwarding it along with the receiving message to the server.

3.10.5.1.1 POST

This operation obtains a request along with a certificate.

The operation is transported by a HTTP **POST** and can be invoked through the following URI:

```
adfs/backendproxytls
```

The server requires authenticating the client using client TLS authentication [RFC2246].

The response message for this operation can result in the following status codes.

Status code
200
400
401
500

3.10.5.1.1.1 Request Body

The request body MUST be a base64url encoded ([RFC4648] section 5) Serialized Request with Certificate (section 2.2.2.11).

3.10.5.1.1.2 Response Body

No response body is returned.

3.10.5.1.1.3 Processing Details

The server MUST treat [Serialized Request with Certificate].SerializedClientCertificate as the certificate of the end user, and SHOULD assume that the client has already verified the original requester’s proof of possession of the private key corresponding to that certificate.

The server MUST process the request as if it was received directly to the endpoint in the server as specified in the request.

3.10.6 Timer Events

None.

3.10.7 Other Local Events

None.

3.11 Proxy Runtime Behaviors Client Details

3.11.1 Abstract Data Model

None.

3.11.2 Timers

None.

3.11.3 Initialization

None.

3.11.4 High-Layer Triggered Events

None.

3.11.5 Message Processing Events and Sequencing Rules

The client SHOULD listen for HTTP requests based on the server characteristics in [Client State].Configuration.

For each object in the EndpointConfiguration (CurrentEndpointConfiguration) element, the client SHOULD do the following:

1. Construct the listen URL based on the following rules and listen for requests on those URLs:

    1. If CurrentEndpointConfiguration.PortType is "HttpPort" then form the URL as "http://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpPort]/[CurrentEndpointConfiguration.Path]"

    2. If CurrentEndpointConfiguration.PortType is "HttpsPort" then form the URL as "https://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpsPort]/[CurrentEndpointConfiguration.Path]"

    3. If CurrentEndpointConfiguration.PortType is "HttpsPortForUserTlsAuth" then form the URL as "https://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpsPortForUserTlsAuth]/[CurrentEndpointConfiguration.Path]"

2. If CurrentEndpointConfiguration.ClientCertificateQueryMode is "QueryAndAccept" then the client SHOULD attempt to retrieve end-user X509 certificate [RFC4158] using client TLS authentication [RFC2246]. If it obtains a certificate the client MUST follow processing in section 3.11.5.1.

3. If CurrentEndpointConfiguration.ClientCertificateQueryMode is "QueryAndRequire" then the client SHOULD attempt to retrieve end-user X509 certificate [RFC4158] using client TLS authentication [RFC2246]. If it obtains a certificate, the client MUST follow the processing in section 3.11.5.1. If it does not obtain a certificate, it SHOULD return a HTTP error code of 204.

4. If no certificate was obtained in steps 2 or 3, then the client SHOULD replay the request as follows:

    1. The request SHOULD be made to the following URL:

        1. If CurrentEndpointConfiguration.ServicePortType is "HttpPort" then form the URL as "http://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpPort]/[CurrentEndpointConfiguration.ServicePath]"

        2. If CurrentEndpointConfiguration.ServicePortType is "HttpsPort" then form the URL as "https://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpsPort]/[CurrentEndpointConfiguration.ServicePath]"

        3. If CurrentEndpointConfiguration.ServicePortType is "HttpsPortForUserTlsAuth" then form the URL as "https://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpsPortForUserTlsAuth]/[CurrentEndpointConfiguration.ServicePath]"

    2. The client SHOULD add the headers in section 2.2.1 to the request.

3.11.5.1 End-user X509 Certificate Processing

If the client obtains a certificate of the end user then the client SHOULD validate the X509 certificate [RFC4158] based on the CurrentEndpointConfiguration.CertificateValidation.

If the CurrentEndpointConfiguration.CertificateValidation value is "None" then no validation SHOULD be performed.

If the CurrentEndpointConfiguration.CertificateValidation value is "Ssl" then the whole chain validation [RFC4158] of the certificate SHOULD be performed.

If the CurrentEndpointConfiguration.CertificateValidation value is "IssuedByDrs" then the client SHOULD validate that the end-user certificate was issued by one of ServiceConfiguration.DeviceCertificateIssuers.

Upon successful validation the client MUST construct a request as in section 3.10.5.1. The [Serialized Request with Certificate].SerializedClientCertificate MUST be set to the base64 string encoded ([RFC4648] section 4) X509 certificate [RFC4158].

