[MS-OAPX]:
OAuth 2.0 Protocol Extensions

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	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/15/2014	3.0	Major	Significantly changed the technical content.

Contents

1 Introduction

The OAuth 2.0 Protocol Extensions specify extensions to [RFC6749] (The OAuth 2.0 Authorization Framework).

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**
- **resource owner**

From [MS-ADFSWAP]:

- **relying party**

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)
globally unique identifier (GUID)
GUID
URI

The following terms are specific to this document:

AD FS server: See authorization server in [RFC6749].

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.

[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

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

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

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

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

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

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

1.3 Overview

Active Directory Federation Services (AD FS) implements parts of the OAuth 2.0 Authorization Framework, as defined in [RFC6749]. Additionally, AD FS implements a few extensions to the core protocol outlined in [RFC6749] that are referred to as the OAuth 2.0 Protocol Extensions and are specified in this document. These mandatory extensions need to be implemented by OAuth 2.0 clients that request authorization from **AD FS servers** using the OAuth 2.0 protocol.

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

1.4 Relationship to Other Protocols

The OAuth 2.0 Protocol Extensions (this document) specify extensions to the industry standard OAuth 2.0 Authorization Framework that is defined in [RFC6749]. These extensions are therefore dependent on the OAuth 2.0 protocol and use HTTPS [RFC2818] as the underlying transport protocol.

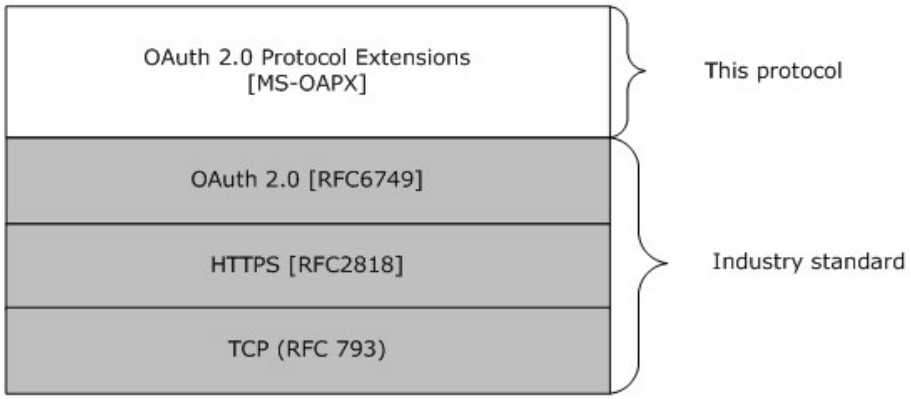

Figure 1: Protocol dependency

1.5 Prerequisites/Preconditions

- The OAuth 2.0 Protocol Extensions define extensions to [RFC6749]. AD FS supports only the Authorization Grant as defined in [RFC6749] section 1.3. A prerequisite to implementing the OAuth 2.0 Protocol Extensions is that the REQUIRED parts of [RFC6749] as they apply to the Authorization Grant have been implemented on the AD FS server.
- The OAuth 2.0 Protocol Extensions also assume that the OAuth 2.0 client knows the identifier for the resource, or relying party [MS-ADFSWAP], secured by the AD FS server for which the client is requesting authorization.

1.6 Applicability Statement

The OAuth 2.0 Protocol Extensions are supported by all AD FS servers that support the OAuth 2.0 protocol.<1> OAuth 2.0 clients that request authorization using the OAuth 2.0 protocol are required to implement the mandatory extensions defined in this protocol document.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

Supported Transports: The OAuth 2.0 Protocol Extensions only support HTTPS [RFC2818] as the transport protocol.

Protocol Versions: The OAuth 2.0 Protocol Extensions do not define protocol versions.

Localization: The OAuth 2.0 Protocol Extensions do not return localized strings.

Capability Negotiation: The OAuth 2.0 Protocol Extensions do not support capability negotiation.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The HTTPS [RFC2818] protocol MUST be used as the transport.

2.2 Common Data Types

2.2.1 HTTP Headers

The messages exchanged in the OAuth 2.0 Protocol Extensions 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, which is used when logging errors or failures that occur while processing the request.

