[MS-OAPX-Diff]:

OAuth 2.0 Protocol Extensions

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	2.0	Major	Significantly changed the technical content.
2/13/2014	2.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	3.0	Major	Significantly changed the technical content.
6/30/2015	4.0	Major	Significantly changed the technical content.
7/14/2016	5.0	Major	Significantly changed the technical content.
6/1/2017	6.0	Major	Significantly changed the technical content.
6/13/2017	7.0	Major	Significantly changed the technical content.
9/15/2017	8.0	Major	Significantly changed the technical content.
12/1/2017	8.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2018	9.0	Major	Significantly changed the technical content.
4/7/2021	10.0	Major	Significantly changed the technical content.

Table of Contents

1 (Updated Section) Introduction

The OAuth 2.0 Protocol Extensions specifyare extensions to [RFC6749] (The the OAuth 2.0 Authorization Framework). [RFC6749]. When no operating system version information isan AD FS behavior level is not specified, informationthe details in this document applies to all relevant versions of Windows. Similarly, when no AD FS behavior level is specified, information in this document appliesapply to all AD FS behavior levels.

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**
- **confidential client**
- **redirection URI**
- **refresh token**
- **resource owner**

From [OIDCCore]:

- **ID token**

From [IETFDRAFT-DEVICEFLOW-11]:

- **Device Verification Code**
- **User code**
- **Verification URI**

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 (Updated Section) 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.

AD FS behavior level: A specification of the functionality available in an AD FS server. Possible values such as AD_FS_BEHAVIOR_LEVEL_1 and AD_FS_BEHAVIOR_LEVEL_2 are described in [MS-OAPX].

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

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

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).

key: In cryptography, a generic term used to refer to cryptographic data that is used to initialize a cryptographic algorithm. Keys are also sometimes referred to as keying material.

multi-resource refresh token: A refresh token (see [RFC6749] section 1.5) that can be redeemed for an access token for any resource. If a refresh token is not a multi-resource refresh token, then it can only be redeemed for an access token for the same resource that was originally requested when the refresh token was granted.

OAuth device flow request: An OAuth request in which the client is an input-constrained device and follows the OAuth 2.0 Device Flow protocol [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] for authorization.

OAuth logon certificate request: An OAuth request in which a resource, or relying party, acts as a client and uses a previously received access token to request an X.509 certificate. The resulting certificate represents the same identity represented by the access token.

OAuth on-behalf-of request: An OAuth request in which a resource, or relying party, acts as a client and uses a previously received access token to request an access token for another resource.

public key: One of a pair of keys used in public-key cryptography. The public key is distributed freely and published as part of a digital certificate. For an introduction to this concept, see [CRYPTO] section 1.8 and [IEEE1363] section 3.1.

relying party (RP): A web application or service that consumes security tokens issued by a security token service (STS).

smart card: A portable device that is shaped like a business card and is embedded with a memory chip and either a microprocessor or some non-programmable logic. Smart cards are often used as authentication tokens and for secure key storage. Smart cards used for secure key storage have the ability to perform cryptographic operations with the stored key without allowing the key itself to be read or otherwise extracted from the card.

trusted platform module (TPM): A component of a trusted computing platform. The TPM stores keys, passwords, and digital certificates. See [TCG-Architect] for more information.

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].

Windows client authentication: An OAuth 2.0 client authentication mechanism (see [RFC6749] section 2.3) in which the client authenticates via the SPNEGO-based Kerberos and NTLM HTTP Authentication mechanism described in [RFC4599].

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.

[IETFDRAFT-DEVICEFLOW-11] Denniss, W., Bradley, J., Jones, M., and Tschofenig, H., "OAuth 2.0 Device Flow for Browserless and Input Constrained Devices", draft-ietf-oauth-device-flow-11, July 2018, https://tools.ietf.org/html/draft-ietf-oauth-device-flow-11

~~[IETFDRAFT-JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key-41, January 2015, https://tools.ietf.org/html/draft-ietf-jose-json-web-key-41~~

~~[IETFDRAFT-JWT] Internet Engineering Task Force (IETF), "JSON Web Token JWT", draft-ietf-oauth-json-web-token, April 2013, http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-08~~

[IETFDRAFT-OAUTH2TOKBIND~~] Popov, A., Ed., Nystroem, M., Balfranz, D., et al-08] Jones, M., Campbell, B., Bradley, J., and Denniss, W~~., "OAuth 2.0 Token Binding", draft-ietf-oauth-token-binding-~~04, July 2017, https://tools.ietf.org/html/draft-ietf-oauth-token-binding-04~~

~~[IETFDRAFT-TOKBINDPROT] Popov, A., Ed., Nystroem, M., Balfanz, D., et al., "The Token Binding Protocol Version 1.0", draft-ietf-tokbind-protocol-19, May~~08, October 2018, https://tools.ietf.org/html/draft-ietf-~~tokbind-protocol-19~~oauth-token-binding-08

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

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

[MS-KPP] Microsoft Corporation, "Key Provisioning Protocol".

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

[MS-OAPXBC] Microsoft Corporation, "OAuth 2.0 Protocol Extensions for Broker Clients".

[MS-OIDCE] Microsoft Corporation, "OpenID Connect 1.0 Protocol Extensions".

[MS-WCCE] Microsoft Corporation, "Windows Client Certificate Enrollment Protocol".

[MSFT-WKPLJOIN] Microsoft Corporation, "Microsoft Workplace Join for non-Windows 10 computers", https://www.microsoft.com/en-us/download/details.aspx?id=53554

[MSKB-3172614] Microsoft Corporation, "July 2016 update rollup for Windows RT 8.1, Windows 8.1, and Windows Server 2012 R2", https://support.microsoft.com/en-us/kb/3172614

[MSKB-4022723] Microsoft Corporation, "June 27, 2017 - KB4022723 (OS Build 14393.1378)", https://support.microsoft.com/en-us/kb/4022723

[MSKB-4034658] Microsoft Corporation, "August 8, 2017—KB4034658 (OS Build 14393.1593)", https://support.microsoft.com/help/4034658

[MSKB-4088889] Microsoft Corporation, "March 22, 2018 - KB4088889 (OS Build 14393.2155)", https://support.microsoft.com/en-us/help/4088889

[MSKB-4457127] Microsoft Corporation, "September ~~18~~20, 2018—KB4457127", (OS Build 14393.2515)", https://support.microsoft.com/en-us/help/4457127/windows-10-update-kb4457127

[OIDCCore] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and Mortimore, C., "OpenID Connect Core 1.0 incorporating errata set 1", November 2014, http://openid.net/specs/openid-connect-core-1_0.html

[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.rfc-editor.org/rfc/rfc2818.txt

[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

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt

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

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, May 2015, http://www.rfc-editor.org/rfc/rfc7517.txt

[RFC7519] Internet Engineering Task Force, "JSON Web Token (JWT)", http://www.rfc-editor.org/rfc/rfc7519.txt

[RFC8471] Popov, A., Ed., Nystroem, M., Balfanz, D., and Hodges, J., "The Token Binding Protocol Version 1.0", RFC 8471, October 2018, https://www.rfc-editor.org/rfc/rfc8471.txt

[RFC8628] Denniss, W., Bradley, J., Jones, M., and Tschofenig, H., "OAuth 2.0 Device Authorization Grant", RFC 8628, August 2019, https://www.rfc-editor.org/rfc/rfc8628.txt

1.2.2 (Updated Section) Informative References

[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.

1.3 (Updated Section) 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~~] and extends it. Those extensions ~~to the core protocol outlined in [RFC6749] that are referred to as~~are the OAuth 2.0 Protocol Extensions and are specified in this document. ~~These mandatory extensions need to be~~They are 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~~, which are ~~used~~examples in [RFC6749].

In addition to the terms specified in the Glossary (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**
- **confidential client**
- **redirection URI**
- **refresh token**
- **resource owner**

From [OIDCCore]:

- **ID token**

From [RFC8628]:

- **Device Verification Code**
- **User code**
- **Verification URI**

1.4 (Updated Section) 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~~are dependent on the OAuth 2.0 protocol and use HTTPS [RFC2818] as the underlying transport protocol. This is illustrated by the following diagram.