If CurrentEndpointConfiguration.CertificateValidation value is "IssuedByDrs" then the [Serialized Request with Certificate].CertificateUsage MUST be set to "Device".

If CurrentEndpointConfiguration.CertificateValidation value is "Ssl" then the [Serialized Request with Certificate].CertificateUsage MUST be set to "User".

The [Serialized Request with Certificate].Request elements values SHOULD be copied from the incoming HTTP request.

The request SHOULD be made to https://[ServiceConfiguration.ServiceHostName]:[ServiceConfiguration.HttpsPort]/adfs/backendproxytls and the client MUST authenticate with client TLS [RFC2246] using [Client State].TrustCertificate.

3.11.6 Timer Events

None.

3.11.7 Other Local Events

None.

3.12 Application Proxy Runtime Behaviors Server Details

3.12.1 Abstract Data Model

None.

3.12.2 Timers

None.

3.12.3 Initialization

None.

3.12.4 High-Layer Triggered Events

None.

3.12.5 Message Processing Events and Sequencing Rules

3.12.5.1 Issue Preauthentication

The server MUST implement the behaviors in this section if and only if the following is met for a particular incoming request:

1. The request contains the header X-MS-Proxy, as defined in section 2.2.1.1.
2. The [Server State].ProxyRelyingPartyTrust.enabled is set to true.
3. The [Relying Party Trust] being preauthenticated exists and has the value of publishedThroughProxy set to true. Note that preauthentication is different for each protocol; refer to subsequent sections for details.

3.12.5.1.1 Proxy Preauthentication

This operation processes a request for authentication and returns a proxy token as described in section 3.13.5.1 upon success.

The operation is transported by a HTTP **GET** and can be invoked through the following URI:

```
adfs/ls?version=1.0&action=signin&realm={web-application-for-client-id}&apprealm={web-application-id}&returnurl={client-url-to-issue-token}
```

The response message for this operation can result in the following status codes.

Status code	Description
200	The operation has succeeded.
403	The access is forbidden.
500	Internal error.

3.12.5.1.1.1 Request Body

The server MUST ignore any request body.

3.12.5.1.1.2 Response Body

No response body is returned.

3.12.5.1.1.3 Processing Details

The server MUST validate that {web-application-for-client-id} corresponds to the value of [Server State].ProxyRelyingPartyTrust.objectIdentifier. If validation fails, the server MUST return a HTTP error code of 500.

The server MUST validate that the request meets the conditions to issue preauthentication (section 3.12.5.1) for the web application in [Server State].RelyingPartyTrusts with objectIdentifier equals to {web-application-id}.

The server MUST validate that the Relying Party Trust (section 2.2.2.6) proxyTrustedEndpoints contains a URL with a scheme, host and port that match those of {client-url-to-issue-token} and that prefix-matches the url-path of {client-url-to-issue-token} (for URL components see [RFC1738] sections 2.1 and 3.1). If validation fails, the server MUST return a HTTP error code of 500.

The server performs authentication of the request based on the server’s authentication policy for [Server State].ProxyRelyingPartyTrust. If authentication fails the server MUST return a HTTP error code of 403.

If authentication succeeds the server MUST return a HTTP status code of 302 with a base64url encoded ([RFC4648] section 5) proxy token (section 3.13.5.1) in the URL query string parameter "authToken".

3.12.5.1.2 SAML-P Extensions for Preauthentication

The server MUST validate that the request meets the conditions to issue preauthentication (section 3.12.5.1) for the web application in [Server State].RelyingPartyTrusts with identifiers containing a string value that matches the <Issuer> element value ([SAMLCore2] section 2.2.5) in the <AuthnRequest> element ([SAMLCore2].

Upon successful authentication ([SAMLCore2] section 3.4.1.4) the server MUST do the following before sending the response to the response URL:

1. Transform the response URL based on the values of [Relying Party Trust].proxyEndpointMappings for the web application by replacing the response URL string portion that matches the Key value (internal URL mapping value) with the value of Value (external URL mapping value). If there is no match the response URL MUST not be changed.

2. If the request is an IdP initiated request the server MUST perform authentication of the request based on the server’s authentication policy for [Server State].ProxyRelyingPartyTrust. If

authentication fails the server MUST respond according to [SAMLCore2] defined behavior for failed authentication.

