[MS-ADFSOAL-Diff]:

Active Directory Federation Services OAuth Authorization Code Lookup Protocol

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might 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 that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
8/8/2013	1.0	New	Released new document.
11/14/2013	1.1	Minor	Clarified the meaning of the technical content.
2/13/2014	2.0	Major	Significantly changed the technical content.
5/15/2014	2.0	None	No change to the meaning, language, or formatting of the technical content.
6/30/2015	3.0	Major	Significantly changed the technical content.
7/14/2016	4.0	Major	Significantly changed the technical content.
6/1/2017	4.0	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	5.0	Major	Significantly changed the technical content.
9/12/2018	6.0	Major	Significantly changed the technical content.
4/7/2021	7.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Active Directory Federation Services OAuth Authcode Lookup Protocol is defined as a RESTful protocol API.

In addition to the terms specified in section 1.1, the following terms are used in this document:

From [RFC6749]:

- access token
- access token request
- access token response
- authorization code
- authorization code grant
- authorization request
- authorization response
- authorization server
- client identifier
- redirection URI
- refresh token

From [MS-ADFSWAP]:

- relying party

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

Active Directory: The Windows implementation of a general-purpose directory service, which uses LDAP as its primary access protocol. Active Directory stores information about a variety of objects in the network such as user accounts, computer accounts, groups, and all related credential information used by Kerberos [MS-KILE]. Active Directory is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS), which are both described in [MS-ADOD]: Active Directory Protocols Overview.

Active Directory Domain Services (AD DS): A directory service (DS) implemented by a domain controller (DC). The DS provides a data store for objects that is distributed across multiple DCs. The DCs interoperate as peers to ensure that a local change to an object replicates correctly across DCs. AD DS is a deployment of Active Directory [MS-ADTS].

Active Directory Federation Services (AD FS): A Microsoft implementation of a federation services provider, which provides a security token service (STS) that can issue security tokens to a caller using various protocols such as WS-Trust, WS-Federation, and Security Assertion Markup Language (SAML) version 2.0.

Active Directory Federation Services (AD FS) farm: A collection of AD FS servers that is typically maintained by an enterprise to obtain greater redundancy and offer more reliable service than a single standalone AD FS server.

AD FS farm with shared artifact store: A type of AD FS farm deployment in which all AD FS servers that are part of the farm use a shared artifact store. The protocol defined in this document is not applicable to and is not used in this type of AD FS farm deployment.

AD FS farm with standalone artifact store: A type of AD FS farm deployment in which each AD FS server that is part of the farm has its own local artifact store that is intended for its exclusive use and is not shared with any other member of the farm. The protocol defined in this document is applicable to this type of AD FS farm deployment.

artifact: An object that is created by an AD FS server when it successfully processes an OAuth client's request for authorization. An artifact object is generated along with the OAuth authorization code. Before issuing an OAuth authorization code to the OAuth client, the AD FS server stores the artifact object in its artifact store. The format of the artifact is defined in section 2.2.4.2.

artifact lifetime: Determines the duration for which an artifact that was generated by an AD FS server is valid and persisted in the artifact store. For details, see section 3.2.2.

artifact store: A local store used by an AD FS server to persist artifacts it has generated after successfully processing an OAuth authorization request.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the GUID. See also universally unique identifier (UUID).

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 (Updated Section) 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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://~~www2~~publications.opengroup.org~~/ogsys/catalog~~/c706

Note Registration is required to download the document.

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

[MS-ADA2] Microsoft Corporation, "Active Directory Schema Attributes M".

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

[MS-ADSC] Microsoft Corporation, "Active Directory Schema Classes".

[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

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

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

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

1.2.2 Informative References

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

[RFC4559] Jaganathan, K., Zhu, L., and Brezak, J., "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006, http://www.rfc-editor.org/rfc/rfc4559.txt

1.3 Overview

Active Directory Federation Services (AD FS) servers can be deployed in AD FS farm configurations, often behind a load-balancer, for increased scalability and reliability. The AD FS server implements the authorization server role for the OAuth 2.0 authorization framework and supports the Authorization Code Grant as defined in [RFC6749].