2.2.1.1 client-request-id

The **client-request-id** header is optional and might be specified by the client role of the OAuth 2.0 Protocol Extensions. This header is used to provide the server role a unique request ID which is then used by the server of the OAuth 2.0 Protocol Extensions 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** header and the *ClientRequestId* query parameter defined in section 2.2.2.3 are mutually exclusive. The client is expected to specify a request identifier by using either one of these mechanisms.

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

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

2.2.2 Common URI Parameters

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

URI parameter	Description
resource	REQUIRED. This query parameter is used by the OAuth 2.0 client to specify the resource secured by the AD FS server for which it requires an authorization grant.
resource_params	OPTIONAL. This query parameter is used to specify a set of parameters corresponding to the resource secured by the AD FS server for which the OAuth 2.0 client requests authorization. The value is base64 URL encoded ([RFC4648] section 5). Padding is not required ([RFC4648] section 3.2).
ClientRequestId	OPTIONAL. This query parameter is used to specify a request ID that is used when logging errors or failures that occur while processing the request.

URI parameter	Description
login_hint OR username	OPTIONAL. This query parameter is used to provide a hint to the AD FS server about the login identifier the end-user may use to log in.

2.2.2.1 resource

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&ClientRequestId={ClientRequestId}&redirect_uri={redirect_uri} HTTP/1.1
```

REQUIRED

The *resource* query parameter is REQUIRED and MUST be specified by the client role of the OAuth 2.0 Protocol Extensions. When an OAuth 2.0 client requests an authorization code grant from an AD FS server (as specified in [RFC6749] section 4.1), it MUST use the *resource* query parameter to specify the resource secured by the AD FS server for which it requires an authorization grant. The value of the *resource* query parameter corresponds to the identifier with which the resource, or relying party [MS-ADFSWAP], is registered with the AD FS server by an administrator.

Note The *resource* query parameter is used in addition to the REQUIRED query parameters defined in [RFC6749] section 4.1.1.

For an example of the *resource* query parameter as it is being used, see section 4.1.

The format for the *resource* query parameter is as follows.

```
String = *(%x20-7E)
resource = String
```

2.2.2.2 resource_params

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&ClientRequestId={ClientRequestId}&resource_params={resource_params}&redirect_uri={redirec
t_uri} HTTP/1.1
```

OPTIONAL.

The *resource_params* query parameter is optional and MAY be specified by the client role of the OAuth 2.0 Protocol Extensions. When an OAuth 2.0 client requests an authorization code grant from an AD FS server (as specified in [RFC6749] section 4.1), it MAY use the *resource_params* query parameter to specify a set of parameters corresponding to the resource secured by the AD FS server. The resource for which these parameters are specified is identified by the value of the mandatory *resource* query parameter defined in the previous section.

The *resource_params* query parameter is a base64 URL encoded JSON-formatted string. Padding is not required. The *resource_params* query parameter MAY contain the optional **acr** element, which is used to specify a **URI** indicating the authentication method wanted. The **acr** element is conceptually similar to the optional *wauth* parameter defined in the Microsoft Web Browser Federated Sign-On Protocol ([MS-MWBF] section 2.2.3).

The following is a representation of the *resource_params* query parameter in base64 URL decoded JSON form:

```
resource_params= {
    "Properties":[{"Key":"acr","Value":"wiaormultiauthn"}]
}
```

The values supported for the **acr** element of the *resource_params* query parameter are:

Method of authentication wanted	acr URI
Windows integrated authentication for intranet access and multiple factor authentication for extranet access	wiaormultiauthn

For an example of the *resource_params* query parameter as it is being used, see section 4.1.

The format for the *resource_params* query parameter is as follows.

```
String = *(%x20-7E)
resource_params = String
```

2.2.2.3 ClientRequestId

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&ClientRequestId={ClientRequestId}&redirect_uri={redirect_uri} HTTP/1.1
```

OPTIONAL

The *ClientRequestId* query parameter is optional and MAY be specified by the client role of the OAuth 2.0 Protocol Extensions. This parameter is used to provide the server role a request identifier which is then used by the server of the OAuth 2.0 Protocol Extensions to log error messages that were encountered while processing that request. The value of the *ClientRequestId* query parameter MUST be a globally unique identifier (GUID) in standard string representation (see [C706] section 3.1.17 (String UUID) for the format).

For an example of the *ClientRequestId* query parameter as it is being used, see section 4.1.

The format for the *ClientRequestId* query parameter is as follows.