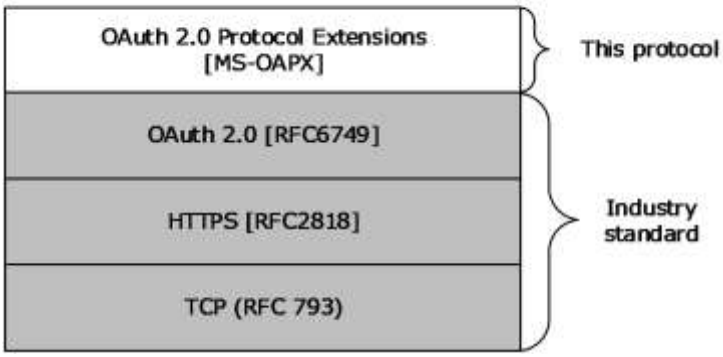

Figure 1: Protocol dependency

1.5 (Updated Section) 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]~~AD FS as they apply to the Authorization Grant ([RFC6749] section 1.3) have been implemented on the AD FS server.

The OAuth 2.0 Protocol Extensions ~~also~~ assume that if the OAuth 2.0 client requests authorization for a particular resource, or relying party, secured by the AD FS server, the client knows the identifier of that resource. These extensions also assume that the OAuth 2.0 client knows its own client identifier and all relevant client authentication information if it is a confidential client.

The OAuth 2.0 Protocol Extensions ~~(this document),~~assume that they, the OAuth 2.0 Protocol Extensions for Broker Clients [MS-OAPXBC], and the OpenID Connect 1.0 Protocol Extensions [MS-OIDCE], if being used, ~~MUST~~are all be running on the same AD FS server.

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.<2>

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 *client-request-id* 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	OPTIONAL. 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. This parameter is REQUIRED when the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_1, and OPTIONAL when the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
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).

URI parameter	Description
client-request-id	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.
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 might use to log in.
domain_hint	OPTIONAL. This query parameter is used to provide a hint to the AD FS server about the backend authentication service the end user can log in to. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
nonce	OPTIONAL. This query parameter is used in the same way as the nonce parameter defined in [OIDCCore] section 3.1.2.1. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
prompt	OPTIONAL. This query parameter is used in the same way as the prompt parameter defined in [OIDCCore] section 3.1.2.1, but the only accepted values for this parameter are "none" and "login". This parameter and the accepted values specified in the preceding paragraph SHOULD<3> be supported for all values of **ad_fs_behavior_level**.
max_age	OPTIONAL. This query parameter is used in the same way as the max_age parameter defined in [OIDCCore] section 3.1.2.1. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
id_token_hint	OPTIONAL. This query parameter is used in the same way as the id_token_hint parameter defined in [OIDCCore] section 3.1.2.1. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
amr_values	OPTIONAL. This query parameter is used by the client to request that a particular authentication method be used to authenticate the user. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.<4>
mfa_max_age	OPTIONAL. This query parameter is used by the client to specify the allowable timespan, in seconds, within which the last multiple factor authentication must have been performed by the end user. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_3 or higher.<5>

2.2.2.1 resource

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

OPTIONAL

The *resource* 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 authorization from an AD FS server (as specified in [RFC6749] sections 4.1 and 4.2), it MAY 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, is registered with the AD FS server by an administrator.

This parameter is REQUIRED when the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_1, and OPTIONAL when the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher, and if the *resource* query parameter is not specified, the server issues an access token to the client that can be used to access the UserInfo endpoint ([OIDCCore] section 5.3), if such endpoint exists. The server supports the use of the returned access token at the UserInfo endpoint regardless of whether the client role also requests the "openid" scope.

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}&client-request-
id={ClientRequestId}&resource_params={resource_params}&redirect_uri={redirect_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 authorization from an AD FS server (as specified in [RFC6749] sections 4.1 and 4.2), 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 optional *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 client-request-id

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

OPTIONAL

The *client-request-id* 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 *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).

For an example of the *client-request-id* query parameter as it is being used, see section 4.1.

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

```
String = *(%x20-7E)
client-request-id = 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 authorization from an AD FS server (as specified in [RFC6749] sections 4.1 and 4.2), 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 might use to log in.

Note: *login_hint* and *username* are aliases that signify the same query parameter. The OAuth 2.0 client can use either of these query parameters to provide a hint to the AD FS server about the login identifier the end user might 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@contoso.com
```

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

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

```
login_hint OR username = String
```

2.2.2.5 domain_hint

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

OPTIONAL

When an OAuth 2.0 client requests authorization from an AD FS server (as specified in [RFC6749] sections 4.1 and 4.2), it MAY use the *domain_hint* query parameter. This query parameter provides a hint to the AD FS server about the backend authentication service the end user can log in to.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

The following is an example of the *domain_hint* query parameter as it is being used.

```
&domain_hint=contoso.com
```

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

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

2.2.2.6 nonce

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&client-request-id={ClientRequestId}&redirect_uri={redirect_uri}&nonce={nonce} HTTP/1.1
```

OPTIONAL

The *nonce* query parameter is OPTIONAL, and can be specified by the client role of the OAuth 2.0 Protocol Extensions. This parameter has the same behavior as the nonce parameter defined in [OIDCCore] section 3.1.2.1, but can be specified regardless of whether the client role also requests the "openid" scope.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *nonce* query parameter being used, see section 4.9.

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

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

2.2.2.7 prompt

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&client-request-id={ClientRequestId}&redirect_uri={redirect_uri}&prompt={prompt} HTTP/1.1
```

OPTIONAL

The *prompt* query parameter is OPTIONAL, and can be specified by the client role of the OAuth 2.0 Protocol Extensions. This parameter has the same behavior as the prompt parameter defined in [OIDCCore] section 3.1.2.1 (see section 2.2.2 for exceptions and support information), but can be specified regardless of whether the client role also requests the "openid" scope.

For an example of the *prompt* query parameter being used, see section 4.10.

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

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

2.2.2.8 max_age

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&client-request-id={ClientRequestId}&redirect_uri={redirect_uri}&max_age={max_age}
HTTP/1.1
```

OPTIONAL

The *max_age* query parameter is OPTIONAL, and can be specified by the client role of the OAuth 2.0 Protocol Extensions. This parameter has the same behavior as the max_age parameter defined in [OIDCCore] section 3.1.2.1, but can be specified regardless of whether the client role also requests the "openid" scope.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *max_age* query parameter being used, see section 4.11.

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

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

2.2.2.9 id_token_hint

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&client-request-
id={ClientRequestId}&redirect_uri={redirect_uri}&id_token_hint={id_token_hint} HTTP/1.1
```

OPTIONAL

The *id_token_hint* query parameter is OPTIONAL, and can be specified by the client role of the OAuth 2.0 Protocol Extensions. This parameter has the same behavior as the id_token_hint parameter

defined in [OIDCCore] section 3.1.2.1, but can be specified regardless of whether the client role also requests the "openid" scope.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *id_token_hint* query parameter being used, see section 4.12.

