[MS-XOAUTH]:
OAuth 2.0 Authorization Protocol Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **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 make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications 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 be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events 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 specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do 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 are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final

documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
04/27/2012	0.1	New	Released new document.

Table of Contents

1 Introduction

The OAuth 2.0 Authorization Protocol Extensions extend the OAuth 2.0 Authentication Protocol: SharePoint Extensions and the JSON Web Token (JWT) to enable server-to-server authentication.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

GUID
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
public key
realm
security token
Transmission Control Protocol (TCP)
user principal name (UPN)
X.509

The following terms are defined in [MS-OXGLOS]:

security principal
security principal identifier
security token service (STS)
Uniform Resource Identifier (URI)
XML schema

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[IETFDRAFT-JWT] Goland, Y., and Jones, M., "JSON Web Token (JWT) Specification Draft", 24 Sep 2010, http://www.ietf.org/mail-archive/web/oauth/current/msg04407.html

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-ODATA] Microsoft Corporation, "Open Data Protocol (OData) Specification".

[MS-SPS2SAUTH] Microsoft Corporation, "OAuth 2.0 Authorization Protocol: Server and Application Authentication Profile".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

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

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., et al., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

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

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, http://www.ietf.org/rfc/rfc0793.txt

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

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

[MS-OXGLOS] Microsoft Corporation, "Exchange Server Protocols Master Glossary".

1.3 Overview

These extensions specify how applications can perform server-to-server authentication using a **security token service (STS)**. For example, an e-mail service might use these extensions to authenticate itself when it makes a call to an instant messaging service. For an example of a server-to-server **security token** that a client application might send to authenticate itself, see section 4.2.

1.4 Relationship to Other Protocols

These extensions extend the OAuth 2.0 Authentication Protocol, as described in [MS-SPS2SAUTH], and JSON Web Token (JWT), as described in [IETFDRAFT-JWT].

1.5 Prerequisites/Preconditions

The client application is required to reside in the same realm as the STS to request a server-to-server security token from it.

1.6 Applicability Statement

These extensions apply only when a service call is made to or from an application that supports server-to-server authentication.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

These extensions transport messages over **TCP**, as specified in [RFC793], and do not pass any specific parameters to the transport. These extensions use **HTTPS**, as specified in [RFC2818], to secure the security tokens.

Messages sent using these extensions are not encoded by Open-Data, as specified in [MS-ODATA], and use the default character set defined by the client or the server.

For details, see section 5.

2.2 Message Syntax

This section contains common definitions used by these extensions. The syntax of the definitions uses **XML schema** as defined in [XMLSCHEMA1] and [XMLSCHEMA2], and WSDL as defined in [WSDL].

A **security principal (1)** is represented as a **security principal identifier** in the messages sent by applications. A security principal identifier is a **GUID**.

2.2.1 Namespaces

None.

2.2.2 Custom HTTP Methods

None.

2.2.3 Custom HTTP Headers

None.

2.2.4 Common URI Parameters

None.

2.2.5 Elements

None.

2.2.6 Complex Types

None.

2.2.7 Simple Types

None.

2.2.8 Attributes

None.

2.2.9 Groups

None.

2.2.10 Attribute Groups

None.

2.2.11 Common Data Structures

None.

2.3 Directory Service Schema Elements

None.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

The following table describes claims that are exchanged in server-to-server security tokens. The claim values are all of data type **STRING**, as specified in [MS-DTYP].

Claim type	Claim value description	Example claim values
aud	The targeted service for which the client issued the server-to-server security token. "aud" stands for "audience".	<security principal identifier>/<hostname>@<realm>
iss	The security principal identifier of the server-to-server security token issuer.	<security principal identifier>@<realm>
nameid	The logged on user's **user principal name (UPN)** value for the security principal (1) that made the request.	contoso\user
nbf	The time at which the server-to-server security token was created. "nbf" stands for "not before".	129592882368666656
exp	The time at which the server-to-server security token expires.	129592882368666656
trustedfordelegation	"true" if the client is trusted to delegate a user identity; otherwise, "false".	true false
identityprovider	The identity provider that authenticated the caller. This is an additional claim that the server can accept and is not required by the OAuth 2.0 Authentication Protocol specified in [MS-SPS2SAUTH].	windows forms trusted
actort	The security token issued and signed by the STS. "actort" stands for "actor token".	
smtp	The logged on user's email address. This is an additional claim that the STS adds and is not required by the OAuth 2.0 Authentication Protocol specified in [MS-SPS2SAUTH].	user@contoso.com
sip	The logged on user's sip address. This is an additional claim that the	user@contoso.com