3. If authentication succeeds the server MUST include in the response URL a query string parameter with name "authToken" with a value of a base64url encoded ([RFC4648] section 5) proxy token (section 3.13.5.1).

The server MUST send the response to the response URL.

3.12.5.1.3 WS-Fed Extensions for Preauthentication

If the server implements [WSFederation1.2] then the server MUST implement the following processing.

The server MUST validate that the request meets the conditions to issue preauthentication (section 3.12.5.1) for the web application in [Server State].RelyingPartyTrusts with identifiers containing a string value that matches the wtrealm query string parameter value.

Upon successful authentication ([WSFederation1.2] section 13.1.1) the server MUST do the following before sending the response to the response URL:

1. Transform the response URL based on the values of [Relying Party Trust].proxyEndpointMappings for the web application by replacing the response URL string portion that matches the Key value (internal URL mapping value) with the value of Value (external URL mapping value). If there is no match the response URL MUST not be changed.

2. If preauthentication has not happened yet<3> the server MUST perform authentication of the request based on the server's authentication policy for [Server State].ProxyRelyingPartyTrust. If authentication fails the server MUST respond according to [WSFederation1.2] defined behavior for failed authentication.

3. If authentication succeeds the server MUST include in the response URL a query string parameter with name "authToken" with a value of a base64url encoded ([RFC4648] section 5) proxy token (section 3.13.5.1).

The server MUST send the response to the response URL.

3.12.5.1.4 OAuth Extensions for Preauthentication

If the server implements [MS-OAPX] then the server MUST implement the following behaviors.

The server MUST validate that the request meets the conditions to issue preauthentication (section 3.12.5.1) for the web application in [Server State].RelyingPartyTrusts with identifiers containing a URI matching the "resource" query string parameter value.

Upon successful authentication [MS-OAPX], the server MUST do the following before sending the response.

The server performs authentication of the request based on the server's authentication policy for [Server State].ProxyRelyingPartyTrust. If authentication fails the server MUST respond according to [MS-OAPX] defined behavior for failed authentication.

If authentication succeeds the server MUST generate a proxy token (section 3.13.5.1). The server MUST take the proxy token and combine it with the token targeted for the application in a [Combined Token] (section 2.2.2.18) and base64url encode ([RFC4648] section 5) the results. The server MUST use this [Combined Token] in all references to "token" in [MS-OAPX].

3.12.6 Timer Events

None.

3.12.7 Other Local Events

None.

3.13 Application Proxy Runtime Behaviors Client Details

3.13.1 Abstract Data Model

None.

3.13.2 Timers

None.

3.13.3 Initialization

None.

3.13.4 High-Layer Triggered Events

None.

3.13.5 Message Processing Events and Sequencing Rules

On receiving any request the client needs to identify if the request is preauthenticated to either allow the access or initiate preauthentication.

3.13.5.1 Preauthentication

A request is preauthenticated if it contains a [Proxy Token] (section 2.2.2.17) signed using JSON Web Signature (JWS) [IETFDRAFT-JWS] with the signing certificate published by the server through the Federation Metadata [WSFederation1.2].

Once a request has been identified as preauthenticated, the proxy MUST allow access by replaying the request to the corresponding internal address without the [Proxy Token].

Other claims might be present as name/value pairs depending on the issuance rules for the proxy configured at the server. It is left to the proxy implementer as to how to use these claims.

3.13.5.1.1 Query String Based Preauthentication

The request is preauthenticated if it contains a valid base64url encoded ([RFC4648] section 5) proxy token (section 3.13.5.1) from the server on the query string parameter "authToken". The token is validated according to section 3.13.5.1.

After successful preauthentication the proxy MUST remove the authToken parameter with its value before replaying the request to the internal URL.

3.13.5.1.2 HTTP Authorization Header Based Preauthentication

If the request contains a HTTP Authorization header with a valid base64URL encoded ([RFC4648] section 5) [Combined Token] (section 2.2.2.18) then request can be preauthenticated by validating [Combined Token].proxy_token as in section 3.13.5.1.

The client MUST use [Combined Token].proxy_token to authorize the access to the web application.

After successful preauthentication the client MUST replace the HTTP Authorization header value with a base64URL encoded ([RFC4648] section 5) value of [Combined Token].access_token before replaying the request to the internal URL.