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

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

2.2.2.10 (Updated Section) amr_values

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&client-request-id={ClientRequestId}&amr_values={amr_values}&redirect_uri={redirect_uri}
HTTP/1.1
```

OPTIONAL

The *amr_values* query parameter is ~~optional~~OPTIONAL and can be specified by the client role of the OAuth 2.0 Protocol Extensions. When an OAuth 2.0 client requests authorization from an AD FS server (as specified in [RFC6749] sections 4.1 and 4.2), it can use the *amr_values* to request that the user be authenticated using a particular authentication method. The *amr_values* query parameter is conceptually similar to the optional *wauth* parameter defined in [MS-MWBF] section 2.2.3.

The following values are supported for the *amr_values* query parameter:

Value	Method of authentication requested
ngcmfa	Multiple factor authentication is required. ~~If the user authenticates~~ User authentication with a certificate or other asymmetric key-based mechanism~~, and the~~ using a key that is ~~used is~~ present in the **msDS-KeyCredentialLink** attribute on the **user** object in Active Directory~~, then the server SHOULD NOT consider that authentication alone as satisfying~~ does not satisfy multiple factors, even if the key ~~that is used~~ is protected by a smart card or requires a personal identification number (PIN) to unlock.

~~Servers might support additional values.~~
The server ignores this parameter if the *resource_params* parameter is given.

The format for the *amr_values* query parameter is as follows:

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

2.2.2.11 mfa_max_age

```
GET
/authorize?response_type={response_type}&client_id={client_id}&state={state}&resource={resour
ce}&client-request-id={ClientRequestId}&redirect_uri={redirect_uri}&mfa_max_age={mfa_max_age}
HTTP/1.1
```

OPTIONAL

The *mfa_max_age* query parameter is OPTIONAL and can be specified by the client role of the OAuth 2.0 Protocol Extensions. The value specifies the allowable elapsed time, in seconds, since the last time the end user was actively multiple-factor authenticated by AD FS.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_3 or higher. See section 2.2.2 for additional support information.

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

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

2.2.3 (Updated Section) Common Data Structures

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

Message body parameter	Description
requested_token_use	OPTIONAL. The OAuth 2.0 client can include this parameter in the POST body of a request to indicate what type of processing it is requesting when providing a *grant_type* parameter of "urn:ietf:params:oauth:grant-type:jwt-bearer". The OAuth 2.0 client sets this parameter to a value of "on_behalf_of" when making an OAuth on-behalf-of request. An OAuth on-behalf-of request is an OAuth request in which a resource, or relying party, acts as a client and uses a previously received access token to request an access token for another resource. See section 3.1.5.2.1.1 for request details, section 3.2.5.2.1.3 for server processing details, and section 4.7 for an example. The OAuth 2.0 client sets this parameter to a value of "logon_cert" when making an OAuth logon certificate request. An OAuth logon certificate request is an OAuth request in which a resource, or relying party, acts as a client and uses a previously received access token to request an X.509 certificate ([RFC5280]), which can be used to log the user represented in the access token onto another network resource without prompting the user for credentials. See section 3.1.5.2.1.1 for request details, section 3.2.5.2.1.3 for server processing details, and section 4.13 for an example. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
assertion	OPTIONAL. The OAuth 2.0 client includes this parameter in the POST body of a request and sets it to the value of an access token previously issued by the AD FS server when making an OAuth on-behalf-of request or an OAuth logon certificate request. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
resource (request parameter)	OPTIONAL. The OAuth 2.0 client includes this parameter in the POST body of a request to specify the resource secured by the AD FS server for which it requires an access token. It can be provided when refreshing an access token (see [RFC6749] section 6) or when making an OAuth on-behalf-of request. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
resource (response parameter)	OPTIONAL. The AD FS server includes this parameter in the response and sets it to the identifier of the current resource when providing a multi-resource refresh token. A multi-resource refresh token is one that can be redeemed for an access token for any resource registered with the AD FS

Message body parameter	Description
	server. The AD FS server does not return this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
use_windows_client_authentication	OPTIONAL. An OAuth 2.0 confidential client includes this parameter in the POST body of a request to indicate that it will use Windows client authentication and authenticate via the mechanism described in [RFC4559]. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
csr_type	OPTIONAL. The OAuth 2.0 client includes this parameter in the POST body of a request when making an OAuth logon certificate request to indicate the format of the request provided in the *csr* parameter (see [MS-WCCE] section 2.2.2.6). The only value supported for this parameter is "http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10". The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
csr	OPTIONAL. The OAuth 2.0 client includes this parameter in the POST body of a request when making an OAuth logon certificate request and sets the value to a base64-encoded PKCS#10 certificate request (see [MS-WCCE] section 3.1.1.4.3.1.1). The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
x5c	OPTIONAL. The AD FS server includes this parameter in the successful response to an OAuth logon certificate request. The value is a base64-encoded CMS certificate chain or CMC full PKI response (see [MS-WCCE] section 2.2.2.8). The AD FS server does not return this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
tbidv2	OPTIONAL. The OAuth 2.0 client includes this parameter in the POST body of a request to indicate that the client is providing a referred token-binding ID to the AD FS server for the current request. See [~~IETFDRAFT-TOKBINDPROT~~RFC8471] for details on referred token-bindings. The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.<6>

2.2.3.1 requested_token_use

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&redirect_uri={redirect_uri}&requested_token_use
={requested_token_use}&assertion={assertion}&resource={resource}
```

OPTIONAL

The *requested_token_use* parameter is optional, and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when making a request to the token endpoint (section 3.2.5.2). The client provides a value of "on_behalf_of" to indicate that the request should be processed as an OAuth on-behalf-of request and a value of "logon_cert" to indicate that the request should be processed as an OAuth logon certificate request.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *requested_token_use* parameter being used, see section 4.7.

The format for the *requested_token_use* parameter is as follows.

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

2.2.3.2 assertion

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&redirect_uri={redirect_uri}&requested_token_use
={requested_token_use}&assertion={assertion}&resource={resource}
```

OPTIONAL

The *assertion* parameter is optional, and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when making a request to the token endpoint (section 3.2.5.2). The client provides an access token previously received from the AD FS server in the *assertion* parameter when making an OAuth on-behalf-of request or an OAuth logon certificate request.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *assertion* parameter being used, see section 4.7.

The format for the *assertion* parameter is as follows.

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

2.2.3.3 resource

The resource parameter can be included in either a request to the AD FS server, or in a response from the AD FS server. The following sections describe each use.

2.2.3.3.1 resource request parameter

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&redirect_uri={redirect_uri}&requested_token_use
={requested_token_use}&assertion={assertion}&resource={resource}
```

OPTIONAL

The *resource* parameter is optional, and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when making a request to the token endpoint (section 3.2.5.2).

When an OAuth 2.0 client makes an OAuth on-behalf-of request to the token endpoint (section 3.2.5.2), it provides the *resource* parameter to specify the resource secured by the AD FS server for which it requires an access token.

An OAuth 2.0 client can also provide the *resource* parameter when using a multi-resource refresh token to request an access token for a different resource than the one that was used when the refresh token was returned (see [RFC6749] section 6). The *resource* parameter can only be used with a refresh token if it is a multi-resource refresh token.

The value of the *resource* parameter corresponds to the identifier with which the resource, or relying party, is registered with the AD FS server by an administrator.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *resource* request parameter being used, see section 4.7.

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

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

2.2.3.3.2 resource response parameter

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{"access_token":{access_token},"token_type":{token_type},"expires_in":{expires_in},"resource"
:{resource},"refresh_token":{refresh_token}}
```

OPTIONAL

The *resource* response parameter is optional, and can be specified by the server role of the OAuth 2.0 Protocol Extensions when returning a refresh token. The AD FS server returns the same value of the *resource* parameter specified by the client in the request, or "urn:microsoft:userinfo" if the *resource* parameter is not specified by the client in the request, to indicate to the client that the refresh token in the response is a multi-resource refresh token.

The AD FS server does not return this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

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

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

2.2.3.4 use_windows_client_authentication

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&code={code}&redirect_uri={redirect_uri}&use_win
dows_client_authentication={use_windows_client_authentication}
```

OPTIONAL

The *use_windows_client_authentication* parameter is optional, and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when making a request to the token endpoint (section 3.2.5.2). The client provides a value of "true" for the *use_windows_client_authentication* parameter to indicate that it will authenticate via the HTTP Negotiate Authentication Scheme described in [RFC4559].

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *use_windows_client_authentication* parameter being used, see section 4.8.

The format for the *use_windows_client_authentication* parameter is as follows.

