[MS-OIDCE-Diff]:

OpenID Connect 1.0 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
7/14/2016	1.0	New	Released new document.
6/1/2017	2.0	Major	Significantly changed the technical content.
6/13/2017	3.0	Major	Significantly changed the technical content.
9/15/2017	4.0	Major	Significantly changed the technical content.

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 7
1.6 Applicability Statement 8
1.7 Versioning and Capability Negotiation 8
1.8 Vendor-Extensible Fields 8
1.9 Standards Assignments 8

2 Messages 9
2.1 Transport 9
2.2 Common Data Types 9
2.2.1 HTTP Headers 9
2.2.2 Common URI Parameters 9
2.2.3 Common Data Structures 9
2.2.3.1 ID Token 9
2.2.3.2 OpenID Provider Metadata 10

3 Protocol Details 11
3.1 OpenID Connect Extension 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 11
3.1.5 Message Processing Events and Sequencing Rules 11
3.1.5.1 Authorization endpoint (/authorize) 12
3.1.5.1.1 GET 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 12
3.1.5.1.2.1 Request Body 12
3.1.5.1.2.2 Response Body 12
3.1.5.1.2.3 Processing Details 12
3.1.5.2 Token endpoint (/token) 13
3.1.5.2.1 POST 13
3.1.5.2.1.1 Request Body 13
3.1.5.2.1.2 Response Body 13
3.1.5.2.1.3 Processing Details 13
3.1.5.3 OpenID Provider Configuration endpoint (/.well-known/openid-configuration) 13
3.1.5.3.1 GET 13
3.1.5.3.1.1 Request Body 14
3.1.5.3.1.2 Response Body 14
3.1.5.3.1.3 Processing Details 14
3.1.5.4 Logout endpoint (/logout) 14
3.1.5.4.1 GET 14
3.1.5.4.1.1 Request Body 14
3.1.5.4.1.2 Response Body 14
3.1.5.4.1.3 Processing Details 14
3.1.6 Timer Events 14
3.1.7 Other Local Events 14

3.2 OpenID Connect Extension Server Details 14
3.2.1 Abstract Data Model 15
3.2.2 Timers 15
3.2.3 Initialization 15
3.2.4 Higher-Layer Triggered Events 15
3.2.5 Message Processing Events and Sequencing Rules 15
3.2.5.1 Authorization endpoint (/authorize) 15
3.2.5.1.1 GET 16
3.2.5.1.1.1 Request Body 16
3.2.5.1.1.2 Response Body 16
3.2.5.1.1.3 Processing Details 16
3.2.5.1.2 POST 17
3.2.5.1.2.1 Request Body 17
3.2.5.1.2.2 Response Body 17
3.2.5.1.2.3 Processing Details 17
3.2.5.2 Token endpoint (/token) 17
3.2.5.2.1 POST 18
3.2.5.2.1.1 Request Body 18
3.2.5.2.1.2 Response Body 18
3.2.5.2.1.3 Processing Details 18
3.2.5.3 OpenID Provider Configuration endpoint (/.well-known/openid-configuration) 18
3.2.5.3.1 GET 19
3.2.5.3.1.1 Request Body 19
3.2.5.3.1.2 Response Body 19
3.2.5.3.1.3 Processing Details 19
3.2.5.4 Logout endpoint (/logout) 19
3.2.5.4.1 GET 20
3.2.5.4.1.1 Request Body 20
3.2.5.4.1.2 Response Body 20
3.2.5.4.1.3 Processing Details 20
3.2.6 Timer Events 20
3.2.7 Other Local Events 20

4 Protocol Examples 21
4.1 Example ID Token 21
4.2 Example OpenID Provider Configuration Response 21

5 Security 22
5.1 Security Considerations for Implementers 22
5.2 Index of Security Parameters 22

6 Appendix A: Product Behavior 23

7 Change Tracking 25

8 Index 26

1 Introduction

The OpenID Connect 1.0 Protocol Extensions specify extensions to [OIDCCore] (OpenID Connect Core 1.0) and [OIDCDiscovery] (OpenID Connect Discovery). 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]:

- client identifier
- confidential client

From [OIDCCore]:

- Issuer

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

1.1 Glossary

This document uses the following terms:

Active Directory 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].

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 [RFC7159]. 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].

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.

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].

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

user principal name (UPN): A user account name (sometimes referred to as the user logon name) and a domain name that identifies the domain in which the user account is located. This is the standard usage for logging on to a Windows domain. The format is: someone@example.com (in the form of an email address). In Active Directory, the userPrincipalName attribute of the account object, as described in [MS-ADTS].