[RFC6749] section 4.1 illustrates the steps required to implement the Authorization Code Grant. In cases where AD FS servers are deployed in an AD FS farm with standalone artifact store, an OAuth client can receive an OAuth authorization code from any AD FS server that is part of the AD FS farm. Subsequently, when the OAuth client attempts to redeem that authorization code for an access token according to the steps outlined in [RFC6749] section 4.1, it might be redirected to a different member of the AD FS farm. Thus, a server that belongs to the AD FS farm with standalone artifact store might receive a request to redeem an OAuth authorization code that was issued by another server in the farm. Under these circumstances, the AD FS servers use the Active Directory Federation Services OAuth Authcode Lookup (ADFSOAL) Protocol defined in this specification in order to detect which server in the farm issued the authorization code and to retrieve the corresponding artifact from that server. The artifact thus retrieved contains the OAuth access token that is thereafter provided to the OAuth client in response to its request to redeem the OAuth authorization code. Note that the ADFSOAL Protocol does not apply to AD FS servers that are deployed in an AD FS farm with shared artifact store.

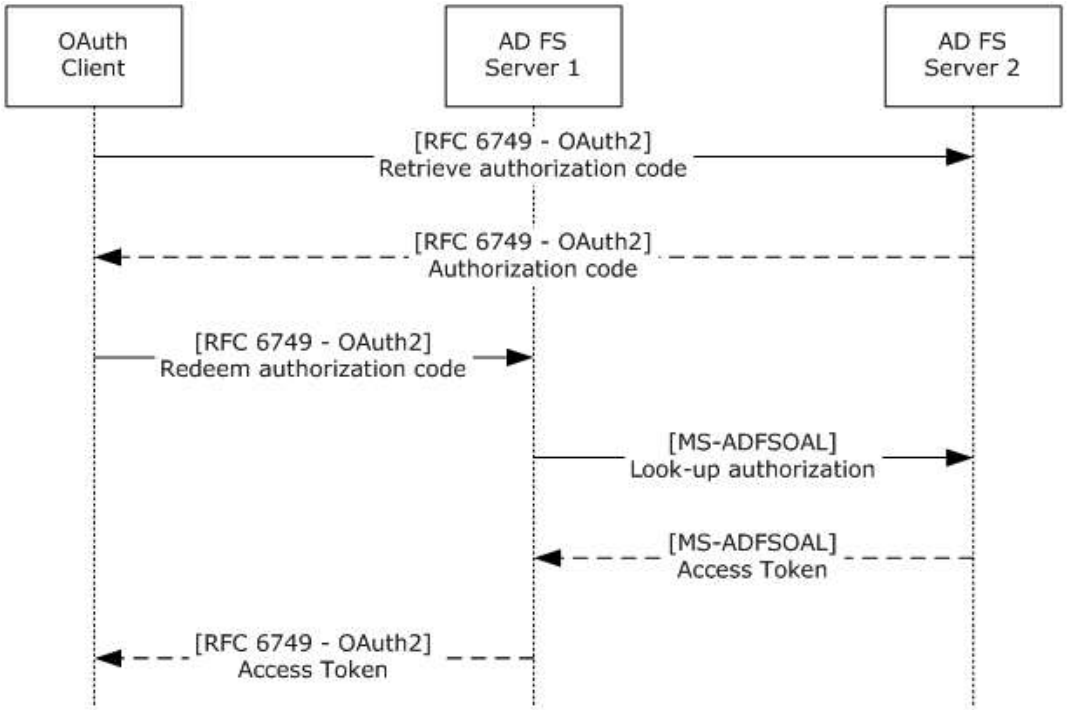

Figure 1: Sequence diagram for the ADFSOAL Protocol

The above sequence diagram illustrates this flow. Two servers "AD FS server 1" and "AD FS server 2" are deployed in an AD FS farm with standalone artifact store configuration. In this scenario, the OAuth client initially received an OAuth authorization code from "AD FS server 2" using the steps defined in [RFC6749]. The OAuth client then attempts to redeem this OAuth authorization code for an access token by using the mechanism defined in [RFC6749]. However, the OAuth client is connected to "AD FS server 1", a different server than the one that originally issued the OAuth authorization code. In this scenario, "AD FS server 1" uses the ADFSOAL Protocol defined in this document to look up the artifact identifier contained within the OAuth authorization code on "AD FS server 2" and to retrieve the corresponding artifact from the artifact store on "AD FS server 2". The artifact thus retrieved contains the OAuth access token that is thereafter returned to the OAuth client in response to its access token request, according to the procedure defined in [RFC6749].