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

2.2.3.5 csr

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&redirect_uri={redirect_uri}&requested_token_use
={requested_token_use}&assertion={assertion}&csr={csr}&csr_type={csr_type}
```

OPTIONAL

The *csr* parameter is optional, and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when making a request to the token endpoint (section 3.1.5.2). The client provides a base64-encoded PKCS#10 certificate request ([MS-WCCE] section 3.1.1.4.3.1.1) in the *csr* parameter when making an OAuth logon certificate request.

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *csr* parameter being used, see section 4.13.

The format for the *csr* parameter is as follows.

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

2.2.3.6 csr_type

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&redirect_uri={redirect_uri}&requested_token_use
={requested_token_use}&assertion={assertion}&csr={csr}&csr_type={csr_type}
```

OPTIONAL

The *csr_type* parameter is optional, and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when making a request to the token endpoint (section 3.1.5.2). The client includes this parameter when providing a *csr* parameter to indicate the format of the *csr* parameter. The only supported value for the *csr_type* parameter is "http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10".

The AD FS server ignores this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *csr_type* parameter being used, see section 4.13.

The format for the *csr_type* parameter is as follows.

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

2.2.3.7 x5c

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{"x5c"={x5c},"token_type":{token_type},"expires_in":{expires_in},"resource":{resource},"refre
sh_token":{refresh_token}}
```

OPTIONAL

The *x5c* response parameter is optional, and is returned by the AD FS server in response to a successful OAuth logon certificate request. The value returned is a base64-encoded CMS certificate chain or a CMC full PKI response (see [MS-WCCE] section 2.2.2.8).

The AD FS server does not return this parameter unless its **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.

For an example of the *x5c* response parameter being used, section 4.13.

The format for the *x5c* response parameter is as follows.

```
String = *(%x20-7E)
x5c = String
```

2.2.3.8 (Updated Section) tbidv2

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type={grant_type}&client_id={client_id}&redirect_uri={redirect_uri}&tbidv2={tbidv2}
```

OPTIONAL

The *tbidv2* parameter is optional and can be specified by the client role of the OAuth 2.0 Protocol Extensions in the POST body when the client is providing a referred token-binding ID as part of the request. For details on referred token-binding IDs, see [~~IETFDRAFT-TOKBINDPROT~~RFC8471].

The format for the *tbidv2* request parameter is as follows:

```
String = *(%x20-7E)
tbidv2 = String
```

2.2.4 (Updated Section) Error Codes

This document defines an extension to the list of error codes defined in [RFC6749] and [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628].<7>

2.2.4.1 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.

2.2.4.2 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 can also be returned by the authorization server if it encountered an error while processing a request to refresh an access token.

2.3 Directory Service Schema Elements

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

For the syntax of <Class> or <Class><Attribute> pairs, refer to the following:

- Active Directory Domain Services (AD DS): [MS-ADA2], [MS-ADSC]

Class	Attribute
user	msDS-KeyCredentialLink

3 Protocol Details

3.1 (Updated Section) 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] and the OAuth 2.0 authorization flow for browserless and input-constrained devices defined in the OAuth 2.0 Device Flow protocol specified in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628].<8>

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 if it requests authorization for a particular resource. The client role sends this value to the AD FS server using the *resource* query string parameter.

The client role is also expected to be aware of its own client identifier and all relevant client authentication information if it is a confidential client.

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] and [IETFDRAFT-DEVICEFLOW-11]. 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.
Device authorization endpoint (/devicecode)	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 *client-request-id* query parameter, it SHOULD NOT set this HTTP header.

3.1.5.2.1.1 (Updated Section) Request Body

The format of the request is defined in [RFC6749] sections 4.1.3 (Access Token Request) and 6 (Refreshing an Access Token), and in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4 (Device Access Token Request).

The client can also provide the additional request parameters listed in section 3.2.5.2.1.1.

If making an OAuth on-behalf-of request, the client sends a request with the following: the *grant_type* parameter set to "urn:ietf:params:oauth:grant-type:jwt-bearer", the *requested_token_use* parameter set to "on_behalf_of", the *assertion* parameter set to an access token that the client previously received from the AD FS server (the token MUST have been issued to a resource having the same identifier as the client), and the *resource* parameter set to the identifier of the new resource that an

access token is being requested for. An OAuth on-behalf-of request is supported only for confidential clients, and the access token presented MUST have been originally issued with the scope "user_impersonation".

If making an OAuth logon certificate request, the client sends a request with the following: the *grant_type* parameter set to "urn:ietf:params:oauth:grant-type:jwt-bearer", the *requested_token_use* parameter set to "logon_cert", the *assertion* parameter set to an access token that the client previously received from the AD FS server (the token MUST have been issued to a resource having the same identifier as the client), the *csr_type* parameter set to "http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10", and the *csr* parameter set to a base64-encoded PKCS#10 certificate request ([MS-WCCE] section 2.2.2.6.1). An OAuth logon certificate request is supported only for confidential clients, and the access token presented MUST have been originally issued with the scope "logon_cert".

3.1.5.2.1.2 (Updated Section) 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), and in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.5 (Device Access Token Response).

The server can also provide the additional response parameters listed in section 3.2.5.2.1.2.

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 2.2.4.2).

3.1.5.3 Device authorization endpoint (/devicecode)

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

HTTP method	Description
POST	For a description, see section 3.2.5.3.

3.1.5.3.1 POST

For the syntax and semantics of the POST method, see section 3.2.5.3.1 with the following addition:

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

3.1.5.3.1.1 (Updated Section) Request Body

The format of the request is defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1 (Device Authorization Request).

The client can also provide the additional request parameters listed in section 3.2.5.3.1.1.

3.1.5.3.1.2 (Updated Section) Response Body

The format of the response body is defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.2 (Device Authorization Response).

The server can also provide the additional response parameters listed in section 3.2.5.3.1.2.

3.1.5.3.1.3 (Updated Section) Processing Details

The steps performed by the OAuth 2.0 client to request device verification code, user code, and verification URIs are defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1 (Device Authorization 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 2.2.4.2).

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

Proper operation of the protocol requires that the AD FS server maintains information about its current AD FS behavior level as well as configuration information about the OAuth 2.0 clients that interact with the AD FS server. This section describes an abstract data model for maintaining that configuration information.

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

Note: The notation (Public) indicates that the element can be directly accessed from outside this protocol.

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

3.2.1.1 Global Server Settings

The AD FS server maintains the following global fields:

- **ad_fs_behavior_level** (Public): Stores the AD FS behavior level, which is a specification of the functionality that is available at the AD FS server. The following values are possible.<9>

Symbolic constant	Value
AD_FS_BEHAVIOR_LEVEL_1	1
AD_FS_BEHAVIOR_LEVEL_2	2
AD_FS_BEHAVIOR_LEVEL_3	3
AD_FS_BEHAVIOR_LEVEL_4	4

3.2.1.2 (Updated Section) OAuth 2.0 client

Before initiating any protocol requests to the AD FS server, a client must first be registered with the server as described in [RFC6749] section 2.

The mechanism by which a client is registered with the server is implementation-specific and is not addressed in this protocol.

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

- **client_id:** A string field that uniquely identifies the client.
- **client_type:** Either public or confidential as described in [RFC6749] section 2.1. Confidential clients are required to authenticate to the AD FS server as described in [RFC6749] section 2.3 when making requests to the token endpoint (section 3.2.5.2). Confidential clients are only supported if the **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher.
- **Windows_client_authentication_accounts:** A collection of identifiers for any Windows accounts that can be used when authenticating this client via Windows client authentication. Any format that uniquely identifies an account can be used. This field is only applicable if the **client_type** is confidential.
- **sign_certificates:** A list of certificates registered by the client to sign future requests that use private_key_jwt as the authentication method, as described in [OIDCCore]. This field is optional and is applicable only if the **client_type** is confidential.
- **jwks_uri**: A URI that hosts a valid JSON Web Key Set (JWK Set) according to the requirements in [~~IETFDRAFT-JWK~~RFC7517]. The public keys that are present in the JWK Set are used by the client to sign future requests that use private_key_jwt as the authentication method, as described in [OIDCCore]. This field is optional and is only applicable if the **client_type** is confidential. The AD FS server stores the public keys that are present in the JWK Set that satisfy all the following requirements. Any keys that do not satisfy the requirements are ignored and not stored by the AD FS server.
  - Field **kty**, as described in [~~IETFDRAFT-JWK~~RFC7517], is "RSA".
  - Field **use**, as described in [~~IETFDRAFT-JWK~~RFC7517], is either "sig" or is not present.
  - Either fields **x5t** and **x5c** are present, as described in [~~IETFDRAFT-JWK~~RFC7517], or fields **kid**, **n**, and **e** are present, as described in [~~IETFDRAFT-JWK~~RFC7517].

