[MS-OAPXBC-Diff]:

OAuth 2.0 Protocol Extensions for Broker Clients

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 ~~as well as overviews of the interaction among each of these technologies~~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 ~~may~~can make copies of it in order to develop implementations of the technologies that are described in ~~the Open Specifications~~ this documentation and ~~may~~can distribute portions of it in your implementations ~~using~~that use these technologies or in your documentation as necessary to properly document the implementation. You ~~may~~can also distribute in your implementation, with or without modification, any ~~schema, IDL's~~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 ~~may~~might cover your implementations of the technologies described in the Open Specifications~~.~~ documentation. Neither this notice nor Microsoft's delivery of ~~the~~this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open ~~Specification may~~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 ~~the Open Specifications~~this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation ~~may~~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, ~~e-mail~~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 ~~do~~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 ~~standard~~standards specifications and network programming art~~,~~ and ~~assumes~~, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Copyright © 2016 Microsoft Corporation

Revision Summary

Date	Revision History	Revision Class	Comments
10/16/2015	1.0	New	Released new document.

Copyright © 2016 Microsoft Corporation

Table of Contents

1 Introduction 5
1.1 Glossary 5
1.2 References 6
1.2.1 Normative References 6
1.2.2 Informative References 7
1.3 Overview 7
1.4 Relationship to Other Protocols 7
1.5 Prerequisites/Preconditions 8
1.6 Applicability Statement 8
1.7 Versioning and Capability Negotiation 8
1.8 Vendor-Extensible Fields 9
1.9 Standards Assignments 9

2 Messages 10
2.1 Transport 10
2.2 Common Data Types 10
2.3 Directory Service Schema Elements 10

3 Protocol Details 11
3.1 OAuthBrokerExtension Client Details 11
3.1.1 Abstract Data Model 11
3.1.2 Timers 11
3.1.3 Initialization 11
3.1.4 Higher-Layer Triggered Events 12
3.1.5 Message Processing Events and Sequencing Rules 12
3.1.5.1 Token endpoint (/token) 12
3.1.5.1.1 POST (Request for Nonce) 12
3.1.5.1.1.1 Request Body 12
3.1.5.1.1.2 Response Body 12
3.1.5.1.1.3 Processing Details 12
3.1.5.1.2 POST (Request for Primary Refresh Token) 12
3.1.5.1.2.1 Request Body 13
3.1.5.1.2.2 Response Body 13
3.1.5.1.2.3 Processing Details 13
3.1.5.1.3 POST (Exchange Primary Refresh Token for Access Token) 13
3.1.5.1.3.1 Request Body 13
3.1.5.1.3.2 Response Body 13
3.1.5.1.3.3 Processing Details 13
3.1.6 Timer Events 14
3.1.7 Other Local Events 14
3.2 OAuthBrokerExtension Server Details 14
3.2.1 Abstract Data Model 14
3.2.2 Timers 14
3.2.3 Initialization 14
3.2.4 Higher-Layer Triggered Events 14
3.2.5 Message Processing Events and Sequencing Rules 14
3.2.5.1 Token endpoint (/token) 14
3.2.5.1.1 POST (Request for Nonce) 15
3.2.5.1.1.1 Request Body 15
3.2.5.1.1.2 Response Body 15
3.2.5.1.1.3 Processing Details 15
3.2.5.1.2 POST (Request for Primary Refresh Token) 16
3.2.5.1.2.1 Request Body 16
3.2.5.1.2.1.1 Username Password Authentication 16
3.2.5.1.2.1.2 User JWT Authentication 17

Copyright © 2016 Microsoft Corporation

3.2.5.1.2.1.3 Refresh Token Authentication 17
3.2.5.1.2.2 Response Body 17
3.2.5.1.2.3 Processing Details 18
3.2.5.1.3 POST (Exchange Primary Refresh Token for Access Token) 19
3.2.5.1.3.1 Request Body 19
3.2.5.1.3.2 Response Body 20
3.2.5.1.3.3 Processing Details 20
3.2.6 Timer Events 21
3.2.7 Other Local Events 21

4 Protocol Examples 22
4.1 Obtain a Nonce 22
4.2 Obtain a Primary Refresh Token 22
4.3 Obtain an Access Token 23

5 Security 25
5.1 Security Considerations for Implementers 25
5.2 Index of Security Parameters 25

6 Appendix A: Product Behavior 26

7 Change Tracking 27

8 Index 29

Copyright © 2016 Microsoft Corporation

1 Introduction

The OAuth 2.0 Protocol Extensions for Broker Clients specify extensions to [RFC6749] (The OAuth 2.0 Authorization Framework) that allow a broker client to obtain access tokens on behalf of calling clients. When no operating system version information is specified, information in this document applies to all relevant versions of Windows. Similarly, when no **AD FS behavior level** is specified, information in this document applies 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 server**
- **client identifier**
- **confidential client**
- **refresh token**
- **resource owner**

From [OIDCCore]:

- **ID token**

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative ~~and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms~~. All other sections and examples in this specification are informative.