The ADFSOAL Protocol defines a client role and a server role. The client role of the ADFSOAL Protocol corresponds to the AD FS server that is part of the AD FS farm and needs to look up the artifact identifier contained within the authorization code presented to it by the OAuth client. The server role of the ADFSOAL Protocol corresponds to the AD FS server that is part of the same AD FS farm and originally issued the authorization code to the OAuth client.

1.4 Relationship to Other Protocols

The ADFSOAL Protocol depends on HTTP [RFC2616].

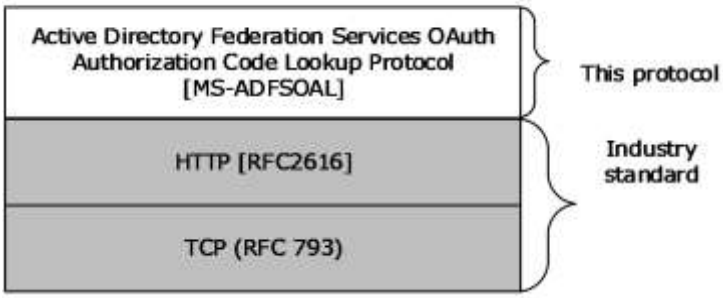

Figure 2: Protocol dependency

1.5 Prerequisites/Preconditions

- The ADFSOAL Protocol is used only when AD FS servers are deployed in an AD FS farm with standalone artifact store configuration. In this deployment configuration, all AD FS servers that belong to the farm have their own artifact store.
- AD FS servers deployed in an AD FS farm with shared artifact store configuration do not use the ADFSOAL Protocol because they have a single artifact store shared by the entire farm and can therefore service OAuth token requests for OAuth authorization codes that were issued by any of the members of the farm.
- The server side of the ADFSOAL Protocol is available on a REST endpoint called the "OAuth artifact lookup REST endpoint" hosted by the AD FS server. The URL of this endpoint can be determined from the hostname of the AD FS server.
- The OAuth artifact lookup REST endpoint on the AD FS server is secured by using Integrated Windows Authentication and is restricted to allow access only from the AD FS server's service account. It is assumed that Integrated Windows Authentication has been established at a lower layer by using [RFC4559] before the protocol defined in this document begins functioning.

1.6 Applicability Statement

The ADFSOAL Protocol was designed to support AD FS servers deployed in an AD FS farm with standalone artifact store configuration.

The ADFSOAL Protocol is not required for stand-alone (non-farm) AD FS server deployments. It is also not required for scenarios where AD FS servers are deployed in an AD FS farm with shared artifact store configuration.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

Supported Transports: The ADFSOAL Protocol supports only HTTP.

Protocol Versions: Versioning for the ADFSOAL Protocol is defined using the mandatory "api-version" query parameter defined in section 2.2.3.1. The only supported version is "1".

Localization: The ADFSOAL Protocol does not return localized strings.

Capability Negotiation: The ADFSOAL Protocol does not support capability negotiation.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The HTTP protocol [RFC2616] MUST be used as the transport.

2.2 Common Data Types

2.2.1 HTTP Methods

The ADFSOAL Protocol does not define any custom HTTP methods in addition to the existing set of standard HTTP methods.

2.2.2 HTTP Headers

The messages exchanged in the ADFSOAL Protocol use the following HTTP headers in addition to the existing set of standard HTTP headers.

Header	Description
client-request-id	This optional header is used to specify a request identifier that is used when logging errors or failures that occur while processing the request.

2.2.2.1 client-request-id

The **client-request-id** HTTP header is optional and MAY be specified by the client role of the ADFSOAL Protocol. This header is used to provide the server role a unique request ID, which is then used by the server of the ADFSOAL Protocol to log error messages that were encountered while processing that lookup request. The value of the **client-request-id** HTTP header MUST be a globally unique identifier (GUID) in standard string representation (see [C706] section 3.1.17 (String UUID) for the format).

Note The **client-request-id** HTTP header and the *client-request-id* query parameter defined in section 2.2.3.2 are designed to be mutually exclusive. The client role of the ADFSOAL Protocol SHOULD use either the HTTP header or the query parameter. If both are specified, the server role of the ADFSOAL Protocol gives precedence to the *client-request-id* query parameter and ignores the value of the **client-request-id** HTTP header.