3.13.5.2 Initiate Preauthentication

If the request does not contain a proxy token then the request is unauthenticated and the client MUST initiate preauthentication.

If the client is servicing a request for the application identified by one of the entries in [Client State].RelyingPartyTrusts then the client MUST initiate preauthentication as follows:

1. If [Relying Party Trust State].RedirectBasedPreauth is "true" then the client MUST follow processing rules in section 3.13.5.2.1.

2. If [Relying Party Trust State].RedirectBasedPreauth is "false" then the client MUST follow processing rules in section 3.13.5.2.2.

3.13.5.2.1 Initiate Redirect-based Preauthentication

Once a request to a web application has been identified as unauthenticated, the proxy MUST initiate preauthentication by returning a HTTP 307 Temporary Redirect message to the client, redirecting the client to the following server end-point URL:

```
"https://" + [Proxy Service State]. Configuration.ServiceConfiguration.ServiceHostName + ":"
+ [Proxy Service State]. Configuration.ServiceConfiguration.HttpsPort + "/adfs/ls"
```

The redirect URL MUST have the following query string parameters.

Parameter	Value
version	Version of the protocol. It MUST be "1.0".
action	Action on authentication request. It MUST be "signin".
realm	Identifier for the proxy relying party trust. It MUST be [Client State].ProxyRelyingPartyTrustIdentifier (section 3.1.1.2).
apprealm	URL of the endpoint of the application being accessed.
returnurl	URL of the incoming request.

3.13.5.2.2 Response to Active Requests

Once a request to a web application has been identified as unauthenticated, the proxy MUST initiate preauthentication. To do this the proxy MUST identify whether the request is from a Microsoft Office application that relies on the Office Forms Based Authentication (OFBA) Protocol [MS-OFBA].

To identify requests from Microsoft Office clients to application services relying on the OFBA protocol, the proxy MUST check if the request is an HTTP OPTIONS with a particular value on the User-Agent HTTP header or with a particular value on the X-Forms_Based_Auth_Accepted HTTP header (any of them):

Header	Value
User-Agent	Any of the following: "Microsoft Data Access Internet Publishing Provider" "Microsoft-WebDAV-MiniRedir" "non-browser" "MSOffice ##" where ## is an integer number "MSOffice XXXX ##" where XXXX is a value of "Word", "Excel", "PowerPoint" and "OneNote" and ## is an integer number "Mozilla/4.0 (compatible; MS FrontPage)" "Microsoft Office Protocol Discovery"
X-Forms_Based_Auth_Accepted	Any of the following: "t"

If the request is from a Microsoft Office client relying on the OFBA protocol, the server MUST return an HTTP error code of 403 to the client with the following headers:

<table>
<tr><th>Header</th><th>Value</th></tr>
<tr><td>X-Forms_Based_Auth_Required</td><td>URL for the sign-in request:
<table>
<tr><th>Parameter</th><th>Value</th></tr>
<tr><td>version</td><td>Version of the protocol. It MUST be "1.0".</td></tr>
<tr><td>action</td><td>Action on authentication request. It MUST be "signin".</td></tr>
<tr><td>realm</td><td>Identifier for the proxy relying party trust. It MUST be [Client State].ProxyRelyingPartyTrustIdentifier (section 3.1.1.2).</td></tr>
<tr><td>apprealm</td><td>URL of the endpoint of the application being accessed.</td></tr>
<tr><td>returnurl</td><td>URL of the incoming request.</td></tr>
</table>
</td></tr>
<tr><td>X-Forms_Based_Auth_Return_Url</td><td>URL of incoming request.</td></tr>
</table>

For requests from non-Microsoft-Office clients accessing services that implement the OFBA protocol [MS-OFBA] that rely on AD FS for authentication, the proxy MUST return an HTTP error code of 401 Unauthorized with the following header.

Header	Value
WWW-	"Bearer authorization_uri=https://" + [Client State]. Configuration.ServiceConfiguration.ServiceHostName + ":" + [Client State].

Header	Value
Authenticate	Configuration.ServiceConfiguration.HttpsPort + "/adfs/oauth2/authorize"

3.13.6 Timer Events

None.

3.13.7 Other Local Events

None.