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.

[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

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

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

[MSKB-4019472] Microsoft Corporation, "May 9, 2017—KB4019472 (OS Build 14393.1198)", https://support.microsoft.com/en-us/kb/4019472

[MSKB-4022723] Microsoft Corporation, "June ~~20~~27, 2017 - KB4022723~~",~~ (OS Build 14393.1378)", https://support.microsoft.com/en-us/kb/4022723

[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

[OIDCDiscovery] Sakimura, N., Bradley, J., Jones, M., and Jay, Edmund, "OpenID Connect Discovery 1.0 incorporating errata set 1", November 2014, http://openid.net/specs/openid-connect-discovery-1_0.html

[OIDCSession] Medeiros, B., Agarwal, N., Sakimura, N. Bradley J., and Jones, M., "OpenID Connect Session Management 1.0 – draft 28", March 2017, http://openid.net/specs/openid-connect-session-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

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

1.2.2 Informative References

None.

1.3 Overview

The OpenID Connect 1.0 identity layer enhances the OAuth 2.0 protocol by providing a means for clients to verify end-user identities. Active Directory Federation Services (AD FS) implements parts of OpenID Connect 1.0, as defined in [OIDCCore] and [OIDCDiscovery]. Additionally, AD FS implements a number of extensions to the core protocol, referred to as the OpenID Connect 1.0 Protocol Extensions, which are specified in this document.

The extensions specified in this document define additional claims to carry information about the end user, including the user principal name (UPN), a locally unique identifier, a time for password expiration, and a URL for password change. These extensions also define additional provider metadata that enable the discovery of the issuer of access tokens and give additional information about provider capabilities.

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 OpenID Connect 1.0 Protocol Extensions (this document) specify extensions to the industry standard OpenID Connect 1.0 Protocol that is defined in [OIDCCore] and [OIDCDiscovery]. These extensions are dependent on the OpenID Connect 1.0 protocol and the OAuth 2.0 protocol [RFC6749] and OAuth 2.0 Protocol Extensions [MS-OAPX], and use HTTPS [RFC2818] as the underlying transport protocol.

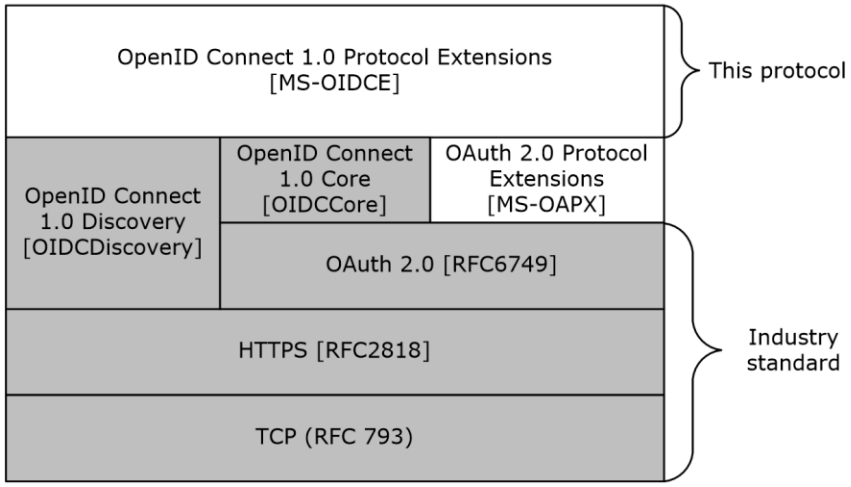

Figure 1: Protocol dependency

1.5 Prerequisites/Preconditions

The OpenID Connect 1.0 Protocol Extensions define extensions to [OIDCCore], [OIDCSession], and [OIDCDiscovery]. The following prerequisites are required for implementing the OpenID Connect 1.0 Protocol Extensions:

- The REQUIRED parts of [OIDCCore] and [OIDCDiscovery] have been implemented on the AD FS server.

- The REQUIRED parts for RP-Initiated Logout, as defined in [OIDCSession] section 5, have been implemented on the AD FS server.

The OpenID Connect 1.0 Protocol Extensions also assume that if the OpenID Connect 1.0 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 OpenID Connect 1.0 client knows its own client identifier and all relevant client authentication information if it is a confidential client.

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

1.6 Applicability Statement

The OpenID Connect 1.0 Protocol Extensions are supported by all AD FS servers that support the OpenID Connect 1.0 Protocol.<1> OpenID Connect 1.0 clients that request authorization using the OpenID Connect 1.0 protocol are required to implement the mandatory extensions defined in this protocol document.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

Supported Transports: The OpenID Connect 1.0 Protocol Extensions support only HTTPS [RFC2818] as the transport protocol.