The format for the **client-request-id** HTTP header is as follows.

```
String = *(%x20-7E)
client-request-id = String
```

2.2.3 Common URI Parameters

The following table summarizes the set of common query parameters defined by this specification.

URI parameter	Description
api-version	This query parameter MUST be present and is used to specify the version of the protocol.
client-	This query parameter is optional and is used to specify a request identifier, which is used when

URI parameter	Description
request-id	logging errors or failures that occur while processing the request.

2.2.3.1 api-version

```
GET http://server/adfs/artifact/{artifactId}?api-version={version} HTTP/1.1
```

The *api-version* query parameter MUST be present and is used to specify the version of the protocol requested by the client of the ADFSOAL Protocol.<1>

The format of the *api-version* query parameter is as follows.

```
String = *(%x20-7E)
api-version = String
```

2.2.3.2 client-request-id

```
GET http://server/adfs/artifact/{artifactId}?api-version={version}&client-request-
id={ClientRequestId} HTTP/1.1
```

The *client-request-id* query parameter is optional and can be specified by the client role of the ADFSOAL Protocol. This parameter is used to provide a request identifier to the server role of the ADFSOAL Protocol, which is then used by the server role of the ADFSOAL Protocol to log error messages that were encountered while processing the corresponding lookup request. The value of the *client-request-id* query parameter MUST be a globally unique identifier (GUID) in standard string representation (see [C706] section 3.1.17 (String UUID) for the format).

The format of the *client-request-id* query parameter is as follows.

```
String = *(%x20-7E)
client-request-id = String
```

2.2.4 Complex Types

The following table summarizes the set of complex type definitions included in this specification.

Complex type	Description
AuthorizationCode	An authorization code [RFC6749] that is issued by an AD FS server to the OAuth client that requests authorization.
Artifact	An object that stores information corresponding to an authorization code issued by an AD FS server.

2.2.4.1 AuthorizationCode

The authorization code is a concatenated string with the following format:

```
issuerGuid.artifactId.signature
```

The authorization code contains a combination of three components with a '.' (period) delimiter:

- **issuerGuid:** A base64 URL encoded ([RFC4648] section 5) string that contains the machine globally unique identifier (GUID) of the AD FS server that issued this authorization code.
- **artifactId:** A base64 URL encoded string that contains the identifier of the artifact that corresponds to this authorization code. The value of the **artifactId** field MUST be unique across all artifact objects (section 2.2.4.2) that are stored in the artifact store of a particular AD FS server.
- **signature:** A base64 URL encoded string that contains a signature over the **issuerGuid** and the **artifactId** fields that can be verified by the server role of the ADFSOAL Protocol.

2.2.4.2 Artifact

The artifact object is created by an AD FS server when it successfully processes an OAuth client's request for authorization, and is generated along with the OAuth authorization code. Before issuing an OAuth authorization code to the OAuth client, the AD FS server stores the artifact object in its artifact store.

Subsequently, when the OAuth client requests an access token by using the authorization code as specified in [RFC6749], the AD FS server processing the request extracts the artifact identifier from the authorization code that was presented by the OAuth client, and also determines which AD FS server issued that authorization code. If the authorization code was issued by the server processing the request, the server examines its local artifact store for an artifact object corresponding to the authorization code.

If the authorization code was issued by another AD FS server in the farm, the server processing the OAuth client's token request uses the ADFSOAL Protocol to look up the authorization code on the AD FS server that issued it. If the authorization code was found on the other AD FS server, the artifact object is returned to the calling AD FS server, that is, to the AD FS server processing the token request. After performing required validation as specified in section 3.1.5.1.1.3, the AD FS server processing the token request responds to the OAuth client with the access token contained in the artifact object.

The artifact object contains the following fields.

```
{
        "description" : "artifact object",
        "type" : "object",
        "properties" :
        {
            "id":
            {
                  "type":"array",
                  "optional":false,
                  "items" : { "type" : "integer", "minimum": 0, "maximum":255}
            },
            "clientId": {"type":"string", "optional":false},
            "redirectUri": {"type":"string", "optional":false},
            "relyingPartyIdentifier": {"type":"string", "optional":false},
            "data": {"type":"string", "optional":false}
        }
}
```

id: The identifier for the artifact. This field contains the same value as the **artifactId** field of the corresponding authorization code (section 2.2.4.1).

clientId: The client identifier [RFC6749] for the OAuth client that originally requested the OAuth authorization code to which this artifact corresponds.

redirectUri: The redirection URI [RFC6749] specified by the OAuth client that originally requested the OAuth authorization code to which this artifact corresponds.

relyingPartyIdentifier: The identifier for the relying party for which the OAuth client originally requested the OAuth authorization code to which this artifact corresponds.

data: Contains the access token and other auxiliary information that was issued by the AD FS server that generated the OAuth authorization code to which this artifact corresponds.