4 Protocol Examples

4.1 Establishing Proxy Trust with the Server

4.1.1 Client Request

```
POST https://sts1.contoso.com/adfs/Proxy/EstablishTrust HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW5pc3RyYXRvcjpBZHJ1bWJsZUA2
Host: sts1.contoso.com
Content-Length: 2388
Expect: 100-continue

{"SerializedTrustCertificate":"MIIG0zCCBLugAwIBAgITOgAAAAWDWt3Svu3yfgAAAAAABTANBgkqhkiG9w0BAQ
sFADAYMRYwFAYDVQQDEw1tdWFsaWRmdDI3LUNBMB4XDTEzMDcxMjIzMDgxNVoXDTE0MDcxMjIzMDgxNVowbjETMBEGCgm
SJomT8ixkARkWA2NvbTETMBEGCgmSJomT8ixkARkWA2RmdDEaMBgGCgmSJomT8ixkARkWCm11YWxpZGZ0MjcxDjAMBgNV
BAMTBVVzZXJzMRYwFAYDVQQDEw1BZG1pbmlzdHJhdG9yMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsFeNg
BQ9p6c6c9yGeXX9g6TavGJHnAn5hlKTHglBAh\/1mD00+FcN2QD8RB2yWu8kH4uXSUWc2VLAbM095M35o\/U0uh1kJODf
bpOu3KL7rufPMeDUHtLNIxyL91gRxoBEPEKv8okMKmQtQE4DgpY5yFiL3G0EGM4S\/QOZxhiztKP9\/ne6PEu\/rMrdc6
8FoxG+6Hwp3WRgYrV+C5\/7UsD5LlWMWXzxM4TDpTjebvcFS9WKD9wd89sEUpvomRQg1Lj+sXSs\/DVpo8IhbbmYSzN6f
\/WESRKrYJoUDBWyMiGj4CA5mgDvjtBeiawC7YDv4E2i8H2RVGtldJtweoeWs2ij5QIDAQABo4ICvjCCArowFwYJKwYBB
AGCNxQCBAoeCABVAHMAZQByMCkGA1UdJQQiMCAGCisGAQQBgjcKAwQGCCsGAQUFBwMEBggrBgEFBQcDAjAOBgNVHQ8BAf
8EBAMCBaAwRAYJKoZIhvcNAQkPBDcwNTAOBggqhkiG9w0DAgICAIAwDgYIKoZIhvcNAwQCAgCAMAcGBSsOAwIHMAoGCCq
GSIb3DQMHMB0GA1UdDgQWBBRmTByqGxQUzt2gjmhucZrVLai65TAfBgNVHSMEGDAWgBRdxEDXM6dBSWx2luJ+kQ2tiLr4
GDCB1AYDVR0fBIHMMIHJMIHGoIHDoIHAhoG9bGRhcDovLy9DTj1tdWFsaWRmdDI3LUNBLENOPW1kZnRkYyxDTj1DRFAsQ
049UHVibGljJTIwS2V5JTIwU2VydmljZXMsQ049U2VydmljZXMsQ049Q29uZmlndXJhdGlvbixEQz1tdWFsaWRmdDI3LE
RDPWRmdCxEQz1jb20\/Y2VydGlmaWNhdGVSZXZvY2F0aW9uTGlzdD9iYXNlP29iamVjdENsYXNzPWNSTERpc3RyaWJ1dG
lvblBvaW50MIHJBggrBgEFBQcBAQSBvDCBuTCBtgYIKwYBBQUHMAKGgalsZGFwOi8vL0NOPW11YWxpZGZ0MjctQ0EsQ04
9QUlBLENOPVB1YmxpYyUyMEtleSUyMFNlcnZpY2VzLENOPVNlcnZpY2VzLENOPUNvbmZpZ3VyYXRpb24sREM9bXVhbGlk
ZnQyNyxEQz1kZnQsREM9Y29tP2NBQ2VydGlmaWNhdGU\/YmFzZT9vYmplY3RDbGFzcz1jZXJ0aWZpY2F0aW9uQXV0aG9y
aXR5MDsGA1UdEQQ0MDKgMAYKKwYBBAGCNxQCA6AiDCBhZG1pbmlzdHJhdG9yQG11YWxpZGZ0MjcuZGZ0LmNvbTANBgkqh
kiG9w0BAQsFAAOCAgEAp5ZEUswq1\/XH6oLedTwtQSdXraP5SprU6mKk+y5+W6osGicAxEwC183wwnmeXh1XRDJXRsX9U
yDsU3f5jJ94MMI7CR6mjLm88r9y8KxVoXikuBAka9+w2LsyxMunhQcd64JqK2lDCgJiEBti6R7+dZe4GRaDe9JpNPKoI4
RqCQ\/TXc3knQ3MyGSbTkTto1iuaIGsmnmKJ5LGG31tszI1vqvLuK+MavnUdLXKGevCAGqYL6ZvinWOOJCXFjgjEOhuOz
XsjzuPHMkHw0Ji6U8AEfnagQntXNGmEohVVEMFue0aRCmko9ragtFsfGlHXjSUoo5spGNOH9k4pmk4eanmJPGmCBB3DVC
gxjAYuIQvEnSV12Oydu6mOEUuY6oLfnKzIHWqmBqrjj2hAta+sNF\/MSQqt2MVL8\/G67F4W6xPfc+nGgh+1EDo+t6pPJ
UHyFog5CYQ+mRGerq2TcBq\/Qv\/MFwO3t2aEMroXdRW2EDnYogHN25L8xrt37fd3s0+32h\/\/Z8d7cmD5j9h7s7fUqH
dISg5U9b8UwFLH4ZAIGOSEaDP73XPlLs7ic4rNJ88Y4e6LEK1UHcTBG0VNvdPHEVhctBKzhFZG0FI2kr0bfupdURymzxd
EHbExP4HErpGTLvcU7\/S3AcMkz8DOvXzG2CQnevAFkDpN8ne0yOraWwKE8Y="}
```