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] and [IETFDRAFT-DEVICEFLOW-11].<10> 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), [IETFDRAFT-DEVICEFLOW-11] section 3.4 (Device Access Token Request), and [IETFDRAFT-DEVICEFLOW-11] section 3.5 (Device Access Token Response), 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, device verification code, or refresh token.
Device authorization endpoint (/devicecode)	As defined in [IETFDRAFT-DEVICEFLOW-11] sections 3.1 (Device Authorization Request) and 3.2 (Device Authorization Response), the device authorization endpoint is used by an OAuth 2.0 client to obtain device verification codes, user codes, and verification URLs.<11>

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}&client-request-
id={ClientRequestId}&login_hint={login_hint}&domain_hint={domain_hint}&nonce={nonce}&prompt={
```

```
prompt}&max_age={max_age}&id_token_hint={id_token_hint}&amr_values={amr_values}&mfa_max_age={
mfa_max_age}
```

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

resource_params: OPTIONAL. The client can 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.

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

domain_hint: OPTIONAL. The client can choose to specify this optional query parameter to provide a hint to the AD FS server about the backend authentication service the end user can log in to.

nonce: OPTIONAL. The client can choose to specify this optional query parameter. It is used in the same way as the nonce parameter defined in [OIDCCore] section 3.1.2.1.

prompt: OPTIONAL. The client can choose to specify this optional query parameter. It is used in the same way as the prompt parameter defined in [OIDCCore] section 3.1.2.1.

Note: Support for the *prompt* parameter depends on the AD FS server's **ad_fs_behavior_level** and the product version. See section 2.2.2 for support information.

max_age: OPTIONAL. The client can choose to specify this optional query parameter. It is used in the same way as the max_age parameter defined in [OIDCCore] section 3.1.2.1.

id_token_hint: OPTIONAL. The client can choose to specify this optional query parameter. It is used in the same way as the id_token_hint parameter defined in [OIDCCore] section 3.1.2.1.

amr_values: OPTIONAL. The client can choose to specify this optional query parameter to request that a particular authentication method be used to authenticate the user.

mfa_max_age: OPTIONAL. The client can choose to include this optional query parameter to specify the allowable elapsed time since the last time the user performed multiple factor authentication.

The request message for this method can 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 *client-request-id* 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:

- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_1, the AD FS server MUST validate that the *resource* query parameter was specified by the OAuth 2.0 client.
- If the OAuth 2.0 client specified the *resource* query parameter, 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 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 2.2.4.1.
- 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 *amr_values* query parameter and did not specify the *resource_params* query parameter:
    - If the OAuth 2.0 client specified an authentication method that is valid, the AD FS server MUST use that authentication method when authenticating the user.
    - If the authentication method that was specified 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 the invalid_request error code as defined in [RFC6749] section 4.1.2.1.

- 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 might use to log in.
- If the OAuth 2.0 client specified either the *client-request-id* 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 *client-request-id* 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's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the OAuth 2.0 client specified the *nonce* query parameter, the AD FS server includes the provided nonce value in any ID tokens issued for this request as described in [OIDCCore] section 3.1.2.1.
- If the *prompt* query parameter is supported and the OAuth 2.0 client provided a value of "none" or "login" for the *prompt* query parameter, the AD FS server follows the behavior described for the prompt parameter in [OIDCCore] section 3.1.2.1.

  Note: Support for the *prompt* parameter depends on the AD FS server's **ad_fs_behavior_level** and the product version. See section 2.2.2 for support information.
- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the OAuth 2.0 client specified the *max_age* query parameter, the AD FS server follows the processing rules for the max_age parameter described in [OIDCCore] section 3.1.2.1.
- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the OAuth 2.0 client specified the *id_token_hint* query parameter, the AD FS server follows the processing rules for the id_token_hint parameter described in [OIDCCore] section 3.1.2.1.
- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_3 or higher and the OAuth 2.0 client included the *mfa_max_age* query parameter, the AD FS server calculates, in an implementation-specific manner, the time that has elapsed since the end user last performed multiple factor authentication. If this time cannot be calculated, or if the elapsed time is greater than the calculated value, AD FS MUST attempt to actively perform multiple factor authentication of the end user. If multiple factor authentication succeeds, the mfa_auth_time claim is added to the ID token. This claim represents the time when the end user last performed multiple factor authentication. Its value is a JSON number representing the number of seconds from January 1, 1970, 00:00:00 (UTC) until the current date/time.

3.2.5.2 (Updated Section) Token endpoint (/token)

As defined in [RFC6749] section 3.2 (Token Endpoint), [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4 (Device Access Token Request), and [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.5 (Device Access Token Response), 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, device verification code, 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) and [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4 (Device Access Token Request).

3.2.5.2.1 (Updated Section) POST

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

The operation can be invoked through the following URI:

```
/token?client-request-id={ClientRequestId}
```

The format of the access token request is specified in [RFC6749] section 4.1.3 (Access Token Request) and in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4 (Device 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 *client-request-id* query parameter.

client-request-id: OPTIONAL. The client can 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 can 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 can return the server_error error code defined in this document.

3.2.5.2.1.1 (Updated Section) Request Body

The format of the request is defined in [RFC6749] sections 4.1.3 (Access Token Request) and 6 (Refreshing an Access Token), and in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4 (Device Access Token Request).

In addition to the POST body parameters described in [RFC6749] section 4.1.3 and [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4, the OAuth 2.0 client can choose to send the following additional parameters:

requested_token_use: OPTIONAL. See sections 2.2.3 and 2.2.3.1.

assertion: OPTIONAL. See sections 2.2.3 and 2.2.3.2.

resource: OPTIONAL. See sections 2.2.3 and 2.2.3.3.1.

use_windows_client_authentication: OPTIONAL. See sections 2.2.3 and 2.2.3.4.

csr: OPTIONAL. See sections 2.2.3 and 2.2.3.5.

csr_type: OPTIONAL. See sections 2.2.3 and 2.2.3.6.

tbidv2: OPTIONAL. See [~~IETFDRAFT-TOKBINDPROT~~RFC8471].

Note: If the request is an OAuth device flow request [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628], the AD FS server accepts the following alternatives for the parameters defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4:

- The grant_type parameter can be set to "device_code".
- The AD FS server will accept the presence of either the device_code parameter or the code parameter (or both). The code parameter has the same semantic meaning as device_code. If both parameters are present, the values MUST be the same.

3.2.5.2.1.2 (Updated Section) 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), and in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.5 (Device Access Token Response).

In addition to the response parameters defined in [RFC6749] and [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628], the server can also send the following response parameters:

resource: OPTIONAL. See sections 2.2.3 and 2.2.3.3.2.

x5c: OPTIONAL. See sections 2.2.3 and 2.2.3.7.

3.2.5.2.1.3 (Updated Section) 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), 5 (Issuing an Access Token), and 6 (Refreshing an Access Token), and in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.5 (Device Access Token Response).

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 *client-request-id* 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 *client-request-id* 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 (section 2.2.4.2).
- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the client is refreshing an access token ([RFC6749] section 6):
    - If the client provides a *resource* parameter in the request and the provided refresh token is a multi-resource refresh token, the AD FS server issues the access token for the resource given in this request.
    - If the client provides a *resource* parameter in the request and the provided refresh token is not a multi-resource refresh token, the AD FS server SHOULD either issue an access token for the resource given in this request, or send an error response to the OAuth 2.0 client according to the requirements of [RFC6749] section 5.2 (Error Response).<12> If sending an error, the

~~recommended~~ value for the REQUIRED error parameter of the response ~~is~~SHOULD be invalid_grant.

- If the client does not provide a *resource* parameter in the request, the AD FS server returns an access token for the same resource as was specified when the refresh token was initially granted to the client.

- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the server is returning a multi-resource refresh token, it includes a *resource* parameter in the response set to the identifier of the resource for which the current access token is being issued.

- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the *use_windows_client_authentication* parameter has a value of "true", the AD FS server authenticates the client via the HTTP Negotiate Authentication Scheme described in [RFC4559].

  - Upon successful authentication using the HTTP Negotiate Authentication Scheme, the AD FS server verifies that the account used to authenticate is one that was previously associated with the client during client registration: if there is a client registration record with **client_id** matching the *client_id* parameter in the request and the account used is included in the **Windows_client_authentication_accounts** field of the client registration record, then the client authentication is successful and processing continues. Otherwise, the AD FS server sends an error response 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 invalid_client.

- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the client is authenticating using private key jwt, as described in [OIDCCore] section 9:

  - If the client is not configured with the AD FS server to use either the **jwks_uri** or **sign_certificates** ADM element, as described in section 3.2.1.2, the AD FS server MUST send an error response 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 invalid_client.

  - If the client is configured with the AD FS server to use either the **jwks_uri** or **sign_certificates** ADM element, as described in section 3.2.1.2, the AD FS server MUST validate the JSON Web Token signature [~~IETFDRAFT-JWT~~RFC7519] using the certificate or public key identified by the **x5t** or **kid** field [~~IETFDRAFT-JWK~~RFC7517] according to the requirements in [~~IETFDRAFT-JWT~~RFC7519]. If the signature cannot be verified, the AD FS server MUST send an error response 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 invalid_client.

- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and the *grant_type* parameter has a value of "urn:ietf:params:oauth:grant-type:jwt-bearer":

  - If the *requested_token_use* parameter is not present or has any value other than "on_behalf_of" or "logon_cert", the AD FS server MUST send an error response 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 invalid_request.

  - If the *assertion* parameter is not present, the AD FS server MUST send an error response 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 invalid_request.

  - If the *resource* parameter is not present, the AD FS server MUST send an error response 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 invalid_request.

- If the *resource* parameter is invalid or not found to be registered on the AD FS server, the AD FS server MUST send an error response 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 invalid_grant.

- If the client specified by the *client_id* parameter (or otherwise identified by a client authentication method) is not a confidential client or did not provide valid client credentials according to [RFC6749] section 2.3, the AD FS server MUST send an error response 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 invalid_client.

- If the *requested_token_use* parameter has a value of "on_behalf_of":

    - If the *assertion* parameter does not contain a valid, non-expired access token previously issued by the AD FS server for the scope "user_impersonation" to the resource whose identifier matches the current client identifier (provided either in the *client_id* parameter or via the client authentication), the AD FS server MUST send an error response 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 invalid_grant.

    - The AD FS server issues a new access token to the resource given in the *resource* parameter.

- If the *requested_token_use* parameter has a value of "logon_cert":

    - If the *assertion* parameter does not contain a valid, non-expired access token that was previously issued by the AD FS server for the scope "logon_cert" to the resource whose identifier matches the current client identifier (provided either in the *client_id* parameter or by using client authentication), the AD FS server MUST send an error response 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 invalid_grant.

    - If the *csr_type* parameter is not present or is not set to a value of "http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10", the AD FS server MUST send an error response 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 invalid_request.

    - If the *csr* parameter is not present or is not a valid base64-encoded PKCS#10 request ([MS-WCCE] section 2.2.2.6.1), the AD FS server MUST send an error response 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 invalid_request.

    - The AD FS server omits the *access_token* parameter from the response and instead provides a base64-encoded CMS certificate chain or a CMC full PKI response ([MS-WCCE] section 2.2.2.8) in the *x5c* response parameter. The response that is given in the *x5c* parameter is created based upon the request in the *csr* parameter, as described in [MS-WCCE] section 3.2.1.4.2.1.4.1, with the following exceptions:

        - All fields in the original request except for **SubjectPublicKeyInfo** ([MS-WCCE] section 2.2.2.6.1) are ignored.

        - The **Subject** field of the response MUST match the identity that is represented by the original access token provided in the *assertion* parameter.

- The Extended Key Usage field ([RFC5280] section 4.2.1.12) contains the OIDs 1.3.6.1.5.5.7.3.2 (clientAuth) and 1.3.6.1.4.1.311.20.2.2 (smartcardLogin).

- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and it has not encountered any prior errors in processing, the AD FS server includes an ID token in the response as described in [OIDCCore] section 3.1.3.3.
- If the AD FS server's **ad_fs_behavior_level** is AD_FS_BEHAVIOR_LEVEL_2 or higher and it has not encountered any prior errors in processing, and a referred token-binding ID was provided (see below), the AD FS server includes a token-binding claim in the Access Token in the response, as defined in [IETFDRAFT-OAUTH2TOKBIND-08]. The token-binding ID can be provided in one of the following ways:
  - If the response is for the OAuth device flow request [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] then use the tbidv2 POST parameter provided on the device authorization endpoint (/devicecode) request.
  - Otherwise use the tbidv2 POST parameter on the current request.

3.2.5.3 (Updated Section) Device authorization endpoint (/devicecode)

As defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] sections 3.1 (Device Authorization Request) and 3.2 (Device Authorization Response), the device authorization endpoint is used by an OAuth 2.0 client to obtain device verification codes, user codes, and verification URLs. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
POST	A device authorization request issued by the OAuth 2.0 client to the device authorization endpoint of the AD FS server in accordance with the requirements of [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1 (Device Authorization Request).

3.2.5.3.1 (Updated Section) POST

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

The operation can be invoked through the following **URI**:

```
/devicecode?client-request-id={ClientRequestId}
```

The format of the device authorization token request is specified in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1 (Device Authorization Request). The OAuth 2.0 client can choose to send the client-request-id query parameter.

client-request-id: OPTIONAL. The client can 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 can 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	A request identifier, which MUST be a **GUID**.

Request header	Usage	Value
	request.	

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 5.2 (Error Response). Additionally, if the **AD FS server** encountered an error while processing the client's access token request, it can return the server_error error code defined in this document.

3.2.5.3.1.1 (Updated Section) Request Body

The format of the request is defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1 (Device Authorization Request).

In addition to the POST body parameters described in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1, the OAuth 2.0 client can choose to send the following additional parameters:

resource: OPTIONAL. See sections 2.2.3 and 2.2.3.3.1.

tbidv2: OPTIONAL. See [~~IETFDRAFT-TOKBINDPROT~~RFC8471].

3.2.5.3.1.2 (Updated Section) Response Body

The format of the response body is defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.2 (Device Authorization Response).

In addition to the response parameters defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628], the server can also send the following response parameters:

message: OPTIONAL. This parameter specifies a message with the user code and the verification URI that can be displayed to the end user.

verification_url: OPTIONAL. This parameter has the same value as the response parameter verification_uri which is described in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.2 (Device Authorization Response).

3.2.5.3.1.3 (Updated Section) Processing Details

The steps performed by the **AD FS server** to process an OAuth 2.0 client's device authorization request are defined in [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.2 (Device Authorization Response).

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 client-request-id 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 client-request-id 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 device authorization 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 (section 2.2.4.2).

- If the OAuth 2.0 client specified the resource parameter, the AD FS server MUST validate that the resource parameter specified by the OAuth 2.0 client matches a resource or **relying party** registered with the AD FS server.
- If the resource 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 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_request error code as defined in [RFC6749] section 4.1.2.1.

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
 &client-request-id=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=h
ttps%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"
}
```

4.6 Access Token Request and Response – Use of Multi-Resource Refresh Token

This example shows the sequence of requests and responses involved in the use of a multi-resource refresh token.