Protocol Versions: The OpenID Connect 1.0 Protocol Extensions do not define protocol versions.

Localization: The OpenID Connect 1.0 Protocol Extensions do not return localized strings.

Capability Negotiation: The OpenID Connect 1.0 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 protocol [RFC2818] MUST be used as the transport.

2.2 Common Data Types

2.2.1 HTTP Headers

In addition to the existing set of standard HTTP headers, messages exchanged in the OpenID Connect 1.0 Protocol Extensions also use the headers defined in [MS-OAPX] section 2.2.1.

2.2.2 Common URI Parameters

In addition to the query parameters defined in [RFC6749] and [OIDCCore], the messages exchanged in the OpenID Connect 1.0 Protocol Extensions also use the URI parameters defined in [MS-OAPX] section 2.2.2.

2.2.3 Common Data Structures

In addition to the request and response parameters defined in [RFC6749] and [OIDCCore], the messages exchanged in the OpenID Connect 1.0 Protocol Extensions also use the parameters defined in [MS-OAPX] section 2.2.3.

The OpenID Connect 1.0 Protocol Extensions also define a number of extensions to existing data structures defined in [OIDCCore] and [OIDCDiscovery]. These extensions are summarized in the following table.

Data structure	Section	Description
ID Token	2.2.3.1	The ID Token is a JSON Web Token (JWT) [IETFDRAFT-JWT] that contains claims about the authentication of an end user, as defined in [OIDCCore] section 2. The OpenID Connect 1.0 Protocol Extensions define additional claims that can be included in the ID Token.
OpenID Provider Metadata	2.2.3.2	OpenID Provider Metadata is a JSON object that provides information about the OpenID connect provider, as defined in [OIDCDiscovery] section 3. The OpenID Connect 1.0 Protocol Extensions define additional values that can be included in the OpenID Provider Metadata.

2.2.3.1 ID Token

The ID Token is a JSON Web Token (JWT) that contains claims about the authentication of an end user as described in [OIDCCore] section 2.

The OpenID Connect 1.0 Protocol Extensions extend ID Token by adding a number of claims. See [OIDCCore] section 2 for definitions of the standard claims. The extended claims are defined as follows.

upn: OPTIONAL. The user principal name (UPN) of the end user represented in this ID Token.

unique_name: REQUIRED. A locally unique identifier within the Issuer for the end user. This is similar to the sub claim ([OIDCCore] section 2), but the value provided is always consistent across all clients (similar to a public subject identifier, as described in [OIDCCore] section 8). Whenever possible, the value should be human-readable and meaningful to the end user who authenticated.

pwd_exp: OPTIONAL. An integer that expresses the number of seconds until the end user's password or a similar authentication secret, such as a PIN, expires.

pwd_url: OPTIONAL. The URL that the end user can visit in order to change their password or similar authentication secret.

2.2.3.2 OpenID Provider Metadata

OpenID Provider Metadata provides information about the OpenID connect provider, as described in [OIDCDiscovery] section 3.

Note: The **end_session_endpoint** metadata field defined in the [OIDCSession] section 2.1 is required for the OpenID Connect 1.0 Protocol Extensions.<2>

The OpenID Connect 1.0 Protocol Extensions extend OpenID Provider Metadata by adding a number of fields. See [OIDCDiscovery] section 3 for the OpenID Provider Metadata with the standard fields. The extended fields are defined as follows.

access_token_issuer: OPTIONAL. A string that specifies the issuer for access tokens issued by the OpenID provider.

microsoft_multi_refresh_token: OPTIONAL. A Boolean value that indicates whether the OpenID provider supports multi-resource refresh tokens, which are refresh tokens that can be redeemed for an access token for any resource registered with the AD FS server.

capabilities: OPTIONAL. A JSON array of strings describing additional protocol capabilities that are supported by the AD FS server.<3>

3 Protocol Details

3.1 OpenID Connect Extension Client Details

The client role of the OpenID Connect 1.0 Protocol Extensions corresponds to any OpenID Connect 1.0 client that needs to request authorization to access a resource secured by an AD FS server.

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 ([MS-OAPX] section 2.2.2.1).

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 OpenID Connect 1.0 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 [OIDCCore], [OIDCSession], and [OIDCDiscovery]. They are also listed below for reference:

Resource	Description
Authorization endpoint (/authorize)	For a description, see section 3.1.5.1.
Token endpoint (/token)	For a description, see section 3.1.5.2.
OpenID Provider Configuration endpoint (/.well-known/openid-configuration)	For a description, see section 3.1.5.3.
Logout endpoint (/logout)	For a description, see section 3.1.5.4.