1.1 Glossary

~~The~~This document uses the following terms ~~are specific to this document~~:

Active Directory: A general-purpose network directory service. **Active Directory** also refers to the Windows implementation of a directory service. **Active Directory** stores information about a variety of objects in the network. Importantly, user accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos are stored in **Active Directory**. **Active Directory** is either deployed as **Active Directory Domain Services (AD DS)** or Active Directory Lightweight Directory Services (AD LDS). [MS-ADTS] describes both forms. For more information, see [MS-AUTHSOD] section 1.1.1.5.2, Lightweight Directory Access Protocol (LDAP) versions 2 and 3, Kerberos, and DNS.

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. For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. For information about product versions, see [MS-ADTS] section 1. See also **Active Directory**.

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

Copyright © 2016 Microsoft Corporation

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].

JavaScript Object Notation (JSON): A text-based, data interchange format that is used to transmit structured data, typically in Asynchronous JavaScript + XML (AJAX) web applications, as described in [RFC4627]. The JSON format is based on the structure of ECMAScript (Jscript, JavaScript) objects.

JSON Web Token (JWT): A type of token that includes a set of claims encoded as a JSON object. For more information, see [IETFDRAFT-JWT].

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

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].

X.509: An ITU-T standard for public key infrastructure subsequently adapted by the IETF, as specified in [RFC3280].

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 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.

[FIPS180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

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

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

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

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

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

[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

Copyright © 2016 Microsoft Corporation

[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

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

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

[RFC7515] Jones, M., Bradley, J., and Sakimura, N., "JSON Web Signature (JWS)", RFC 7515, May 2015, http://www.rfc-editor.org/rfc/rfc7515.txt

[RFC7516] Jones, M., and Hildebrand, J., "JSON Web Encryption (JWE)", RFC 7516, May 2015, http://www.rfc-editor.org/rfc/rfc7516.txt

[SP800-108] National Institute of Standards and Technology., "Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions", October 2009, http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf

1.2.2 Informative References

None.

1.3 Overview

Active Directory Federation Services (AD FS) implements parts of the OAuth 2.0 Authorization Framework, as defined in [RFC6749] as well as the extensions described in [MS-OAPX]. In addition to these, AD FS also implements extensions to enable broker clients to retrieve tokens from an authorization server on behalf of other clients. These extensions for broker clients are specified in this document.

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

1.4 Relationship to Other Protocols

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

Copyright © 2016 Microsoft Corporation

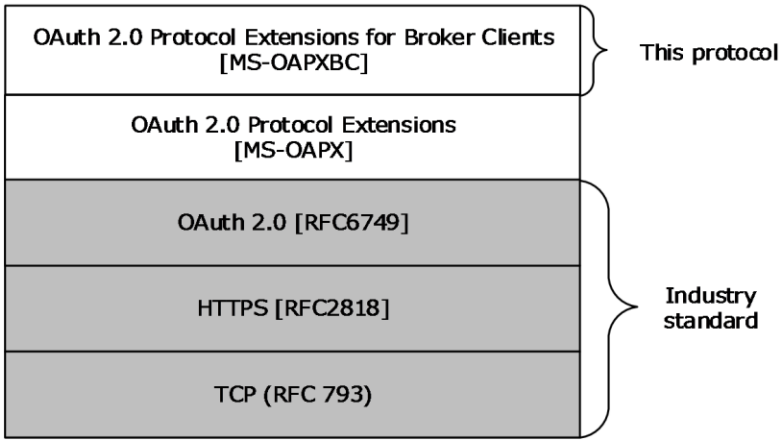

Figure 1: Protocol dependency

1.5 Prerequisites/Preconditions

The OAuth 2.0 Protocol Extensions for Broker Clients define extensions to [RFC6749] and [MS-OAPX]. A prerequisite to implementing the OAuth 2.0 Protocol Extensions is that the REQUIRED parts of [RFC6749] have been implemented on the **AD FS server**.

These 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 client runs on a device for which there is a corresponding msDS-Device object in Active Directory with the following additional requirements:

- The client has access to the private key of a device certificate. The public portion of the device certificate is stored in the altSecurityIdentities attribute of the device's msDS-Device object in Active Directory.
- The client has access to the private key of a session transport key (STK). The public portion of the STK is stored in the msDS-KeyCredentialLink attribute of the device's msDS-Device object in Active Directory.