The **data** field of the artifact object is a JavaScript Object Notation (JSON) formatted string that adheres to the following structure, as defined in [RFC6749] section 4.1.4.

```
{
    "access_token": {"type":"string", "optional":false},
    "token_type": {"type":"string", "optional":false},
    "expires_in": {"type":"int", "optional":false},
    "refresh_token": {"type":"string", "optional":true},
}
```

2.2.5 ErrorDetails

This object contains a collection of human-readable details that describe an error encountered by the server role of the ADFSOAL Protocol. It can be used by the client role of the ADFSOAL Protocol for logging purposes or for providing information to an administrator. This object contains the following fields.

```
{
    "description" : "error details",
    "type" : "object",
    "properties" :
    {
        "message": {"type":"string", "optional":true},
        "type": {"type":"string", "optional":true},
        "id": {"type":"string", "optional":true},
        "debugInfo": {"type":"string", "optional":true}
    }
}
```

message: A text message explaining the error.

type: The type of the error encountered.

id: An identifier assigned to the error by the server role of the ADFSOAL Protocol.

debugInfo: Additional information regarding where and how the error occurred. This information is implementation-specific.

2.3 Directory Service Schema Elements

The protocol accesses the following Directory Service schema classes and attributes.

For the syntactic specifications of the following <Class> or <Class><Attribute> pairs, refer to Active Directory Domain Services (AD DS) [MS-ADA1] [MS-ADA2] [MS-ADA3] [MS-ADSC].

Class	Attribute
Computer	objectGUID dNSHostName

3 Protocol Details

3.1 OAuthAuthorizationCodeLookup Client Details

The "client role" of the protocol corresponds to the AD FS server that needs to retrieve an access token, corresponding to an OAuth authorization code presented to it by the OAuth client, from the AD FS server that originally issued the authorization code. In the client role of this protocol, an AD FS server looks up the authorization code presented to it by an OAuth client and determines which AD FS server in its farm originally issued that authorization code. Thereafter, using the ADFSOAL Protocol, the AD FS server in the client role issues an HTTP **GET** request to the AD FS server in the "server role" of the ADFSOAL Protocol in order to look up the OAuth authorization code. If the request is successful, the AD FS server implementing the server role returns the corresponding access token in the HTTP **GET** response.

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

The server implementing the client role of the ADFSOAL Protocol must be able to connect to Active Directory and perform the queries referenced in section 3.1.5.1.1.3.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

Resource	Description
http://server/adfs/artifact/{artifactId}?api-version={version}	The artifact identifier that the AD FS server implementing the server role of the ADFSOAL Protocol must look up in its artifact store.

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

Status code	Reason phrase	Description
200	OK	An artifact corresponding to the given artifact identifier was found by the AD FS server implementing the server role of the ADFSOAL Protocol in its artifact store. In this case, the AD FS server implementing the server role also returns the corresponding artifact in the response.
404	NOT FOUND	An artifact corresponding to the given artifact identifier was NOT found by the AD FS server implementing the server role of the ADFSOAL Protocol in its artifact store.
401	Unauthorized / Access Denied	The client did not provide credentials, or the credentials provided were incorrect.

Status code	Reason phrase	Description
501	Missing VERSION PARAMETER Unknown Version Parameter	The version requested by the client is not implemented.
500	Internal Server Error	The server encountered an error while trying to process the request.

The request messages for these methods do not use any custom HTTP headers.

The response messages for these methods do not use any custom HTTP headers.

The request body for messages to this service, unless otherwise noted, has the same encoding rules.

The response body for messages from this service, unless otherwise noted, has the same encoding rules.