The HTTP responses to all the HTTP methods are defined in corresponding sections of [OIDCCore], [OIDCSession], and [OIDCDiscovery].

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

3.1.5.1 Authorization endpoint (/authorize)

As defined in [OIDCCore] sections 3.1.2, 3.2.2, and 3.3.2, the authorization endpoint performs authentication of the end user and returns an ID Token, access token, and/or authorization grant as required by the request. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
GET	For a description, see section 3.1.5.1.1.
POST	For a description, see section 3.1.5.1.2.

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 [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

3.1.5.1.1.2 Response Body

The format of the response body is defined in [OIDCCore] sections 3.1.2.5, 3.2.2.5, and 3.3.2.5 for successful responses, and sections 3.1.2.6, 3.2.2.6, and 3.3.2.6 for error responses.

3.1.5.1.1.3 Processing Details

The steps performed by the OpenID Connect 1.0 client to request authentication are defined in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

The client MUST meet the requirements specified in [MS-OAPX] section 3.1.5.1.1.3.

3.1.5.1.2 POST

For the syntax and semantics of the POST method, see section 3.2.5.1.2.

3.1.5.1.2.1 Request Body

The format of the request is defined in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

3.1.5.1.2.2 Response Body

The format of the response body is defined in [OIDCCore] sections 3.1.2.5, 3.2.2.5, and 3.3.2.5 for successful responses, and sections 3.1.2.6, 3.2.2.6, and 3.3.2.6 for error responses.

3.1.5.1.2.3 Processing Details

The steps performed by the OpenID Connect 1.0 client to request authentication are defined in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

The client MUST meet the requirements specified in [MS-OAPX] section 3.1.5.1.1.3.

3.1.5.2 Token endpoint (/token)

As defined in [OIDCCore] sections 3.1.3 and 3.3.3, the client uses the token endpoint to retrieve an ID Token, an access token, and optionally a refresh token by presenting its authorization grant or refresh token. The following HTTP methods are allowed to be performed on this resource.

HTTP method	Description
POST	For a description, see section 3.1.5.2.1.

3.1.5.2.1 POST

For the syntax and semantics of the POST method, see section 3.2.5.2.1.

The client MUST meet the requirements specified in [MS-OAPX] section 3.1.5.2.1.

3.1.5.2.1.1 Request Body

The format of the request is defined in [OIDCCore] sections 3.1.3.1 and 3.3.3.1.

The client can also provide the additional request parameters listed in [MS-OAPX] section 3.2.5.2.1.1.

3.1.5.2.1.2 Response Body

The format of the response is defined in [OIDCCore] sections 3.1.3.3 and 3.3.3.3 for successful responses, and sections 3.1.3.4 and 3.3.3.4 for error responses.

The server can also provide the additional response parameters listed in [MS-OAPX] section 3.2.5.2.1.2.

3.1.5.2.1.3 Processing Details

The steps performed by the OpenID Connect 1.0 client to request an access token are defined in [OIDCCore] sections 3.1.3.1 and 3.3.3.1.

Additionally, the OpenID Connect 1.0 client MUST expect the AD FS server to respond with an error response according to the requirements of [OIDCCore] sections 3.1.3.4 and 3.3.3.4, with the error parameter of the response set to the server_error error code as defined in [MS-OAPX] section 2.2.4.2.

3.1.5.3 OpenID Provider Configuration endpoint (/.well-known/openid-configuration)

As defined in [OIDCDiscovery] section 4, the OpenID Provider Configuration endpoint serves the OpenID provider's configuration information as a JSON object. The following HTTP methods are allowed to be performed on this endpoint.<4>

HTTP method	Description
GET	For a description, see section 3.1.5.3.1.

3.1.5.3.1 GET

For the syntax and semantics of the GET method, see section 3.2.5.3.1.

3.1.5.3.1.1 Request Body

The format of the request is defined in [OIDCDiscovery] section 4.1.

3.1.5.3.1.2 Response Body

The format of the response is defined in [OIDCDiscovery] section 4.2.

The server can also include additional fields, described in section 2.2.3.2, in the returned OpenID Provider Metadata.

3.1.5.3.1.3 Processing Details

The steps performed by the OpenID Connect 1.0 client to request the OpenID Provider Configuration are defined in [OIDCDiscovery] section 4.