4.6.1 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
 &client-request-id=EC09AB2D-9655-453B-B555-3317011523E8

&resource_params=eyJQcm9wZXJ0aWVzIjpbeyJLZXkiOiJhY3IiLCJWYWx1ZSI6IndpYW9ybXVsdGlhdXRobiJ9XX
0
 &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

4.6.2 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
```

4.6.3 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
```

4.6.4 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"
}
```

4.6.5 Access Token Request – Using Multi-Resource Refresh Token

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=refresh_token&assertion=tGzv3JOkF0XG5Qx2TlKWIA&client_id=s6BhdRkqt3&code=SplxlOBeZ
QQYbYS6WxSbIA&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb&resource=https:%2F%2Fres
ource_server
```

4.6.6 Access Token Response for Multi-Resource Refresh Token Request

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":"X0RJQk5FS1NESlNabFNE",
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"U0lETkRKNDMyMzRORVVE"
}
```

4.7 Access Token Request and Response - OAuth on-behalf-of Requests

This example shows the sequence of requests and responses involved in the use of the *requested_token_use* parameter.

4.7.1 Authorization Code Request

Below is the initial authorization code request made by the client. Note that the client requests the "user_impersonation" scope, because only an access token that was granted with this scope can be used later when making an OAuth on-behalf-of request.

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3
  &resource=https%3A%2F%2Fresource_server1
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
  &scope=user_impersonation HTTP/1.1
Host: server.example.com
```

4.7.2 Authorization Code Response

In this example sequence of requests and responses, the AD FS server returns the message below in response to the request in section 4.7.1. Note that because the AD FS server has not rejected the request or indicated a reduced scope via the *scope* response parameter, this response was granted with the previously requested "user_impersonation" scope.

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

4.7.3 Initial Access Token Request

In this example sequence of requests and responses, the client redeems the authorization code received in section 4.7.2 by making the request below.

```
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=h
ttps%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

4.7.4 Initial Access Token Response

In this example sequence of requests and responses, the AD FS server returns the message below in response to the request in section 4.7.3. Note that because the AD FS server has not rejected the request or indicated a reduced scope via the *scope* response parameter, this response was granted with the "user_impersonation" scope originally requested in section 4.7.1.

```
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"
}
```

4.7.5 OAuth on-behalf-of Request

In this example sequence of requests and responses, the first resource, "https://resource_server1", having received the original access token shown in section 4.7.4, acts as a client and plays that access token to the AD FS server in order to request an access token for a new resource, "https://resource_server2".

Note that the *grant_type* is "urn:ietf:params:oauth:grant-type:jwt-bearer", the *requested_token_use* is "on_behalf_of", the *assertion* is the access token returned in section 4.7.3, the *client_id* is the same as the resource given in the initial request in section 4.7.1, that this is a confidential client, and that the *resource* parameter is for the new resource, "https://resource_server2".

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-
bearer&requested_token_use=on_behalf_of&assertion=2YotnFZFEjr1zCsicMWpAA&client_id=https%3A%2
F%2Fresource_server1&client_secret=7Fjfp0ZBr1KtDRbnfVdmIw&resource=https%3A%2F%2Fresource_ser
ver2
```

4.7.6 OAuth on-behalf-of Response

In this example sequence of requests and responses, the AD FS server returns the below in response to the request in section 4.7.5. The new access token is now intended for resource "https://resource_server2" rather than "https://resource_server1" as the token returned in section 4.7.4 was.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":"2YotnFZFEjr1zCsicMWpAA2B",
  "token_type":"bearer",
  "expires_in":3600,
}
```

4.8 Access Token Request using Windows Client Authentication

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
Authorization: Negotiate 89a8742aa8729a8b028

grant_type=authorization_code&client_id=s6BhdRkqt3&code=SplxlOBeZQQYbYS6WxSbIA&redirect_u
ri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb&use_windows_client_authentication=true
```

4.9 Authorization Code Request with nonce Parameter

Refer to [RFC6749] section 4.1.1 (Authorization Request). For more information on the *nonce* parameter, see [OIDCCore] section 3.1.2.1.

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
  &resource= https:%2F%2Fresource_server
  &client-request-id=EC09AB2D-9655-453B-B555-3317011523E8
  &nonce=abc123
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

4.10 Authorization Code Request with prompt Parameter

Refer to [RFC6749] section 4.1.1 (Authorization Request). For more information on the *prompt* parameter, see section 2.2.2 and [OIDCCore] section 3.1.2.1.

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
  &resource= https:%2F%2Fresource_server
  &client-request-id=EC09AB2D-9655-453B-B555-3317011523E8
  &prompt=login
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

4.11 Authorization Code Request with max_age Parameter

Refer to [RFC6749] section 4.1.1 (Authorization Request). For more information on the *max_age* parameter, see [OIDCCore] section 3.1.2.1.

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
  &resource= https:%2F%2Fresource_server
  &client-request-id=EC09AB2D-9655-453B-B555-3317011523E8
  &max_age=6000
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

4.12 Authorization Code Request with id_token_hint Parameter

Refer to [RFC6749] section 4.1.1 (Authorization Request). For more information on the *id_token_hint* parameter, see [OIDCCore] section 3.1.2.1.

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
  &resource= https:%2F%2Fresource_server
  &client-request-id=EC09AB2D-9655-453B-B555-3317011523E8
&id_token_hint=eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzcyI6ICJodHRwOi8vc2VydmVyLmV
4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAi
bi0wUzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5NzAKfQ.ggW8hZ1EuVLuxNuuIJ
KX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6qJp6IcmD3HP99Obi1PRs-cwh3LO-
p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJNqeGpe-
gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7TpdQyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoa
oZTF_m0_N0YzFC6g6EJbOEoRoSK5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4XUVr
WOLrLl0nx7RkKU8NXNHq-rvKMzqg
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

4.13 Access Token Request and Response - OAuth logon certificate requests

This example shows the sequence of requests and responses involved in an OAuth logon certificate request.

4.13.1 Authorization Code Request

The following message is the initial authorization code request made by the client. Note that the client requests the "logon_cert" scope, because only an access token that was granted with this scope can be used later when making an OAuth logon certificate request.

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3
  &resource=https%3A%2F%2Fresource_server1
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
  &scope=logon_cert HTTP/1.1
Host: server.example.com
```

4.13.2 Authorization Code Response

The following message is returned by the AD FS server in response to the request shown in section 4.13.1. Note that because the AD FS server has not rejected the request or indicated a reduced scope via the *scope* response parameter, this response was granted with the previously requested "logon_cert" scope.

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

4.13.3 Initial Access Token Request

The client redeems the authorization code that was received in the message shown in section 4.13.2 by making the following 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=h
ttps%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

4.13.4 Initial Access Token Response

This message is returned by the AD FS server in response to the request shown in section 4.13.3. Note that because the AD FS server has not rejected the request or indicated a reduced scope via the *scope* response parameter, this response was granted with the "logon_cert" scope originally requested in section 4.13.1.

```
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"
}
```

4.13.5 OAuth logon certificate Request

The first resource, "https://resource_server1", having received the original access token shown in section 4.13.4, acts as a client and sends that access token to the AD FS server in order to request a certificate.

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-
bearer&requested_token_use=logon_cert&assertion=2YotnFZFEjr1zCsicMWpAA&client_id=https%3A%2F%
2Fresource_server1&client_secret=7Fjfp0ZBr1KtDRbnfVdmIw&resource=https%3A%2F%2Fresource_serve
r1&csr_type=http%3A%2F%2Fschemas.microsoft.com%2Fwindows%2Fpki%2F2009%2F01%2Fenrollment%23PKC
S10&csr=MIIDYzCCA
```

Note the following:

- The *grant_type* parameter is "urn:ietf:params:oauth:grant-type:jwt-bearer".
- The *requested_token_use* parameter is "logon_cert".
- The *assertion* parameter is the access token returned in section 4.13.4.
- The *client_id* parameter is the same as the resource given in the initial request in section 4.13.1.
- This is a confidential client (as indicated by the *client_secret* parameter).
- The *csr_type* parameter is "http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10".
- The *csr* parameter is a base64-encoded PKCS#10 certificate request.