4.1.2 Server Response

```
HTTP/1.1 200 OK
Content-Length: 0
```

4.2 Getting Information about All Relying Party Trusts

4.2.1 Client Request

```
GET https://sts1.contoso.com/adfs/proxy/relyingpartytrusts?api-version=1 HTTP/1.1
Host: sts1.contoso.com
```

4.2.2 Server Response

```
HTTP/1.1 200 OK
Cache-Control: no-store
```

```
Pragma: no-cache
Content-Length: 469
Content-Type: application/json;charset=UTF-8

[{"enabled":true,"name":"Device Registration
Service","nonClaimsAware":false,"objectIdentifier":"4646dd08-49eb-e211-9867-
00155d6ff01e","publishedThroughProxy":false},{"enabled":true,"name":"fedpassive","nonClaimsAw
are":false,"objectIdentifier":"011ab67d-49eb-e211-9867-
00155d6ff01e","publishedThroughProxy":false},{"enabled":true,"name":"integratedWindowsRp","no
nClaimsAware":true,"objectIdentifier":"071ab67d-49eb-e211-9867-
00155d6ff01e","publishedThroughProxy":true}]
```

4.3 Create a New Set of Published Settings on a Relying Party Trust

4.3.1 Client Request

```
POST https://sts1.contoso.com/adfs/proxy/relyingpartytrusts/7aeee25c-4beb-e211-9867-
00155d6ff01e/fedpassive/publishedsettings?api-version=1 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: sts1.contoso.com
Content-Length: 264
Expect: 100-continue

{"internalUrl":"https://urlInternal","externalUrl":"https://urlExternal","proxyTrustedEndpoin
t":"https://urlExternal"}
```

4.3.2 Server Response

```
HTTP/1.1 200 OK
Content-Length: 0
```

4.4 Remove an Existing Set of Published Settings on a Relying Party Trust

4.4.1 Client Request

```
DELETE https://sts1.contoso.com/adfs/proxy/relyingpartytrusts/0b153cca-4beb-e211-9867-
00155d6ff01e/fedpassive/publishedsettings?api-version=1 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: sts1.contoso.com
Content-Length: 155
Expect: 100-continue

{"externalUrl":"https://urlExternal","proxyTrustedEndpoint":"https://urlExternal"}
```

4.4.2 Server Response

```
HTTP/1.1 200 OK
Content-Length: 0
```

4.5 Add a Key Value Pair to the Store

4.5.1 Client Request

```
POST
https://sts1.contoso.com/adfs/proxy/webapplicationproxy/store/DLOWTTYDQMB2NAPRXFITNYKZXSVW8D7
J0KCQEH0EA?api-version=1 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: sts1.contoso.com
Content-Length: 33
Expect: 100-continue

{"value":"SOMEVALUE_THAT_I_HAVE"}
```

4.5.2 Server Response

```
HTTP/1.1 200 OK
Content-Length: 0
```

4.6 Retrieve a Value of a Key from the Store

4.6.1 Client Request

```
GET https://sts1.contoso.com/adfs/proxy/webapplicationproxy/store/MY_KEY?api-version=1
HTTP/1.1
Host: sts1.contoso.com
```