3.1.5.1 http://server/adfs/artifact/{artifactId}?api-version={version}

The {artifactId} component of the URI corresponds to the identifier of the artifact that the AD FS server implementing the client role of the ADFSOAL Protocol needs to look up on the AD FS server implementing the server role of the ADFSOAL Protocol.

artifactId: A base64 URL encoded ([RFC4648] section 5) blob that contains the artifact identifier, which is extracted from the authorization code and needs to be looked up by the server role of the ADFSOAL Protocol. The format of the authorization code is defined in section 2.2.4.1.

The following HTTP methods are allowed to be performed on this resource:

HTTP method	Description
GET	Look up the artifact corresponding to the {artifactId} identifier.

3.1.5.1.1 GET

This method is transported by an HTTP GET.

The method can be invoked through the following URI:

```
http://server/adfs/artifact/{artifactId}?api-version={version}
```

The URI parameters supported for the **GET** request are the common URI parameters documented in section 2.2.3 (Common URI Parameters).

The request message for this method does not contain any custom HTTP headers.

The response message for this method does not contain any custom HTTP headers.

The response message for this method can result in the status codes defined in the status table in section 3.1.5.

3.1.5.1.1.1 Request Body

None.

3.1.5.1.1.2 Response Body

If an artifact corresponding to the artifact identifier that was specified in the request was found in the artifact store on the AD FS server implementing the server role of the ADFSOAL Protocol, the HTTP 200 status code is returned. Additionally, the response body for the **GET** response contains the JSON-formatted artifact object. The format of the JSON artifact object that is returned in the response body is defined in section 2.2.4.2 (Artifact) of this document.

If an artifact corresponding to the artifact identifier specified in the request was not found in the artifact store on the AD FS server implementing the server role of the ADFSOAL Protocol, the HTTP 404 status code is returned. The response body for the **GET** response is empty in this case.

3.1.5.1.1.3 Processing Details

When an AD FS server receives a request from an OAuth client to redeem an OAuth authorization code, it performs the following operations before determining whether to look up the authorization code on another AD FS server in its AD FS farm:

- It extracts the **issuerGuid** and the **artifactId** from the given authorization code. The format of the authorization code is defined in section 2.2.4.1:
- If the **issuerGuid** is null or empty or corresponds to its own machine GUID, the AD FS server does not invoke the ADFSOAL Protocol. This means that the received OAuth authorization code was originally issued by the AD FS server itself and therefore there is no need to look up the artifact identifier on another AD FS server.
- If the **issuerGuid** does not match the above criteria, the AD FS server queries Active Directory (for the attributes defined in section 2.3) to find the computer account whose **objectGUID** matches the value of the **issuerGuid** that was extracted from the received OAuth authorization code.
    - If a corresponding computer account was found in Active Directory, the following steps are taken.
        1. The AD FS server implementing the client role of the ADFSOAL Protocol, that is, the AD FS server that received the token request from the OAuth client, determines the **dnsHostName** from the AD computer object (section 2.3). This is the AD FS server that originally issued the OAuth authorization code.
        2. The AD FS server implementing the client role of the ADFSOAL Protocol then issues an HTTP **GET** request to the AD FS server identified in step 1 using the protocol described by this document in order to look up the artifact identifier (**artifactId**) and retrieve a corresponding artifact. If the server was able to complete the lookup operation successfully, an artifact object is returned in the HTTP **GET** response. The format of the artifact returned in the HTTP **GET** response is documented in section 2.2.4.2 of this document.
        3. The contents of the data field (section 2.2.4.2) in the artifact received in the HTTP **GET** response from the AD FS server implementing the server role of the ADFSOAL Protocol is then returned to the OAuth client in accordance with the requirements of [RFC6749] section 5.1 (Successful Response).
    - If a corresponding computer account was not found in Active Directory, the AD FS server that received the token request from the OAuth client responds to the OAuth client with an invalid_grant error as specified in [RFC6749] section 5.2 (Error Response).

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 OAuthAuthorizationCodeLookup Server Details

The "server role" of the protocol corresponds to the AD FS server that receives an HTTP GET request to lookup an artifact identifier from another AD FS server in its farm. The request is authenticated by using Integrated Windows Authentication as described in section 1.5. In the server role of the ADFSOAL Protocol, the AD FS server first checks to see if the request was sent by another AD FS server in its farm. After the request is authenticated, the AD FS server in the server role of the ADFSOAL Protocol checks its artifact store to see if the received artifact identifier corresponds to an authorization code that was originally issued by it to an OAuth client. If the authorization code was originally issued by the AD FS server, it has a corresponding artifact stored in its artifact store. This JSON formatted artifact is then base64 URL encoded and returned in the HTTP GET response to the caller, that is, to the AD FS server that implements the client role of the ADFSOAL Protocol.