1.6 Applicability Statement

The OAuth 2.0 Protocol Extensions for Broker Clients are supported by all AD FS servers that are at an AD FS behavior level of AD_FS_BEHAVIOR_LEVEL_2 or higher. See [MS-OAPX] section 3.2.1.1 for the formal definition of AD FS behavior level.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

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

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

Copyright © 2016 Microsoft Corporation

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

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

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Copyright © 2016 Microsoft Corporation

2 Messages

2.1 Transport

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

2.2 Common Data Types

None.

2.3 Directory Service Schema Elements

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

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

- **Active Directory Domain Services (AD DS)** [MS-ADA1] [MS-ADA2] [MS-ADSC]

Class	Attribute
msDS-Device	altSecurityIdentities msDS-KeyCredentialLink
user	msDS-KeyCredentialLink

Copyright © 2016 Microsoft Corporation

3 Protocol Details

3.1 OAuthBrokerExtension Client Details

The client role<1> of the OAuth 2.0 Protocol Extensions for Broker Clients is the initiator of requests for access tokens on behalf of other clients. The client role also stores data that is important to these requests such as a nonce and the primary refresh token.

3.1.1 Abstract Data Model

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

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. See [MS-OAPX] section 3.2.5.2.1.1 for information about the *resource* parameter.

The following elements are defined by this protocol:

Client Identifier: An identifier, represented as a string, that uniquely identifies the client to the server.

Nonce: An opaque, base64-encoded value that is provided by the server and used in requests for a primary refresh token.

Primary Refresh Token: A refresh token that the client can exchange for access tokens from the server.

Session Key: A key used to sign access token requests and decrypt access token responses. The client receives this key from the server in the response that is described in section 3.1.5.1.2.2. This key MUST be stored in a secure manner.

Device Certificate: An **X.509** certificate that represents the device on which the client runs. The client MUST have access to the private key. The altSecurityIdentities attribute of an msDS-Device object in **Active Directory** is used to store and access the public portion of the certificate.

Session Transport Key: A key used to decrypt the session key. The msDS-KeyCredentialLink attribute of an msDS-Device object in Active Directory is used to store and access the key. The msDS-Device object MUST be the same object in Active Directory that contains the public portion of the **Device Certificate**.

User Authentication Key: A key used to authenticate an end user. The msDS-KeyCredentialLink attribute of a user object in Active Directory is used to store and access the public portion of the key.

3.1.2 Timers

None.

3.1.3 Initialization

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

Copyright © 2016 Microsoft Corporation

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The resource that is accessed and manipulated by this protocol is defined in [RFC6749] and shown below for reference.

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

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

3.1.5.1 Token endpoint (/token)

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

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

3.1.5.1.1 POST (Request for Nonce)

This method requests a nonce value from the server that the client then includes in a future request for a primary refresh token, as defined in section 3.1.5.1.2.

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

```
/token
```

3.1.5.1.1.1 Request Body

The format of the request is defined in section 3.2.5.1.1.1.

3.1.5.1.1.2 Response Body

The format of the response is defined in section 3.2.5.1.1.2.

3.1.5.1.1.3 Processing Details

The nonce that is received in the response body of this request is stored in the **Nonce** abstract data model element (section 3.1.1). This nonce is used in a future request for a primary refresh token, as defined in section 3.1.5.1.2.

3.1.5.1.2 POST (Request for Primary Refresh Token)

This method requests a primary refresh token that the client can then exchange for access tokens, as defined in section 3.1.5.1.3.

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

Copyright © 2016 Microsoft Corporation

```
/token
```

3.1.5.1.2.1 Request Body

The format of the request is defined in section 3.2.5.1.2.1.

3.1.5.1.2.2 Response Body

The format of the response is defined in section 3.2.5.1.2.2.

3.1.5.1.2.3 Processing Details

Request processing:

The client uses the **Nonce** ADM element value (section 3.1.1) that it received from the server in a previous nonce request (section 3.1.5.1.1) to populate the **request_nonce** field of the request.

The client signs the request **JSON Web Token (JWT)** described in section 3.1.5.1.2.1 using the private key of the **Device Certificate** ADM element (section 3.1.1).

If using user JWT authentication as described in section 3.2.5.1.2.1.2, the client signs the **assertion** JWT using the private key of the **User Authentication Key** ADM element (section 3.1.1), and sets the **kid** field of the **assertion** JWT header to the SHA-256 hash (see [FIPS180-2~~])~~] section 6.2.2) of the public key of the **User Authentication Key** ADM element (section 3.1.1).