4.6.2 Server Response

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
Content-Length: 60
Content-Type: application/json;charset=UTF-8

{"key":"MY_KEY","version":0,"value":"SOMEVALUE_THAT_I_HAVE"}
```

4.7 Update the Value of a Key Already in the Store

4.7.1 Client Request

```
PUT https://sts1.contoso.com/adfs/proxy/webapplicationproxy/store/MY_KEY?api-version=1
HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: sts1.contoso.com
Content-Length: 44
Expect: 100-continue

{"value":"ANOTHER VALUE___ NEW","version":0}
```

4.7.2 Server Response

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
Content-Length: 28
Content-Type: application/json;charset=UTF-8

{"key":"MY_KEY","version":1}
```

4.8 Create a new Proxy Relying Party Trust

4.8.1 Client Request

```
POST https://sts1.contoso.com/adfs/proxy/webapplicationproxy/trust?api-version=1 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: sts1.contoso.com
Content-Length: 35
Expect: 100-continue

{"Identifier":"https:\/\/appProxy"}
```

4.8.2 Server Response

```
HTTP/1.1 200 OK
Content-Length: 0
```

4.9 Get the Proxy Relying Party Trust

4.9.1 Client Request

```
GET https://sts1.contoso.com/adfs/proxy/webapplicationproxy/trust?api-version=1 HTTP/1.1
Host: sts1.contoso.com
```

4.9.2 Server Response

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
Content-Length: 35
Content-Type: application/json;charset=UTF-8