```
String = *(%x20-7E)
ClientRequestId = String
```

2.2.2.4 login_hint OR username

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&login_hint={login_hint}&redirect_uri={redirect_uri} HTTP/1.1
```

OPTIONAL.

When an OAuth 2.0 client requests an authorization code grant from an AD FS server (as specified in [RFC6749] section 4.1), it MAY use the *login_hint* query parameter. This query parameter provides a hint to the AD FS server about the login identifier the end-user may use to log in.

Note *login_hint* and *username* are aliases that signify the same query parameter and the OAuth 2.0 client may use either of these query parameters to provide a hint to the AD FS server about the login identifier the end-user may use to log in.

The following is an example of the *login_hint* query parameter as it is being used (which could be added to the example in section 4.1).

```
&login_hint=janedow
```

The format for the *login_hint* query parameter is as follows.

```
String = *(%x20-7E)
login_hint OR username = String
```

2.3 Error Codes

This document defines an extension to the list of error codes defined in [RFC6749].

2.3.1 invalid_resource

[RFC6749] section 4.1.2.1 (Error Response) defines the error response and error codes sent by the authorization server to the OAuth 2.0 client if the authorization request fails. In addition to the error codes defined in [RFC6749] section 4.1.2.1 (Error Response), the OAuth 2.0 Protocol Extensions define the invalid_resource error code that can be returned by the authorization server when processing an authorization request. If the OAuth 2.0 client specified an invalid resource in its authorization request using the *resource* query parameter defined in section 2.2.2.1, the authorization server returns the invalid_resource error code in the authorization response.

2.3.2 server_error

As defined in [RFC6749] section 4.1, after successfully retrieving an authorization grant, the OAuth 2.0 client subsequently requests an access token from the authorization server's token endpoint by including the authorization code received in the previous step. [RFC6749] section 5.2 (Error Response) defines the Error Response returned by the authorization server, if it encountered an error while processing the access token request.

In addition to the error codes defined in [RFC6749] section 5.2 (Error Response), the OAuth 2.0 Protocol Extensions define the server_error error code. Note that this error code is already defined in [RFC6749] section 4.1.2.1 as an error code returned by the authorization server when processing an authorization code grant request. The OAuth 2.0 Protocol Extensions define the same error code as a possible error code returned by the authorization server when processing an access token request. According to the requirement outlined in [RFC6749] section 6, this error code may also be returned by the authorization server if it encountered an error while processing a request to refresh an access token.

3 Protocol Details

3.1 OAuthExtension Client Details

The client role of the OAuth 2.0 Protocol Extensions corresponds to any OAuth 2.0 client that needs to request authorization to access a resource secured by an AD FS server using the Authorization Code Grant flow defined in the OAuth 2.0 protocol specified in [RFC6749].

The client role of this protocol uses the extensions defined in this document.

3.1.1 Abstract Data Model

The client role is expected to be aware of the relying party or resource identifier of the resource server for which it requires authorization. The client role sends this value to the AD FS server using the *resource* query string parameter.

3.1.2 Timers

None.

3.1.3 Initialization

The OAuth 2.0 Protocol Extensions do not define any special initialization requirements.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The resources accessed and manipulated by this protocol are the same as those defined in [RFC6749]. They are also listed below for reference:

Resource	Description
Authorization endpoint (/authorize)	For a description, see section 3.2.5.
Token endpoint (/token)	For a description, see section 3.2.5.

The HTTP responses to all the HTTP methods are defined in corresponding sections of [RFC6749].

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

3.1.5.1 Authorization endpoint (/authorize)

As defined in [RFC6749] section 3.1 (Authorization Endpoint), the authorization endpoint on the authorization server is used to interact with the resource owner and obtain an authorization grant. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
GET	For a description, see section 3.2.5.1.

3.1.5.1.1 GET

For the syntax and semantics of the GET method, see section 3.2.5.1.1.