Response processing:

The client stores the **refresh_token** field of the response in the **Primary Refresh Token** ADM element (section 3.1.1).

The client decrypts the **session_key_jwe** field of the response by following the process described in [RFC7516] section 5.2 and by using the **Session Transport Key** ADM element (section 3.1.1). The client stores the decrypted key in the **Session Key** ADM element.

3.1.5.1.3 POST (Exchange Primary Refresh Token for Access Token)

This method exchanges a primary refresh token for an access token.

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

```
/token
```

3.1.5.1.3.1 Request Body

The format of the request is defined in section 3.2.5.1.3.1.

3.1.5.1.3.2 Response Body

The format of the response is defined in section 3.2.5.1.3.2.

3.1.5.1.3.3 Processing Details

The client first requests a primary refresh token from the server as defined in section 3.2.5.1.2. It then uses the **Primary Refresh Token** ADM element (section 3.1.1) to populate the **refresh_token** field in this request for the access token.

Copyright © 2016 Microsoft Corporation

The client derives a signing key from the **Session Key** ADM element (section 3.1.1), the constant label "AzureAD-SecureConversation", and the *ctx* value provided in the JWT header of the request by using the process described in [SP800-108]. The client uses this signing key to sign the request.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 OAuthBrokerExtension Server Details

The server role<2> of the OAuth 2.0 Protocol Extensions for Broker Clients corresponds to the notion of an authorization server as defined in [RFC6749] section 1.1 (Roles). The server role responds to the client's requests for a nonce, a primary refresh token, and access tokens.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The OAuth 2.0 Protocol Extensions for Broker Clients 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 resource accessed and manipulated by this protocol is defined in [RFC6749] and is shown below for reference.

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

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

3.2.5.1 Token endpoint (/token)

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

Copyright © 2016 Microsoft Corporation

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

3.2.5.1.1 POST (Request for Nonce)

This method requests a nonce value from the server that the client then includes in a future request for a primary refresh token, as defined in section 3.2.5.1.2.

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

```
/token
```

3.2.5.1.1.1 Request Body

To request a nonce, the client creates and sends the following request body.

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

grant_type=srv_challenge
```

3.2.5.1.1.2 Response Body

The server sends the following response body for this request.

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

{"Nonce":<nonce>}
```

The response contains a **JSON** object with one element:

Nonce (REQUIRED): An opaque, base64 URL encoded value ([RFC4648] section 5). Padding is not required ([RFC4648] section 3.2). It is to be used by the client in a future request for a primary refresh token.

3.2.5.1.1.3 Processing Details

Generation of the **Nonce** field of the response is implementation specific, provided that the nonce meets the following requirements:

- The server MUST be able to verify that any nonce value received from the client in a request for a primary refresh token (section 3.2.5.1.2) matches a nonce that was previously issued by the server.
- The server SHOULD be able to verify that any nonce value received from the client in a request for a primary refresh token matches a nonce that was issued recently (see section 3.2.5.1.2.3).
- The server SHOULD use a method that makes it difficult for an attacker to guess valid nonce values.

Copyright © 2016 Microsoft Corporation

3.2.5.1.2 POST (Request for Primary Refresh Token)

This method requests a primary refresh token that the client can then exchange for access tokens, as defined in section 3.2.5.1.3.

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

```
/token
```

3.2.5.1.2.1 Request Body

A signed request is passed as a JSON Web Token (JWT), as specified in [OIDCCore] section 6.1. The JWTs are signed either with a device key or session keys.

The format of the signed request is as follows:

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&request=<signed JWT>
```

The signed JWT format is defined in [RFC7515].

The JWT fields MUST be given the following values:

client_id (REQUIRED): A unique identifier for the broker client.<3>

scope (REQUIRED): MUST contain at least the scopes "aza" and "openid". Additional scopes can be included and follow the format described in [RFC6749] section 3.3.

request_nonce (REQUIRED): A nonce previously obtained from the server by making the request described in section 3.1.5.1.1.

Additionally, the client MUST provide user authentication in the request. The client does this by including the JWT fields from one of the following:

- Section 3.2.5.1.2.1.1 for username and password authentication.
- Section 3.2.5.1.2.1.2 if using a signed JWT for authentication.
- Section 3.2.5.1.2.1.3 if using a previous refresh token for authentication.

The signature header fields MUST be given the following values:

typ (REQUIRED): "JWT"

alg (REQUIRED): "RS256"

x5c (REQUIRED): The certificate used to sign the request, following the format described in [RFC7515] section 4.1.6.

3.2.5.1.2.1.1 Username Password Authentication

If authenticating the user by using username and password, the client includes the following fields in the JWT described in section 3.2.5.1.2.1:

grant_type (REQUIRED): "password"

username (REQUIRED): The username of the user for which the primary refresh token is requested.

Copyright © 2016 Microsoft Corporation

password (REQUIRED): The password of the user for which the primary refresh token is requested.

3.2.5.1.2.1.2 User JWT Authentication

If authenticating the user by using a signed JWT, the client includes the following fields in the JWT described in section 3.2.5.1.2.1:

grant_type (REQUIRED): "urn:ietf:params:oauth:grant-type:jwt-bearer"

assertion (REQUIRED): A signed JWT used to authenticate the user.

The JWT fields for the JWT provided in the **assertion** field MUST be given the following values:

iss (REQUIRED): The username of the user for which the primary refresh token is requested.

iat (REQUIRED): See [OIDCCore] section 6.1.

exp (REQUIRED): See [OIDCCore] section 6.1.

~~**use** (REQUIRED): "ngc"~~

aud (REQUIRED): The Issuer Identifier ([OIDCCore] section 1.2) of the server that the client is sending the request to.

~~**use** (REQUIRED): "ngc"~~

The signature header fields of the **assertion** field MUST be given the following values:

typ (REQUIRED): "JWT"

alg (REQUIRED): "RS256"

kid (REQUIRED): The identifier for the key used to sign the request.

use (REQUIRED): "ngc"

3.2.5.1.2.1.3 Refresh Token Authentication

If authenticating the user by using a previously obtained refresh token, the client includes the following fields in the JWT described in section 3.2.5.1.2.1:

grant_type (REQUIRED): "refresh_token"

refresh_token (REQUIRED): A refresh token ([RFC6749] section 1.5) that was previously obtained from the server.

3.2.5.1.2.2 Response Body

The response to the request is a JSON object with the following fields:

token_type (REQUIRED): The string "pop", indicating that the returned refresh token requires proof of possession.

refresh_token (REQUIRED): A primary refresh token. Like a refresh token described in [RFC6749] section 1.5, this can be used by clients to obtain fresh access tokens. Unlike the refresh tokens described in [RFC6749], the primary refresh token requires additional proof of possession to use as described in section 3.2.5.1.3, and can be used by any client known to the server.

refresh_token_expires_in (REQUIRED): The validity interval for the primary refresh token in seconds, as an integer.

Copyright © 2016 Microsoft Corporation

session_key_jwe (REQUIRED): A base64 URL–encoded and encrypted key value. The key is encrypted using the JSON Web Encryption (JWE) standard [RFC7516]. The relevant part of the JWE is the encrypted key section, which the client will use for future signature and decryption operations as described in section 3.1.5.1.3.

id_token (REQUIRED): An ID token for the user that is authenticated in the request, as described in [OIDCCore]. The audience for the ID token, that is, the **aud** field, is the same value given in section 3.2.5.1.2.1 for the **client_id** field. The token does not need to be signed.

3.2.5.1.2.3 Processing Details

After receiving the request, the server verifies the signature of the request and also verifies that the **request_nonce** is a nonce value previously issued by the server as defined in section 3.2.5.1.1. The server SHOULD also verify that the nonce was issued recently.<4> If the signature or nonce are invalid, the server returns the error "invalid_grant" using the format described in [RFC6749] section 5.2.

The server then processes the request as a resource owner password credentials grant (see [RFC6749] section 4.3) using the **client_id** field of the request with the following modifications:

- The server authenticates the user based on the fields of the request:
  - If the request uses username and password authentication as in section 3.2.5.1.2.1.1, the server authenticates the user as in a resource owner password credentials grant ([RFC6749] section 4.3) using the **client_id**, **scope**, and **password** fields of the request.
  - If the request uses user JWT authentication as in section 3.2.5.1.2.1.2, the server processes the request as follows:
    1. The server finds the user object in Active Directory with a user principal name ([MS-ADTS] section 5.1.1.1.1) matching the **iss** field of the **assertion** JWT.
    2. It finds the public key for the signature by finding the value of the msDS-KeyCredentialLink attribute on the user object for which the SHA-256 hash ([FIPS180-2] section 6.2.2) of the attribute value matches the **kid** field of the **assertion** JWT.
    3. The server then verifies the signature of the **assertion** JWT by using the public key that was found in the previous step.
    4. If any of the corresponding objects or values cannot be found or the signature of the **assertion** JWT is not valid, the server returns the "invalid_grant" error using the format described in [RFC6749] section 5.2.
  - If the request uses refresh token authentication as in section 3.2.5.1.2.1.3, the server validates the refresh token as in [RFC6749] section 6.
- The server uses the response format described in section 3.2.5.1.2.2 for successful responses; error responses are returned as described in [RFC6749] section 5.2.
- If the server requires user interaction at the authorization endpoint ([MS-OAPX] section 3.2.5.1) before processing this request (for example, to give consent or to provide additional authentication), the server returns the interaction_required error using the format described in [RFC6749] section 5.2.
- The server does NOT issue an access token.
- The server MUST issue a primary refresh token (in place of a normal refresh token) and include it in the **refresh_token** field of the response.
- The server MUST include an ID token [OIDCCore] in the **id_token** field response.

Copyright © 2016 Microsoft Corporation

The server finds the msDS-Device object in Active Directory that has an ~~alternateSecurityIdentifiers~~altSecurityIdentities value matching the value of the *x5c* parameter of the request header. The server then populates the **session_key_jwe** field of the response by creating a session key and encrypting it by following the process in [RFC7516] section 5.1 and by using the session transport key found in the msDS-KeyCredentialLink attribute of the previously located msDS-Device object.

3.2.5.1.3 POST (Exchange Primary Refresh Token for Access Token)

Given the primary refresh token that was obtained in section 3.2.5.1.2, this method requests an access token.

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