{"Identifier":"https:\/\/appProxy"}
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full JSON Schema

```
{
  "title" : "Proxy Trust",
  "type" : "object",
  "properties" :
  {
    "SerializedTrustCertificate" : {"type" : "string"}
  }
}

{
  "title" : "Proxy Trust Renewal",
  "type" : "object",
  "properties" :
  {
    "SerializedReplacementCertificate" : {"type" : "string"}
  }
}

{
  "title" : "Proxy Relying Party Trust",
  "type" : "object",
  "properties" :
  {
    "Identifier" : {"type" : "string"}
  }
}

{
  "title" : "Configuration",
  "type" : "object",
  "properties" :
  {
    "ServiceConfiguration" :
    {
      "type" : "object",
      "properties" :
      {
        "ServiceHostName" : {"type" : "string"},
        "HttpPort" : {"type" : "integer"},
        "HttpsPort" : {"type" : "integer"},
        "HttpsPortForUserTlsAuth" : {"type" : "integer"},
        "DeviceCertificateIssuers" :
        {
          "type" : "array",
          "items" : {"type" : "string"}
        },
        "ProxyTrustCertificateLifetime" : {"type" : "integer"}
      }
    },
    "EndpointConfiguration" :
    {
      "type" : "array",
      "items" :
      {
        "type" : "object",
        "properties" :
```

```
        {
          "Path" : {"type" : "string"},
          "PortType" :
          {
            "enum" : ["HttpPort", "HttpsPort", "HttpsPortForUserTlsAuth"]
          },
          "AuthenticationSchemes" :
          {
            "enum" : [8, 32768]
          },
          "ClientCertificateQueryMode" :
          {
            "enum" : ["None", "QueryAndAccept", "QueryAndRequire"]
          },
          "CertificateValidation" :
          {
            "enum" : ["None", "User", "Device"]
          },
          "ServicePath" : {"type" : "string"},
          "ServicePortType" :
          {
            "enum" : ["HttpPort", "HttpsPort", "HttpsPortForUserTlsAuth"]
          }
        }
      }
    }
  }
}

{
  "title" : "Relying Party Trust List",
  "type" : "object",
  "properties" :
  {
    "relyingPartyTrustListArray" :
    {
      "type" : "array",
      "items" :
      {
        "type" : "object",
        "properties" :
        {
          "objectIdentifier" : {"type" : "string"},
          "name" : {"type" : "string"},
          "publishedThroughProxy" : {"type" : "boolean"},
          "nonClaimsAware" : {"type" : "boolean"},
          "enabled" : {"type" : "boolean"}
        }
      }
    }
  }
}

{
  "title" : "Relying Party Trust",
  "type" : "object",
  "properties" :
  {
    "objectIdentifier" : {"type" : "string"},
    "name" : {"type" : "string"},
    "publishedThroughProxy" : {"type" : "boolean"},
    "nonClaimsAware" : {"type" : "boolean"},
    "enabled" : {"type" : "boolean"},
    "identifiers" :
    {
      "type" : "array",
      "items" : {"type" : "string"}
    },
    "proxyTrustedEndpoints" :
    {
      "type" : "array",
      "items" : {"type" : "string"}
    },
    "proxyEndpointMappings" :
    {
      "type" : "array",
      "items" :
      {
        "type" : "object",
        "properties" :
        {
          "Key" : {"type" : "string"},
          "Value" : {"type" : "string"}
        }
      }
    }
  }
}


{
  "title" : "Relying Party Trust Publishing Settings",
  "type" : "object",
  "properties" :
  {
    "externalUrl" : {"type" : "string"},
    "internalUrl" : {"type" : "string"},
    "proxyTrustedEndpointUrl" : {"type" : "string"}
  }
}

{
  "title" : "Store Entry List",
  "type" : "object",
  "properties" :
  {
    "storeEntryListArray" :
    {
      "type" : "array",
      "items" : {"type" : "Store Entry"}
    }
  }
}

{
  "title" : "Store Entry",
  "type" : "object",
  "properties" :
```

```
  {
    "key" : {"type" : "string"},
    "version" : {"type" : "integer"},
    "value" : {"type" : "string"}
  }
}

{
  "title" : "Store Entry Key and Value",
  "type" : "object",
  "properties" :
  {
    "key" : {"type" : "string"},
    "value" : {"type" : "string"}
  }
}

{
  "title" : "Serialized Request with Certificate",
  "type" : "object",
  "properties" :
  {
    "Request" :
    {
      "type" : "object",
      "properties" :
      {
        "AcceptTypes" : {"type" : "string"},
        "ContentEncoding" : {"type" : "string"},
        "ContentLength" : {"type" : "integer"},
        "ContentType" : {"type" : "string"},
        "Cookies" :
        {
          "type" : "object",
          "properties" :
          {
            "Name" : {"type" : "string"},
            "Value" : {"type" : "string"},
            "Path" : {"type" : "string"},
            "Domain" : {"type" : "string"},
            "Expires" : {"type" : "integer"},
            "Version" : {"type" : "integer"}
          }
        },
        "Headers" :
        {
          "type" : "array",
          "items" :
          {
            "type" : "object",
            "properties" :
            {
              "Name" : {"type" : "string"},
              "Value" : {"type" : "string"}
            }
          }
        },
        "HttpMethod" : {"type" : "string"},
        "RequestUri" : {"type" : "string"},
        "QueryString" : {"type" : "string"},
        "UserAgent" : {"type" : "string"},
        "UserHostAddress" : {"type" : "string"},
        "UserHostName" : {"type" : "string"},
        "UserLanguages" : {"type" : "string"}
      }
    },
    "SerializedClientCertificate" : {"type" : "string"},
    "CertificateUsage" :
    {
      "enum" : ["User", "Device"]
    }
  }
}

{
  "title" : "Proxy Token",
  "type" : "object",
  "properties" :
  {
    "ver" : {"type" : "number"},
    "aud" : {"type" : "string"},
    "iat" : {"type" : "integer"},
    "exp" : {"type" : "integer"},
    "iss" : {"type" : "string"},
    "relyingpartytrustid" : {"type" : "string"},
    "deviceregid" : {"type" : "string"},
    "authinstant" : {"type" : "integer"},
    "authmethod" : {"type" : "string"},
    "upn" : {"type" : "string"}
  }
}


{
  "title" : "Combined Token",
  "type" : "object",
  "properties" :
  {
    "proxy_token" : {"type" : "Proxy Token"},
    "access_token" : {"type" : "string"}
  }
}
```

7 Appendix B: 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 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 3.1.1.1: Any writes to [Server State] require, by default, 5 minutes to propagate to other nodes in the server in an AD FS farm configuration using WID.

<2> Section 3.3.5.2.1.3: Windows does not remove the old certificate from [Server State].

<3> Section 3.12.5.1.3: Windows validates that the sign-in request comes from a SAML-P IdP initiated request with a query string parameter RelayState containing an identifier of a web application in the server that relies on the WS-Fed protocol for authentication.

8 Change Tracking

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

9 Index