3.1.5.4 Logout endpoint (/logout)

As defined in the [OIDCSession] section 5, the Logout endpoint logs out the user from the AD FS server. The following HTTP methods are allowed to be performed on this endpoint.<5>

HTTP method	Description
GET	For a description, see section 3.1.5.4.1.

3.1.5.4.1 GET

For the syntax and semantics of the GET method, see section 3.2.5.4.1.

3.1.5.4.1.1 Request Body

The format of the request is defined in [OIDCSession] section 5.

3.1.5.4.1.2 Response Body

The format of the response is defined in [OIDCSession] section 5.

3.1.5.4.1.3 Processing Details

The steps performed by the OpenID Connect 1.0 client to request the Logout endpoint are defined in [OIDCSession] section 5.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 OpenID Connect Extension Server Details

The server role of the OpenID Connect 1.0 Protocol Extensions corresponds to the notion of an OpenID provider, as defined in [OIDCCore] section 1.

The server role of this protocol implements support for the extensions defined in this document.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The OpenID Connect 1.0 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 [OIDCCore], [OIDCSession], and [OIDCDiscovery]. They are also listed below for reference:

Resource	Description
Authorization endpoint (/authorize)	For a description, see section 3.2.5.1.
Token endpoint (/token)	For a description, see section 3.2.5.2.
OpenID Provider Configuration endpoint (/.well-known/openid-configuration)	For a description, see section 3.2.5.3
Logout endpoint (/logout)	For a description, see section 3.2.5.4.

The HTTP responses to all the HTTP methods are defined in corresponding sections of [OIDCCore], [OIDCSession], and [OIDCDiscovery].

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

3.2.5.1 Authorization endpoint (/authorize)

As defined in [OIDCCore] sections 3.1.2, 3.2.2, and 3.3.2, the authorization endpoint performs authentication of the end user and returns an ID Token, an access token, and/or an authorization grant, as required by the request. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
GET	An authorization request is issued by the OpenID Connect 1.0 client to the authorization endpoint of the AD FS server in accordance with the requirements of [OIDCCore] sections 3.1.2, 3.2.2, and 3.3.2. When using the GET method, the request parameters are provided in the query string as specified in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.
POST	An authorization request is issued by the OpenID Connect 1.0 client to the authorization endpoint

HTTP method	Description
	of the AD FS server in accordance with the requirements of [OIDCCore] sections 3.1.2, 3.2.2, and 3.3.2. When using the POST method, the request parameters are provided in the POST body as specified in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

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}
```

The format of the authorization request is specified in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1. The OpenID Connect 1.0 client MUST specify the parameters marked as REQUIRED in these sections.

In addition to the [OIDCCore] parameters mentioned previously, the OpenID Connect 1.0 client uses the parameters and HTTP headers given in [MS-OAPX] section 3.2.5.1.1.

3.2.5.1.1.1 Request Body

The format of the request is defined in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

3.2.5.1.1.2 Response Body

The format of the response body is defined in [OIDCCore] sections 3.1.2.5, 3.2.2.5, and 3.3.2.5 for successful responses, and sections 3.1.2.6, 3.2.2.6, and 3.3.2.6 for error responses.

3.2.5.1.1.3 Processing Details

The steps performed by the AD FS server to respond to an authentication request are defined in [OIDCCore] sections 3.1.2, 3.2.2, and 3.3.2.

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

- The AD FS server performs the additional processing steps listed in [MS-OAPX] section 3.2.5.1.1.3
- When providing an ID Token in the response, the AD FS server includes additional claims in the ID Token:
  - The AD FS server can include a **upn** claim in the ID Token. If provided, the **upn** claim value MUST be a string that is set to the user principal name (UPN) of the end user that is represented in the ID Token. If there is no known UPN for the end user that is represented in the ID Token, the server SHOULD NOT provide a value for this claim.
  - The AD FS server MUST include a **unique_name** claim in the ID Token. The **unique_name** claim value MUST be a string that is set to a locally unique identifier for the end user within the Issuer. Construction of a locally unique identifier is implementation-specific; OpenID Connect 1.0 clients MUST NOT assume any particular format or meaning.<6>

Note The **unique_name** claim is similar to the **sub** claim ([OIDCCore] section 2), but the value provided is always consistent across all clients, similar to a public subject identifier, as described in [OIDCCore] section 8.

- The AD FS server can include a **pwd_exp** claim in the ID Token. If provided, the **pwd_exp** claim value MUST be an integer that is set to the number of seconds from the current time until the password of the end user represented in the ID Token expires. If the end user does not have a password but does have a similar authentication secret that will expire at some time, such as a personal identification number (PIN), the server can provide the number of seconds until that secret expires instead.<7>
- The AD FS server can include a **pwd_url** claim in the ID Token. If provided, the **pwd_url** claim value MUST be a string that is set to a URL that can be visited by the end user represented in the ID Token in order to change the user password. If the end user does not have a password but does have a similar authentication secret such as a PIN that can be interactively changed by visiting a particular URL, the server can provide that URL instead.<8>

3.2.5.1.2 POST

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

The method can be invoked through the following URI:

```
/authorize
```

The format of the authorization request is specified in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1. The OpenID Connect 1.0 client MUST specify the parameters marked as REQUIRED in those sections.

In addition to the [OIDCCore] parameters mentioned previously, the OpenID Connect 1.0 client uses the parameters and HTTP headers given in [MS-OAPX] section 3.2.5.1.1.

3.2.5.1.2.1 Request Body

The format of the request is defined in [OIDCCore] sections 3.1.2.1, 3.2.2.1, and 3.3.2.1.

3.2.5.1.2.2 Response Body

The format of the response body is defined in [OIDCCore] sections 3.1.2.5, 3.2.2.5, and 3.3.2.5 for successful responses, and sections 3.1.2.6, 3.2.2.6, and 3.3.2.6 for error responses.

3.2.5.1.2.3 Processing Details

The steps performed by the AD FS server to respond to an authentication request are the same as those defined in section 3.1.5.1.1.3 for the client.

3.2.5.2 Token endpoint (/token)

As defined in [OIDCCore] sections 3.1.3 and 3.3.3, the client uses the token endpoint to retrieve an ID Token, an access token, and optionally a refresh token by presenting its authorization grant or refresh token. The following HTTP methods are allowed to be performed on this resource.

HTTP method	Description
POST	A token request is issued by the OpenID Connect 1.0 client to the token endpoint of the AD FS

HTTP method	Description
	server in accordance with the requirements of [OIDCCore] sections 3.1.3 and 3.3.3.

3.2.5.2.1 POST

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

The method can be invoked through the following URI:

```
/token
```

The format of the authorization request is specified in [OIDCCore] sections 3.1.3.1 and 3.3.3.1. The OpenID Connect 1.0 client MUST specify the parameters marked as REQUIRED in these sections.

In addition to the [OIDCCore] parameters mentioned previously, the OpenID Connect 1.0 client uses the parameters and HTTP headers given in [MS-OAPX] section 3.1.5.2.1.

3.2.5.2.1.1 Request Body

The format of the request is defined in [OIDCCore] sections 3.1.3.1 and 3.3.3.1.

3.2.5.2.1.2 Response Body

The format of the response is defined in [OIDCCore] sections 3.1.3.3 and 3.3.3.3 for successful responses, and sections 3.1.3.4 and 3.3.3.4 for error responses.

3.2.5.2.1.3 Processing Details

The steps performed by the AD FS server to respond to a token request are defined in [OIDCCore] sections 3.1.3 and 3.3.3.

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

- The AD FS server performs the additional processing steps listed in [MS-OAPX] section 3.1.5.2.1.3
- When providing an ID Token in the response, the AD FS server includes additional claims in the ID Token as described in section 3.2.5.1.1.3 (specifically, the information about additional ID Token claims).

3.2.5.3 OpenID Provider Configuration endpoint (/.well-known/openid-configuration)

As defined in [OIDCDiscovery] section 4, the OpenID Provider Configuration endpoint serves the OpenID provider's configuration information as a JSON object. The following HTTP methods are allowed to be performed on this endpoint.

HTTP method	Description
GET	An OpenID Provider Configuration request is issued by the OpenID Connect 1.0 client to the OpenID Provider Configuration endpoint of the AD FS server in accordance with the requirements of [OIDCDiscovery] section 4.

3.2.5.3.1 GET

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

The method can be invoked through the following URI:

```
/.well-known/openid-configuration
```

3.2.5.3.1.1 Request Body

The format of the OpenID Provider Configuration request is specified in [OIDCDiscovery] section 4.1.

3.2.5.3.1.2 Response Body

The format of the OpenID Provider Configuration response is specified in [OIDCDiscovery] section 4.2.

3.2.5.3.1.3 Processing Details

The steps performed by the AD FS server to respond to an OpenID Provider Configuration request are defined in [OIDCDiscovery] section 4.

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

- The AD FS server includes additional fields in the OpenID Provider Metadata:
  - The AD FS server can include an **access_token_issuer** field in the OpenID Provider Metadata. If provided, the **access_token_issuer** value MUST be a string that is set to the issuer of any access tokens issued by the AD FS server.
  - The AD FS server can include a **microsoft_multi_refresh_token** field in the OpenID Provider Metadata. If provided, the **microsoft_multi_refresh_token** value is set to true if the server supports multi-resource refresh tokens.
  - The AD FS server can include a **capabilities** field in the OpenID Provider Metadata. If the server supports exchanging a primary refresh token for a user authentication certificate ([MS-OAPXBC] section 3.2.5.1.4), it includes the ~~value "winhello_cert"~~following values in the **capabilities** field~~.~~:
    - "winhello_cert" (AD FS behavior level is AD_FS_BEHAVIOR_LEVEL_2 or higher).
    - "winhello_cert_kr" (AD FS behavior level is AD_FS_BEHAVIOR_LEVEL_3 or higher). The server can include "winhello_cert_kr" in the **capabilities** field if it supports the *krctx* parameter as part of the OAuth token request ([MS-OAPXBC] section 2.2.2).

    See [MS-OAPX] section 3.2.1.1 for the formal definition of AD FS behavior level.

3.2.5.4 Logout endpoint (/logout)

As defined in [OIDCSession] section 5, the Logout endpoint logs out the user from the AD FS server. The following HTTP methods are allowed to be performed on this endpoint.<9>

HTTP method	Description
GET	A logout request is issued by the OpenID Connect 1.0 client to the Logout endpoint of the AD FS server in accordance with the requirements of

HTTP method	Description
	[OIDCSession] section 5.

3.2.5.4.1 GET

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

The method can be invoked through the following URI:

```
/logout?post_logout_redirect_uri=[redirect_uri]&state=[state]&id_token_hint=[token_hint]
```

3.2.5.4.1.1 Request Body

The format of the Logout request is specified in [OIDCSession] section 5.

3.2.5.4.1.2 Response Body

The format of the Logout response is specified in [OIDCSession] section 5.

3.2.5.4.1.3 Processing Details

The steps performed by the AD FS server to respond to a logout request are defined in [OIDCSession] section 5.

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", "server.example.com", and "janedoe@example.com" are used as they are used in [OIDCCore].

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

4.1 Example ID Token

The following example shows an ID Token that contains the additional claims defined in section 2.2.3.1:

```
{
 "iss": "https://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "at_hash": "77QmUPtjPfzWtF2AnpK9RQ",
 "c_hash": "LDktKdoQak3Pk0cnXxCltA",
 "upn": "janedoe@example.com",
 "unique_name": "janedoe@example.com",
 "pwd_exp": 5000,
 "pwd_url": "https://server.example.com/changePassword"
}
```

4.2 Example OpenID Provider Configuration Response

The following example shows a response to an OpenID Provider Configuration request. The response includes the additional fields defined in section 2.2.3.2.

```
HTTP/1.1 200 OK
Content-Type: application/json

  {
    "issuer":"https://server.example.com",
    "authorization_endpoint":"https://server.example.com/authorize/",
    "token_endpoint":"https://server.example.com/token/",
    "jwks_uri":"https://server.example.com/discovery/keys",
    "token_endpoint_auth_methods_supported":
["client_secret_post","client_secret_basic","private_key_jwt","windows_client_authentication"
],
    "response_types_supported":["code","id_token","code id_token","token id_token"],
    "response_modes_supported":["query","fragment","form_post"],

"grant_types_supported":["authorization_code","refresh_token","client_credentials","urn:ietf:
params:oauth:grant-type:jwt-bearer","implicit","password"],
    "subject_types_supported":["pairwise"],
    "scopes_supported":["profile","email","openid"],
    "id_token_signing_alg_values_supported":["RS256"],
    "token_endpoint_auth_signing_alg_values_supported":["RS256"],

"claims_supported":["aud","iss","iat","exp","auth_time","nonce","at_hash","c_hash","sub","upn
","unique_name","pwd_url","pwd_exp"],
    "access_token_issuer":"https://server.example.com",
    "microsoft_multi_refresh_token":true
  }
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

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~~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 release	**Client role**	**Server role**
Windows 10 v1511 operating system	Yes	No

Windows Server release	**Client role**	**Server role**
Windows Server 2016 operating system	Yes	Yes
Windows Server operating system	Yes	Yes

Exceptions, if any, are noted ~~below.~~in this section. If ~~a~~ an update version, service pack or ~~Quick Fix Engineering (QFE~~Knowledge Base (KB) number appears with ~~the~~a product ~~version,~~name, the behavior changed in that ~~service pack or QFE.~~update. The new behavior also applies to subsequent ~~service packs of the product~~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 OpenID Connect 1.0 protocol in AD FS is available in Windows 10 v1511 and later and in Windows Server 2016 and later.