3.1.5.1.1.1 Request Body

The format of the request is defined in [RFC6749] section 4.1.1 (Authorization Request).

3.1.5.1.1.2 Response Body

The format of the response body is defined in [RFC6749] section 4.1.2 (Authorization Response).

3.1.5.1.1.3 Processing Details

The steps performed by the OAuth 2.0 client to request an authorization code grant are defined in [RFC6749] section 4.1.1 (Authorization Request).

If the client chooses to send the optional *resource_params* query parameter, it MUST send it as a base64 URL encoded JSON-formatted string. The *resource_params* query parameter MAY include the optional **acr** element that specifies the URI of the authentication method wanted.

3.1.5.2 Token endpoint (/token)

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

HTTP method	Description
POST	For a description, see section 3.2.5.2.

3.1.5.2.1 POST

For the syntax and semantics of the POST method, see section 3.2.5.2.1 with the following addition:

- In the usage of the **client-request-id** header, if the client chooses to use the *ClientRequestId* query parameter, it SHOULD not set this HTTP header.

3.1.5.2.1.1 Request Body

The format of the request is defined in [RFC6749] sections 4.1.3 (Access Token Request) and 6 (Refreshing an Access Token).

3.1.5.2.1.2 Response Body

The format of the response body is defined in [RFC6749] sections 4.1.4 (Access Token Response) and 5 (Issuing an Access Token).

3.1.5.2.1.3 Processing Details

The steps performed by the OAuth 2.0 client to request an access token are defined in [RFC6749] section 4.1.3 (Access Token Request).

Additionally, the OAuth 2.0 client MUST expect the AD FS server to respond with an error response according to the requirements of [RFC6749] section 5.2 (Error Response) with the error parameter of the response set to the server_error error code (defined in section 2.3.2).

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 OAuthExtension Server Details

The server role of the OAuth 2.0 Protocol Extensions corresponds to the notion of an authorization server as defined in [RFC6749] section 1.1 (Roles).

The server role of this protocol implements support for the extensions defined in this document (the OAuth 2.0 Protocol Extensions).

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The OAuth 2.0 Protocol Extensions do not define any special initialization requirements.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The resources accessed and manipulated by this protocol are the same as those defined in [RFC6749]. They are also listed below for reference:

Resource	Description
Authorization endpoint (/authorize)	As defined in [RFC6749] section 3.1 (Authorization Endpoint), the authorization endpoint is used to interact with the resource owner and obtain an authorization grant.
Token endpoint (/token)	As defined in [RFC6749] section 3.2 (Token Endpoint), the token endpoint on the authorization server is used by an OAuth 2.0 client to obtain an access token by presenting its authorization grant or refresh token.

The HTTP responses to all the HTTP methods are defined in corresponding sections of [RFC6749].

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

3.2.5.1 Authorization endpoint (/authorize)

As defined in [RFC6749] section 3.1 (Authorization Endpoint), the authorization endpoint on the authorization server is used to interact with the resource owner and obtain an authorization grant. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
GET	An authorization request issued by the OAuth 2.0 client to the authorization endpoint of the AD FS server in accordance with the requirements of [RFC6749] section 4.1.1 (Authorization Request).

3.2.5.1.1 GET

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

The method can be invoked through the following URI:

```
/authorize?response_type={response_type}&client_id={client_id}&redirect_uri={redirect_uri}&sc
ope={scope}&state={state}&resource={resource}&resource_params={resource_p
arams}&ClientRequestId={ClientRequestId}&login_hint={login_hint}
```

The format of the authorization request is specified in [RFC6749] section 4.1.1 (Authorization Request). The OAuth 2.0 client MUST specify the query parameters marked as REQUIRED in [RFC6749] section 4.1.1.

In addition to the query parameters marked as REQUIRED in [RFC6749] section 4.1.1, the OAuth 2.0 client uses the following query parameters, which are defined in section 2.2.2 of this document.

resource: REQUIRED. The client MUST indicate the resource for which it requires authorization from the AD FS server using the *resource* parameter.

resource_params: OPTIONAL. The client may choose to specify this optional query parameter to specify a set of parameters corresponding to the resource secured by the AD FS server for which it requires authorization.