Claim type	Claim value description	Example claim values
	STS adds and is not required by the OAuth 2.0 Authentication Protocol specified in [MS-SPS2SAUTH].	
msexchuid	A unique identifier that the STS can give the user. This is an additional claim that the STS adds and is not required by the OAuth 2.0 Authentication Protocol specified in [MS-SPS2SAUTH].	objectGUID@contoso.com
appctx	The application context. This claim contains a subset of claims that is specific to the implementation.	"smtp":"user@contoso.com", "sip":"user@contoso.com", "nameid":"user@contoso.com", "msexchuid":"objectGUID@contoso.com"

The following list describes the header fields in a server-to-server security token. The field values are all of data type **STRING**, as specified in [MS-DTYP].

- **typ**. The token type. The value should always be "JWT".
- **alg**. The algorithm used to encrypt the contents of the token.
- **xt5**. The thumbprint of the certificate used to sign the security token.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Authentication Within a Single Organization

The following procedure shows the authentication that takes place when a client application makes a call to a server application in the same organization using the server-to-server authentication protocol specified in [MS-SPS2SAUTH]. Note that the client and server applications are in fact both server applications, but we use the terms "client application" and "server application" here to distinguish them based on their roles with respect to each other.

1. The organization's IT administrator sets up an STS and configures it with the security principal identifiers for the client and server applications. The client and server applications each exchange public keys, carried in **X.509** certificates, with the STS. The administrator also configures the client and server applications to trust security tokens issued by the STS.

2. The client application makes an anonymous request to the server application.

3. The server application responds with an HTTP 401 challenge. HTTP 401 is specified in [RFC2616] and [RFC2617].

4. The client application requests a security token from the STS. It does this by sending a self-issued security token that is signed with its **public key**. The security token contains the **aud**, **iss**, **nameid**, **nbf**, **exp**, and **trustedfordelegation** claims as specified in section 3.1.1. The client request also includes a *resource* parameter, a *realm* parameter, and an optional *state* parameter. The value of the *resource* parameter is the **Uniform Resource Identifier (URI)** of the server application. For an example of a self-issued security token, see section 4.2.

5. The STS validates the public key of the security token provided by the client application, verifies that the client application is authorized to access the requested resource, and responds to the client with a server-to-server security token that is signed with a public key that the server application trusts. The security token contains the **aud**, **iss**, **nameid**, **nbf**, **exp**, and **identityprovider** claims, as specified in section 3.1.1. The response also includes an optional *state* parameter. For an example of a server-to-server security token issued by an STS, see section 4.1.

6. The client application sends the server-to-server security token to the server application.

7. The server application validates the server-to-server security token by checking the values of the **aud**, **iss**, and **exp** claims and the public key provided by the STS. It performs additional validation checks to ensure that the client application is authorized to access the requested resource. It then responds to the client application with the requested resource.

3.2.5.2 Authentication with User Information Within a Single Organization

The following procedure shows the authentication that takes place when a client application makes a call to a server application in the same organization using the server-to-server authentication protocol specified in [MS-SPS2SAUTH]. In this case, the client application also sends user information to the server application, and the server application uses the information to determine whether to return the requested resource. Note that the client and server applications are in fact both server applications, but we use the terms "client application" and "server application" here to distinguish them based on their roles with respect to each other.

1. The organization's IT administrator sets up an STS and configures it with the security principal identifiers for the client and server applications. The client and server applications each exchange public keys, carried in X.509 certificates, with the STS. The administrator also configures the client and server applications to trust security tokens issued by the STS.