4.13.6 OAuth logon certificate Response

This message is returned by the AD FS server in response to the request shown in section 4.13.5. The response contains the certificate response in the *x5c* parameter rather than an access token.

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

```
{
  "x5c":"MIIELDCCA",
  "token_type":"bearer",
  "expires_in":3600
}
```

4.14 Access Token Request and Response – OAuth device flow request

4.14.1 (Updated Section) Device Authorization Request

Refer to [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.1 (Device Authorization Request).

```
POST /devicecode HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: server.example.com
client_id=s6BhdRkqt3&resource= https:%2F%2Fresource_server
```

4.14.2 (Updated Section) Device Authorization Response

Refer to [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.2 (Device Authorization Response).

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "device_code":"k6lpUTG5vID0ZXda4",
  "expires_in":900,
  "interval":5,
  "message":"To sign in, use a web browser to open the page
https:\/\/server.example.com\/deviceauth and enter the code JCFQCXCPL to authenticate.",
  "user_code":"JCFQCXCPL",
  "verification_uri":"https:\/\/server.example.com\/deviceauth",
  "verification_uri_complete":"
https:\/\/server.example.com\/deviceauth?user_code=JCFQCXCPL",
  "verification_url":"https:\/\/server.example.com\/deviceauth"
}
```

4.14.3 (Updated Section) Device Access Token Request

Refer to [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.4 (Device Access Token Request).

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: server.example.com
grant_type=urn:ietf:params:oauth:grant-type:device_code& client_id=s6BhdRkqt3&code=
k6lpUTG5vID0ZXda4
```

4.14.4 (Updated Section) Device Access Token Response

Refer to [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] section 3.5 (Device Access Token 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"
}
```

5 Security

5.1 (Updated Section) Security Considerations for Implementers

When processing a request with the *amr_values* parameter set to a value of "ngcmfa", ~~server implementations should not consider~~ certificate or other asymmetric key-based authentication alone ~~as satisfying~~does not satisfy multiple factors if the key that is used is present in the **msDS-KeyCredentialLink** attribute on the **user** object in Active Directory, even if the key ~~that is used~~ is protected by a smart card or requires a PIN to unlock.

Clients use the *amr_values* parameter when requesting an access token ~~that will be used~~ to register keys via the request described in [MS-KPP] section 3.1.5.1. It is expected that:

- The client only registers keys that are protected from roaming to other machines, such as by storing the private-key portion in a hardware trusted platform module (TPM).
- Multiple factor authentication ~~should have~~has been performed in order to register a key.

Keys registered this way can then be exchanged by the client for user certificates using the request described in [MS-OAPXBC] section 3.1.5.1.4. The returned certificates can be marked as smart card certificates or have a PIN associated with them.

If the server allows certificates returned by the flow described in [MS-OAPXBC] section 3.1.5.1.4 to count as multiple factor authentication, then a malicious application running in the user's context could potentially use a previously received certificate with a hardware-bound private key to get a new access token and register a new key. The new key can then be roamed to an attacker's machine, where the attacker can exchange it for further certificates or use it directly to impersonate the user.

5.2 Index of Security Parameters

None.

6 Appendix A: Full JSON Schema

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

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.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

The following tables show the relationships between Microsoft product versions or supplemental software and the roles they perform.

Windows Client releases	Client role	Server role
Windows 7 operating system	Yes	No
Windows 8 operating system	Yes	No
Windows 8.1 operating system	Yes	No
Windows 10 operating system	Yes	No

Windows Server releases	Client role	Server role
Windows Server 2008 operating system	Yes	No
Windows Server 2008 R2 operating system	Yes	No
Windows Server 2012 operating system	Yes	No
Windows Server 2012 R2 operating system	Yes	Yes
Windows Server 2016 operating system	Yes	Yes
Windows Server operating system	Yes	Yes
Windows Server 2019 operating system	Yes	Yes
Windows Server 2022 operating system	Yes	Yes

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 1.6: Support for the OAuth 2.0 protocol in AD FS is available in Windows Server 2012 R2 and later.

<2> Section 1.6: OAuth 2.0 clients running on Windows 8.1 and later implement these mandatory extensions by default.

OAuth 2.0 clients running on Windows 7, Windows 8, Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012 implement these mandatory extensions if [MSFT-WKPLJOIN] is installed. However, even with [MSFT-WKPLJOIN] installed, these products support only the *resource* and *resource_params* URI parameters.

<3> Section 2.2.2: The *prompt* parameter is not supported on Windows Server 2012 R2 unless [MSKB-3172614] is installed. Even with [MSKB-3172614] installed, the "none" value for the parameter is not supported on Windows Server 2012 R2.

The *prompt* parameter is not supported on Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012.

<4> Section 2.2.2: Even though AD_FS_BEHAVIOR_LEVEL_2 is supported on Windows Server 2016, the *amr_values* parameter is ignored on Windows Server 2016 unless [MSKB-4022723] is applied.

~~<~~<5> Section 2.2.2: Even though AD_FS_BEHAVIOR_LEVEL_3 is supported on Windows Server 2016, the *mfa_max_age* parameter is supported on Windows Server 2016 only if [MSKB-4088889] is installed.

<6> Section 2.2.3: Even though AD_FS_BEHAVIOR_LEVEL_2 is supported on Windows Server 2016, the *tbidv2* parameter is ignored on Windows Server 2016 unless [MSKB-4034658] is applied.

<7> Section 2.2.4: [RFC8628] is supported in Windows Server v1809 operating system and later and in Windows Server 2019 and later. It is also supported in Windows Server 2016 if [MSKB-4457127] is installed.

<8> Section 3.1: [~~IETFDRAFT-DEVICEFLOW-11~~RFC8628] is supported in Windows Server v1809 and later and in Windows Server 2019 and later. It is also supported in Windows Server 2016 if [MSKB-4457127] is installed.

<9> Section 3.2.1.1: The following table shows what values **ad_fs_behavior_level** can be set to on applicable Windows Server releases.

Operating System	**ad_fs_behavior_level values supported**
Windows Server 2012 R2	AD_FS_BEHAVIOR_LEVEL_1
Windows Server 2016	AD_FS_BEHAVIOR_LEVEL_1, AD_FS_BEHAVIOR_LEVEL_2, AD_FS_BEHAVIOR_LEVEL_3
Windows Server operating system	AD_FS_BEHAVIOR_LEVEL_1, AD_FS_BEHAVIOR_LEVEL_2, AD_FS_BEHAVIOR_LEVEL_3
Windows Server 2019	AD_FS_BEHAVIOR_LEVEL_1, AD_FS_BEHAVIOR_LEVEL_2, AD_FS_BEHAVIOR_LEVEL_3, AD_FS_BEHAVIOR_LEVEL_4

<10> Section 3.2.5: [IETFDRAFT-DEVICEFLOW-11] ~~[IETFDRAFT-DEVICEFLOW-11]~~ is supported in Windows Server v1809 and later and in Windows Server 2019 and later. It is also supported in Windows Server 2016 if [MSKB-4457127] is installed.

<11> Section 3.2.5: The device authorization endpoint is available in Windows Server v1809 and later and in Windows Server 2019 and later. It is also available in Windows Server 2016 if [MSKB-4457127] is installed.

<12> Section 3.2.5.2.1.3: Windows implementations return an access token for the resource given in this request even if the provided refresh token is not a multi-resource refresh token.

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
2.2.2.10 amr_values	10632 : Removed an instance of "SHOULD NOT".	Minor
5.1 Security Considerations for Implementers	10632 : Removed an instance of "should not" and an instance of "should".	Minor
7 Appendix B: Product Behavior	Updated for this version of Windows Server.	Major

9 Index

P

R

S

T

V