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains 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.

Artifact Store: An artifact store that is used by an AD FS server to store the artifact created when the server successfully processes an OAuth client's request for authorization. The artifact is stored for the duration of the artifact lifetime. See section 2.2.4.2 for the definition of an artifact's data structure.

3.2.2 Timers

ArtifactExpiryTimer: Artifacts are stored in the artifact store for a period corresponding to the artifact lifetime. The server MUST delete artifacts older than the artifact lifetime from its **Artifact Store** ADM element. The artifact lifetime SHOULD be defined as 10 minutes and MUST be equivalent to the OAuth authorization code lifetime, as defined in [RFC6749] section 4.1.2. This timer is used to delete artifacts older than the artifact lifetime.

3.2.3 Initialization

When the protocol is first initialized, the AD FS server must have access to its **Artifact Store** ADM element, where it stores state about OAuth authorization codes issued by it for the duration of the artifact lifetime. The AD FS server stores an artifact, whose format is defined in section 2.2.4.2. The artifact identifier is used to look up and retrieve the artifact from the artifact store. Therefore, the AD FS server MUST ensure that the artifact identifier is unique across its **Artifact Store** ADM element. Access to the **Artifact Store** ADM element must be initialized before the AD FS server services requests by using the ADFSOAL Protocol.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Resource	Description
http://server/adfs/artifact/{artifactId}	The artifact identifier that the AD FS server implementing the server role of the ADFSOAL Protocol must look up in its **Artifact Store** ADM element. The artifact identifier MUST be base64 URL encoded.

The responses to all the methods can result in the status codes defined in section 3.1.5.

The request messages for these methods do not use any custom HTTP headers.

The response messages for these methods do not use any custom HTTP headers.

The request body for messages to this service, unless otherwise noted, has the same encoding rules.

The response body for messages from this service, unless otherwise noted, has the same encoding rules.

3.2.5.1 http://server/adfs/artifact/{artifactId}

This URI corresponds to the artifact identifier that the AD FS server implementing the client role of the ADFSOAL Protocol needs to look up on the AD FS server implementing the server role of the ADFSOAL Protocol.

artifactId: A base64 URL encoded blob that contains the artifact identifier to be looked up.

The following HTTP methods are allowed to be performed on this resource.

HTTP method	Description
GET	Look up the artifact corresponding to the {artifactId} identifier.

3.2.5.1.1 GET

This method is transported by an HTTP **GET**.

The method can be invoked through the following URI:

```
http://server/adfs/artifact/{artifactId}?api-version={version}&client-request-
id={ClientRequestId}
```

The request message for this method supports the query parameters defined in section 2.2.3 (Common URI Parameters).

The request message for this method does not contain any custom HTTP headers.

The response message for this method does not contain any custom HTTP headers.

The response message for this method can result in the status codes defined in the status table in section 3.1.5.

3.2.5.1.1.1 Request Body

None.

3.2.5.1.1.2 Response Body

If an artifact corresponding to the artifact identifier that was specified in the request was found in the artifact store on the AD FS server implementing the server role of the ADFSOAL Protocol, the HTTP 200 status code is returned. Additionally, the response body for the **GET** response contains the JSON-formatted artifact object. The format of the JSON artifact object that is returned in the response body is defined in section 2.2.4.2 (Artifact) of this document.

If an artifact corresponding to the artifact identifier specified in the request was not found in the artifact store on the AD FS server implementing the server role of the ADFSOAL Protocol, the HTTP 404 status code is returned. The response body for the **GET** response is empty in this case.