ClientRequestId: OPTIONAL. The client may choose to specify this optional query parameter to specify a request ID which is used when logging errors or failures that occur while processing the request.

login_hint: OPTIONAL. The client may choose to specify this optional query parameter to provide a hint to the AD FS server about the login identifier the end-user may use to log in.

The request message for this method may contain the following optional HTTP headers. The header syntax is defined in section 2.2.1.

Request header	Usage	Value
client-request-id	This optional header is used to specify a request identifier which is used when logging errors or failures that occur while processing the request. If the client chooses to use the *ClientRequestId* query parameter, it SHOULD not set this HTTP header.	A request identifier, which MUST be a **GUID**.

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 [RFC6749] section 4.1.2.

3.2.5.1.1.1 Request Body

The format of the request is defined in [RFC6749] section 4.1.1 (Authorization Request).

3.2.5.1.1.2 Response Body

The format of the response body is defined in [RFC6749] section 4.1.2 (Authorization Response).

3.2.5.1.1.3 Processing Details

The steps performed by the AD FS server to respond to an authorization code request are defined in [RFC6749] section 4.1.2 (Authorization Response).

The following additional processing steps are expected as a result of the extensions included in this document:

- The AD FS server MUST validate that the *resource* query parameter was specified by the OAuth 2.0 client.
- The AD FS server MUST validate that the *resource* query parameter specified by the OAuth 2.0 client matches a resource or relying party registered with the AD FS server.
- If the *resource* query parameter is invalid or not found to be registered on the AD FS server, the AD FS server must respond to the OAuth 2.0 0 client as per the requirements of [RFC6749] section 4.1.2.1 (Error Response). The REQUIRED error parameter of the response MUST be set to the invalid_resource error code as defined in section 2.3.1.
- If the OAuth 2.0 client specified the *resource_params* query parameter the AD FS server MUST base64 URL decode the value of this query parameter, treating padding characters as optional, and convert it to a JSON object for further processing (that is, parse the string value of the query parameter and convert it to a JSON object).
    - If the OAuth 2.0 client specified an authentication method URI as part of the **acr** element of the *resource_params* query parameter and if the authentication method is valid, the AD FS server MUST use that authentication method when authenticating the user.
    - If the authentication method specified as part of the **acr** element is invalid or not supported by the AD FS server, the AD FS server MUST respond to the OAuth 2.0 client according to the requirements of [RFC6749] section 4.1.2.1. The REQUIRED error parameter of the response MUST be set to invalid_request error code as defined in [RFC6749] section 4.1.2.1. This error code is also returned if the value of the *resource_params* query parameter is invalid (that is, if it cannot be base64 URL decoded or is an invalid JSON-formatted string).
- If the OAuth 2.0 client specified the *login_hint* query parameter, the AD FS server SHOULD use the value of the *login_hint* query parameter as a hint about the login identifier the end-user may use to log in.
- If the OAuth 2.0 client specified either the *ClientRequestId* query parameter or the **client-request-id** HTTP header in the access token request, the AD FS server MUST use the request identifier specified in the request when logging errors or failures that occur while processing that authorization request.

- If the OAuth 2.0 client specifies both the *ClientRequestId* query parameter as well as the **client-request-id** HTTP header, the AD FS server MUST use the value specified in the query parameter, when logging errors or failures that occur while processing that authorization request and ignore the value specified in the HTTP header.

3.2.5.2 Token endpoint (/token)

As defined in [RFC6749] section 3.2 (Token Endpoint), the token endpoint on the AD FS server is used by an OAuth 2.0 client to obtain an access token by presenting its authorization grant or refresh token. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
POST	An access token request issued by the OAuth 2.0 client to the token endpoint of the AD FS server in accordance with the requirements of [RFC6749] section 4.1.3 (Access Token Request).

3.2.5.2.1 POST

This operation is transported by an HTTP **POST**

The operation can be invoked through the following URI:

```
/token?ClientRequestId={ClientRequestId}
```

The format of the access token request is specified in [RFC6749] section 4.1.3 (Access Token Request). The OAuth 2.0 client MUST specify the query parameters marked as REQUIRED in [RFC6749] section 4.1.3.