<2> Section 2.2.3.2: Windows implementations of the AD FS server can be configured in an implementation-specific way to either return or not return the end_session_endpoint metadata.

<3> Section 2.2.3.2: The **capabilities** field is not supported on Windows 10 v1511 or Windows 10 v1607 operating system. It is also not supported on Windows Server 2016 without [MSKB-4022723] installed.

<4> Section 3.1.5.3: Windows 10 v1511 and Windows 10 v1607 do not use the extensions to OpenID Connect Discovery.

<5> Section 3.1.5.4: Windows Client operating systems (Windows 10 v1511 and later) do not implement the extensions to OpenID Connect Session Management.

<6> Section 3.2.5.1.1.3: Windows implementations of the AD FS server use the UPN or Windows account name for the locally unique identifier if either of these is available. Otherwise, the identifier depends on configuration by an administrator.

<7> Section 3.2.5.1.1.3: Windows implementations of the AD FS server include a **pwd_exp** claim only if the identity store provides a value for it.

<8> Section 3.2.5.1.1.3: Windows implementations of the AD FS server include a **pwd_url** claim only if the identity store provides a value for it.

<9> Section 3.2.5.4: The Logout endpoint is not supported on Windows Server 2016 unless [MSKB-4019472] is installed.

7 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
~~1.2.1 Normative References~~	~~Added reference for [MSKB-4022723].~~	~~Major~~
~~2.2.3.2 OpenID Provider Metadata~~	~~Added the capabilities field to the OpenID Provider metadata.~~	~~Major~~
3.2.5.3.1.3 Processing Details	~~Added the capabilities field of the OpenID Provider metadata to the processing details.~~Updated content for this version of Windows 10 and Windows Server operating system. Added the "winhello_cert_kr" value in the capabilties field.	Major
6 Appendix A: Product Behavior	Added Windows Server operating system to the list of applicable products.	Major

8 Index

A

Abstract data model
  client 11
  server 15
Applicability 8
Assumptions - initial 7

C

Capability negotiation 8
Change tracking 25
Claims - extensions to 9
Client
  abstract data model 11
  higher-layer triggered events 11
  initialization 11
  message processing 11
  other local events 14
  overview – OpenID Connect Extension 11
  sequencing rules 11
  timer events 14
  timers 11

D

Data structures - extensions to 9

E

Examples
  Example ID Token example 21
  Example OpenID Provider Configuration Response example 21
  overview 21
Extensions
  claims - end user authentication 9
  data structures 9

F

Fields - vendor-extensible 8

G

Glossary 5

H

Headers - additional 9
Higher-layer triggered events
  client 11
  server 15

I

Implementer - security considerations 22
Index of security parameters 22
Informative references 7
Initialization
  client 11
  server 15

Introduction 5

L

Localized strings 8

M

Message parameters 9
Message processing
  client 11
  server 15
Messages
  transport 9

N

Normative references 6

O

Openid connect extension client
  Abstract data model 11
  Higher-layer triggered events 11
  Initialization 11
  Message processing events and sequencing rules 11
  Other local events 14
  Timer events 14
  Timers 11
Openid connect extension server
  Abstract data model 15
  Higher-layer triggered events 15
  Initialization 15
  Message processing events and sequencing rules 15
  Other local events 20
  Timer events 20
  Timers 15
Other local events
  client 14
  server 20
Overview (synopsis) 7

P

Parameters - query 9
Parameters - security index 22
Preconditions 7
Prerequisites 7
Product behavior 23
Protocol Details
  client overview 11
  OpenID Connect Extension Client 11
  OpenID Connect Extension Server 14
  server overview 14
Protocol examples
  Example ID Token 21
  Example OpenID Provider Configuration Response 21
Protocol versions 8
Protocols - relationships 7

Q

Query parameters 9

R

References
  informative 7
  normative 6
Relationship to other protocols 7
Request and response parameters 9

S

Security
  implementer considerations 22
  parameter index 22
Sequencing rules
  client 11
  server 15
Server
  abstract data model 15
  higher-layer triggered events 15
  initialization 15
  message processing 15
  other local events 20
  overview – OpenID Connect Extension 14
  sequencing rules 15
  timer events 20
  timers 15
Standards assignments 8
Supported transports 8

T

Timer events
  client 14
  server 20
Timers
  client 11
  server 15
Tracking changes 25
Transport 9

U

URI parameters 9

V

Vendor-extensible fields 8
Versioning 8