```
/token
```

3.2.5.1.3.1 Request Body

A signed request is passed as a JSON Web Token (JWT), as specified in [OIDCCore] section 6.1. The JWTs are signed either with a device key or session keys.

The format of the signed request is as follows:

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&request=<signed JWT>
```

The signed JWT format is defined in [RFC7515].

The JWT fields MUST be given the following values:

client_id (REQUIRED): The client identifier for the client to which an access token is to be issued, as in [RFC6749] section 1.1. If the request is made through a broker client, then this is the client identifier of the client that the broker is acting on behalf of.

scope (REQUIRED): The scope that the client requests for the access token, as in [RFC6749] section 3.3. The client MUST include the scope "openid" in the request. If the scope "aza" is included in the request, the server includes a new primary refresh token in the response.

resource (REQUIRED): The resource for which the access token is requested, as in [MS-OAPX] section 2.2.3.

iat (REQUIRED): See [OIDCCore] section 6.1.

exp (REQUIRED): See [OIDCCore] section 6.1.

grant_type (REQUIRED): "refresh_token"

refresh_token (REQUIRED): A primary refresh token that was previously received from the server. See section 3.1.5.1.2.

The JWT header fields MUST be given the following values. See [RFC7515] section 4 for field descriptions.

alg (REQUIRED): The supported value is "HS256", which indicates the algorithm used for the signature.

ctx (REQUIRED): The base64 encoded bytes used for signature key derivation.

Copyright © 2016 Microsoft Corporation

kid (REQUIRED): The only supported value is "session", which indicates that a session key is used for the signature.

3.2.5.1.3.2 Response Body

The response format is an encrypted JWT. The encrypted JWT (or JWE) format is described in [RFC7516].

The JWT header fields MUST be given the following values:

alg (REQUIRED): "dir"

enc (REQUIRED): "A256GCM"

ctx (REQUIRED): The base64-encoded binary value used for encryption key derivation.

kid (REQUIRED): "session"

After decryption, the JWT response MUST contain the following elements:

access_token (REQUIRED): An access token for the client. See the *access_token* parameter in [RFC6749] section 5.1.

token_type (REQUIRED): "bearer"

expires_in (REQUIRED): The lifetime, in seconds, of the access token. See the *expires_in* parameter in [RFC6749] section 5.1.

refresh_token (OPTIONAL): The new primary refresh token.

refresh_token_expires_in (OPTIONAL): The lifetime, in seconds, of the primary refresh token returned in the **refresh_token** field of the response.

scope (REQUIRED): The scopes included in the access token.

id_token (OPTIONAL): An ID token for the user that was authenticated in the request, as defined in [OIDCCore]. The audience for the ID token, that is, the **aud** field, is the same value given in section 3.2.5.1.3.1 for the **client_id** field. The token does not need to be signed.

3.2.5.1.3.3 Processing Details

The server verifies that the request was signed by the client with a key derived from the session key previously issued to the client using the process for deriving the signing key described in section 3.1.5.1.3.3. If the signature is invalid, the server returns the error "invalid_grant" using the format described in [RFC6749] section 5.2.

If the *resource* query parameter is invalid or is not found to be registered on the AD FS server, the AD FS server responds to the OAuth 2.0 client according to 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, which is defined in [MS-OAPX] section 2.3.1.