In addition to the query parameters marked as REQUIRED in [RFC6749] section 4.1.3, the OAuth 2.0 client can choose to send the *ClientRequestId* query parameter.

ClientRequestId: OPTIONAL. The client may choose to specify this optional query parameter to specify a request ID which is used when logging errors or failures that occur while processing the request.

The request message for this method may contain the following optional HTTP headers. The header syntax is defined in section 2.2.1.

Request header	Usage	Value
client-request-id	This optional header is used to specify a request identifier which is used when logging errors or failures that occur while processing the request.	A request identifier, which MUST be a GUID.

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 [RFC6749] sections 5.1 (Successful Response) and 5.2 (Error Response). Additionally, if the AD FS server encountered an error while processing the client's access token request, it may return the server_error error code defined in this document.

3.2.5.2.1.1 Request Body

The format of the request is defined in [RFC6749] sections 4.1.3 (Access Token Request) and 6 (Refreshing an Access Token).

3.2.5.2.1.2 Response Body

The format of the response body is defined in [RFC6749] sections 4.1.4 (Access Token Response) and 5 (Issuing an Access Token).

3.2.5.2.1.3 Processing Details

The steps performed by the AD FS server to process an OAuth 2.0 client's access token request are defined in [RFC6749] sections 4.1.3 (Access Token Response) and 5 (Issuing an Access Token).

The following additional processing steps are expected as a result of the extensions included in this document:

- If the OAuth 2.0 client specified either the *ClientRequestId* query parameter or the **client-request-id** HTTP header in the access token request, the AD FS server MUST use the request identifier specified in the request when logging errors or failures that occur while processing that access token request.
- If the OAuth 2.0 client specifies both the *ClientRequestId* query parameter as well as the **client-request-id** HTTP header, the AD FS server MUST use the value specified in the query parameter, when logging errors or failures that occur while processing that authorization request and ignore the value specified in the HTTP header.
- If the AD FS server encountered an internal error when processing the OAuth 2.0 client's access token request, it MUST respond to the OAuth 2.0 client according to the requirements of [RFC6749] section 5.2 (Error Response). The REQUIRED error parameter of the response MUST be set to server_error.

3.2.6 Timer Events

None.

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 Authorization Code Request

Refer to [RFC6749] section 4.1.1 (Authorization Request).

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
  &resource= https:%2F%2Fresource_server
  &ClientRequestId=EC09AB2D-9655-453B-B555-3317011523E8

&resource_params=eyJQcm9wZXJ0aWVzIjpbeyJLZXkiOiJhY3IiLCJWYWx1ZSI6IndpYW9ybXVsdGlhdXRobiJ9XX0
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

4.2 Authorization Code Response

Refer to [RFC6749] section 4.1.2 (Authorization Response).

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
&state=xyz
```

4.3 Access Token Request

Refer to [RFC6749] section 4.1.3 (Access Token Request).

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&client_id=s6BhdRkqt3&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

4.4 Access Token Response

Refer to [RFC6749] section 5.1 (Successful Response).

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

4.5 Access Token Error Response – server_error

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "error":"server_error"
}
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full JSON Schema

```
resource_params= {

"Properties":[{"Key":"acr","Value":"wiaormultiauthn"}]

}
```

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 Server 2012 R2 operating system

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

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

<1> Section 1.6: Support for the OAuth 2.0 protocol in AD FS is available starting with the Windows Server 2012 R2.

8 Change Tracking

This section identifies changes that were made to the [MS-OAPX] protocol document between the February 2014 and May 2014 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

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

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

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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

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

- Obsolete document removed.

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.2 Common URI Parameters	70943 Revised description of the URI parameter "resource_params".	Y	Content updated.
2.2.2.2 resource_params	70943 Changed "base64 encoded" to "base64 URL encoded".Added "padding is not required".	Y	Content updated.
3.1.5.1.1.3 Processing Details	70943 Changed "base64 encoded" to "base64 URL encoded".	Y	Content updated.
3.2.5.1.1.3 Processing Details	70943 Changed "base64" to "base64 URL".Added "treating padding characters as optional".	Y	Content updated.

9 Index