3.2.5.1.1.3 Processing Details

When an AD FS server implementing the server role of the ADFSOAL Protocol receives an HTTP **GET** request to look up a specified artifact identifier, it implements the following processing logic:

- The AD FS server extracts the {artifactId} component from the request URL.
- The AD FS server then examines its **Artifact Store** ADM element to see if an artifact with the specified artifact identifier exists.
- **Success Response**: If the artifact was found in the **Artifact Store**, the AD FS server retrieves the corresponding access token.
  - The AD FS server responds to the HTTP **GET** request with an HTTP response with the HTTP status code set to 200 ("OK").
  - Additionally, the body of the HTTP response contains the JSON artifact.
- **Error Response (artifact not found)**: If the artifact was not found in the **Artifact Store**, the AD FS server responds to the HTTP **GET** request in the following manner:
  - The AD FS server responds with an HTTP response with the HTTP status code set to 404 ("Not Found").
  - The body of the HTTP response SHOULD contain an **ErrorDetails** object (section 2.2.5) that provides the client with additional information about the error.
- **Error Response**: If the AD FS server implementing the server role of the ADFSOAL Protocol encounters an error while processing the request, it returns one of the HTTP error status codes defined in section 3.1.5. In addition, the body of the HTTP response SHOULD contain an **ErrorDetails** object that provides the client with additional information about the error.

3.2.6 Timer Events

When the **ArtifactExpiryTimer** expires, the artifact is deleted from the **Artifact Store** ADM element because it is older than the artifact lifetime.

3.2.7 Other Local Events

None.

4 Protocol Examples

Note Throughout these examples, the fictitious names "client.example.com" and "server.example.com" are used as they are used in [RFC6749].

Note Throughout these examples, the HTTP samples contain extra line breaks to enhance readability.

4.1 Artifact Request

The following shows an example of a GET request from the "client role" of the ADFSOAL Protocol.

```
GET /adfs/artifact/yQNiQL5P0AgDAIaw0rL0FUcWQWs?api-version=1 HTTP/1.1
Host: server
```

4.2 Artifact Response

The following shows an example of a successful server response in the ADFSOAL Protocol.

```
HTTP/1.1 200 OK

{
    "clientId":"s6BhdRkqt3",

"data":"{\"access_token\":\"2YotnFZFEjr1zCsicMWpAA\",\"token_type\":\"bearer\",\"expires_in\"
:3600,\"refresh_token\":\"tGzv3JOkF0XG5Qx2TlKWIA\"}",
    "id":[0,86,136,145,194,79,208,8,4,0,27,34,218,191,131,30,238,186,221,5],
    "redirectUri":"https:\/\/client.example.com\/cb",
    "relyingPartyIdentifier":"https:\/\/resource_server"
 }
```

4.3 Artifact Error Response – Not Found

The following shows an example of a server response in the ADFSOAL Protocol when the requested artifact was not found.

```
HTTP/1.1 404 Not Found

{
    "debuginfo":null,
    "id":"",
    "message":"MSIS3106: SQL command returns no result when looking for artifact.",
"type":"Microsoft.IdentityServer.Service.ArtifactResolutionService.ArtifactNotFoundException"
}
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

The ADFSOAL Protocol assumes that security has already been negotiated by using [RFC4559] prior to the protocol starting.

6 Appendix A: Full JSON Schema

6.1 artifact Object

```
{
        "description" : "artifact object",
        "type" : "object",
        "properties" :
        {
            "id":
            {
                  "type":"array",
                  "optional":false,
                  "items" : { "type" : "integer", "minimum": 0, "maximum":255}
            },
            "clientId": {"type":"string", "optional":false},
            "redirectUri": {"type":"string", "optional":false},
            "relyingPartyIdentifier": {"type":"string", "optional":false},
            "data": {"type":"string", "optional":false}
        }
}
```

6.1.1 data Field

```
{
    "access_token": {"type":"string", "optional":false},
    "token_type": {"type":"string", "optional":false},
    "expires_in": {"type":"int", "optional":false},
    "refresh_token": {"type":"string", "optional":true},
}
```

6.2 ErrorDetails

```
{
    "description" : "error details",
    "type" : "object",
    "properties" :
    {
        "message": {"type":"string", "optional":true},
        "type": {"type":"string", "optional":true},
        "id": {"type":"string", "optional":true},
        "debugInfo": {"type":"string", "optional":true}
    }
}
```

7 (Updated Section) Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system
- Windows Server 2022 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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 2.2.3.1: The following table identifies the supported values for the *api-version* query parameter and the Windows products that support each value.

api-version value	Windows versions
"1"	Windows Server 2012 R2 Windows Server 2016 Windows Server operating system Windows Server 2019

8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

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

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

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

Section	Description	Revision class
7 Appendix B: Product Behavior	Updated for this version of Windows Server.	Major

9 Index

P

R

S

T

V