The server then issues an access token for the requested resource following the process in [RFC6749] section 6, using the **scope** and **refresh_token** values provided in the request, with the following exceptions:

- The response format is as described in section 3.2.5.1.3.2 for successful responses; error responses are returned as described in [RFC6749] section 5.2.
- If the server requires user interaction at the authorization endpoint ([MS-OAPX] section 3.2.5.1) before processing this request (for example, to give consent or to provide additional

Copyright © 2016 Microsoft Corporation

authentication), the server returns the interaction_required error using the format described in [RFC6749] section 5.2.

- If the scope parameter contains the scope "aza", the server issues a new primary refresh token and sets it in the **refresh_token** field of the response, as well as setting the **refresh_token_expires_in** field to the lifetime of the new primary refresh token if one is enforced.
- The scope of the issued access token is always returned in the **scope** response field, even if it is the same as the scope in the request.
- The server can include an ID token (see [OIDCCore]) in the **id_token** field of the response.

The server encrypts the response using a key that was derived by using the same process as that used for deriving the signing key, as defined in section 3.1.5.1.3.3.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

Copyright © 2016 Microsoft Corporation

4 Protocol Examples

The following sections show examples of the requests and responses that are defined by the OAuth 2.0 Protocol Extensions for Broker Clients.

Note Throughout these examples, the fictitious name "server.example.com" is used as it is used in [RFC6749].

Note Throughout these examples, the HTTP samples line breaks were added and irrelevant fields were removed to enhance readability.

4.1 Obtain a Nonce

The following example shows a request from the broker client to the AD FS server for a nonce (section 3.2.5.1.1.1) and the response from the AD FS server that contains the nonce (section 3.2.5.1.1.2).

Request:

```
POST https://server.example.com/adfs/oauth2/token/
HTTP/1.1
{
 Content-Type=application/x-www-form-urlencoded,
 Host=server.example.com,
 Content-Length=24,
 Expect=[100-continue]
}
grant_type=srv_challenge
```

Response:

```
HTTP/1.1 200 OK
{
 Content-Length=1200,
 Content-Type=application/json;charset=UTF-8
}
{"Nonce":"eyJWZXJza..."}
```

4.2 Obtain a Primary Refresh Token

The following example shows a request from the broker client to the AD FS server for a primary refresh token (section 3.2.5.1.2.1) using the obtained nonce (section 4.1) and the response from the AD FS server that contains the primary refresh token (section 3.2.5.1.2.2).

Request:

```
POST https://server.example.com/adfs/oauth2/token/
HTTP/1.1
{
 Content-Type=application/x-www-form-urlencoded,
 Host=server.example.com,
 Content-Length=4176,
 Expect=[100-continue]
}
MessageOffset:251
grant_type=urn~~:ietf:params:oauth:grant-type:jwt~~%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
&request=eyJ0eXAiOiJKV1...
```


Copyright © 2016 Microsoft Corporation

As described in sections 3.2.5.1.2.1 and 3.2.5.1.2.1.1, the content of the request parameter above is a signed JWT. An example of the raw JWT with header is given below.

```
{
  "typ":"JWT",
  "alg":"RS256",
  "x5c":["MIIEMzC..."]
}
{
  "client_id":"38aa3b87-a06d-4817-b275-7a316988d93b",
  "scope":"aza openid",
  "grant_type":"password",
  "username":"janedoe@example.com",
  "password":"password",
  "request_nonce":"eyJWZXJza..."
}
```

Response:

```
HTTP/1.1 200 OK
{
 Content-Length=6123,
 Content-Type=application/json;charset=UTF-8
}
{
 "token_type":"pop",
 "refresh_token":"rghyF1xMq2YQTbE..."
 "refresh_token_expires_in":604800,
 "session_key_jwe":"eyJlbmMiOiJBMjU2R0NNIi...",
 "id_token":"eyJ0eXAiOiJKV1QiLCJhbGci..."
}
```

4.3 Obtain an Access Token

The following example shows a request from the broker client to the AD FS server for an access token (section 3.2.5.1.3.1) using the obtained primary refresh token (section 4.2) and the response from the AD FS server that contains the access token (section 3.2.5.1.3.2).

Request:

```
POST https://server.example.com/adfs/oauth2/token/
HTTP/1.1
{
 Content-Type=application/x-www-form-urlencoded,
 Host=fs.lindft5.com,
 Content-Length=4630,
 Expect=[100-continue]
}
grant_type=urn~~:ietf:params:oauth:grant-type:jwt~~%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-
bearer
&request=eyJhbGciOiJIUz...
```

As described in section 3.2.5.1.3.1, the content of the request parameter above is a signed JWT. An example of the raw JWT with header is given below.

```
{
  "alg":"HS256",
  "ctx":"alusEDoF8fY+3p3EPnLFzBjl2DUty0Ov",
  "kid":"session"
}
```


Copyright © 2016 Microsoft Corporation

```
{
  "client_id":"s6BhdRkqt3",
  "scope":"aza openid",
  "resource":"https://resource_server1",
  "iat":1443739462,
  "exp":1443743062,
  "grant_type":"refresh_token",
  "refresh_token":"rghyF1xMq2YQTbE..."
}
```

Response:

```
HTTP/1.1 200 OK
{
 Content-Length=8739,
 Content-Type=application/json;charset=UTF-8
}
eyJhbGciOiJka...
```

As described in section 3.2.5.1.3.2, the content of the response above is an encrypted JWT. An example of the decrypted JWT with header is given below.

```
{
  "alg":"dir",
  "enc":"A256GCM",
  "ctx":"alusEDoF8fY+3p3EPnLFzBjl2DUty0Ov",
  "kid":"session"
}
{
  "access_token":"eyJ0eXAiOiJKV1QiL...",
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"xWsRetnGYw6T...",
  "refresh_token_expires_in":604800,
  "scope":"profile",
  "id_token":"eyJ0eXAiOiJKV1..."
}
```


Copyright © 2016 Microsoft Corporation

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

Copyright © 2016 Microsoft Corporation

6 Appendix A: Product Behavior

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

- Windows 10 v1511 operating system
- Windows Server 2016 ~~Technical Preview~~ operating system

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

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

<1> Section 3.1: The client role of the OAuth 2.0 Protocol Extensions for Broker Clients can be exercised by Windows client operating systems and by Windows server operating systems.

<2> Section 3.2: The server role of the OAuth 2.0 Protocol Extensions for Broker Clients can be exercised by Windows server operating systems, but not by Windows client operating systems.

<3> Section 3.2.5.1.2.1: Windows clients use the identifier "38aa3b87-a06d-4817-b275-7a316988d93b" to represent the broker client.

<4> Section 3.2.5.1.2.3: The Windows implementation of the AD FS server verifies that the nonce was issued within the last 10 minutes.

Copyright © 2016 Microsoft Corporation

7 Change Tracking

~~No table of~~ This section identifies changes ~~is available. The~~that were made to this document ~~is either new or has had no changes~~ since ~~its~~the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

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

Copyright © 2016 Microsoft Corporation

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.2.5.1.2.1.2 User JWT Authentication	73060 : Removed the redundant "use" fields from the "assertion" field.	Y	Content update.
3.2.5.1.2.3 Processing Details	73061 : Updated alternateSecurityIdentifiers to altSecurityIdentities.	Y	Content update.
4.2 Obtain a Primary Refresh Token	73059 : Updated the example for grant_type.	Y	Content update.
4.3 Obtain an Access Token	73059 : Updated the example for grant_type.	Y	Content update.

Copyright © 2016 Microsoft Corporation

8 Index

A

Applicability 8

C

Capability negotiation 8
Change tracking 27
Common data types 10

D

Directory service schema elements 10

E

Examples
  Obtain a Nonce example 22
  Obtain a Primary Refresh Token example 22
  Obtain an Access Token example 23

F

Fields - vendor-extensible 9

G

Glossary 5

I

Implementer - security considerations 25
Index of security parameters 25
Informative references 7
Introduction 5

M

Messages
  transport 10

N

Normative references 6

O

Oauthbrokerextension client
  Abstract data model 11
  Higher-layer triggered events 12
  Initialization 11
  Message processing events and sequencing rules 12
  Other local events 14
  Timer events 14
  Timers 11
Oauthbrokerextension server
  Abstract data model 14
  Higher-layer triggered events 14
  Initialization 14
  Message processing events and sequencing rules 14

Copyright © 2016 Microsoft Corporation

Other local events 21
Timer events 21
Timers 14
Overview (synopsis) 7

P

Parameters - security index 25
Preconditions 8
Prerequisites 8
Product behavior 26
Protocol Details
OAuthBrokerExtension Client 11
OAuthBrokerExtension Server 14
Protocol examples
Obtain a Nonce 22
Obtain a Primary Refresh Token 22
Obtain an Access Token 23

R

References
informative 7
normative 6
Relationship to other protocols 7

S

Security
implementer considerations 25
parameter index 25
Standards assignments 9

T

Tracking changes 27
Transport 10
common data types 10
Directory service schema elements 10

V

Vendor-extensible fields 9
Versioning 8

Copyright © 2016 Microsoft Corporation