2. The client application makes an anonymous request to the server application.

3. The server application responds with an HTTP 401 challenge, as specified in [RFC2616] and [RFC2617].

4. The client application requests a security token from the STS. It does this by sending a self-issued security token that is signed with its public key. The security token contains the **aud**, **iss**, **nameid**, **trustedfordelegation**, **nbf**, and **exp** claims, as specified in section 3.1.1. The client request also includes a *resource* parameter, a *realm* parameter, and an optional *state* parameter. The value of the *resource* parameter is the URI of the server application. For an example of a self-issued security token, see section 4.2.

5. The STS validates the public key of the security token provided by the client application, verifies that the client application is authorized to access the requested resource, and responds to the client with a server-to-server security token that is signed with a public key that the server application trusts. The security token contains the **aud**, **iss**, **nameid**, **nbf**, and **exp** claims. The response also includes an optional *state* parameter. For an example of a server-to-server security token issued by an STS that contains user information, see section 4.3.

6. The client application sends a self-issued security token to the server application that includes the server-to-server security token it received from the STS, as well as additional claims that contain the user information. The additional claims are the **aud**, **iss**, **nameid**, **nbf**, **exp**, **smtp**, **sip**, **msexchuid**, and **actort** claims, as specified in section 3.1.1. The **actort** claim contains the server-to-server security token provided by the STS. Note that the server-to-server security token is signed, but the user information is not. For an example of a self-issued server-to-server security token that contains user information, see section 4.4.

7. The server application validates the request by checking the user information contained in the **aud**, **iss**, and **exp** claims and the public key used to sign the security token provided by the STS. Because the user information is not signed, the server application validates the user information by checking that the values of the **aud** and **iss** claims in the user information match the values of the **aud** and **iss** claims in the server-to-server security token contained in the **actort** claim. For security considerations regarding the unsigned user information, see section 5.1.

8. The server application performs additional validation checks to ensure that the client application is authorized to access the requested resource. It then responds to the client application with the requested resource.

3.2.5.3 Authentication with Third-Party Application

The following procedure shows the authentication that takes place when a client application makes a call to a third-party application using these extensions. In this example, the client and third-party applications are in the same organization.

1. The organization's IT administrator sets up an STS and configures it with the security principal identifiers for the client and third-party applications. The client and third-party applications each exchange public keys, carried in X.509 certificates, with the STS. The administrator also configures the client and third-party applications to trust security tokens issued by the STS.

2. The client application makes an anonymous request to the third-party application.

3. The third-party application responds with an HTTP 401 challengeI, as specified in [RFC2616] and [RFC2617].

4. The client application requests a security token from the STS. It does this by sending a self-issued security token that is signed with its public key. The security token contains the **aud**, **iss**, **nameid**, **nbf**, and **exp** claims, as specified in section 3.1.1.

5. The STS validates the public key of the security token provided by the client application, verifies that the client application is authorized to access the requested resource, and returns a server-to-server security token that is signed with a public key that the third-party application trusts. The security token contains the **aud**, **iss**, **nameid**, **nbf**, **exp**, and **appctx** claims, as specified in section 3.1.1. The **appctx** claim contains information that is implementation-specific to the third-party application. For an example of a server-to-server security token that is used to access a third-party application, see section 4.5.

6. The client application sends the server-to-server security token to the third-party application.

7. The third-party application validates the server-to-server security token by checking the values of the **aud**, **iss**, and **exp** claims and the public key provided by the STS. It performs additional validation checks to ensure that the client application is authorized to access the requested resource. It then responds to the client application with the requested resource.

3.2.5.4 Realm Autodiscovery Through HTTP 401 Challenge

When the client application sends an anonymous request to the server application, the server application responds with an HTTP 401 challenge, as specified in [RFC2616] and [RFC2617]. The **Bearer** authorization header, specified in [IETFDRAFT-JWT], contains an empty value.

The HTTP 401 challenge contains the following fields.

- **client_id**. The client application's security principal identifier.
- **realm**. The server application MAY<1> return this field. This is the source **realm** of the client application.
- **trusted_issuers**. A comma-separated list of all security token issuers the server application trusts. The client can then select a security token issuer to request a security token.

3.2.5.5 Server-to-Server Security Token Contents

The server-to-server security token sent by the client application MUST be compatible with the JWT format specified in [IETFDRAFT-JWT] and [MS-SPS2SAUTH].

The server application can accept server-to-server security tokens with the claim types specified in section 3.1.1.

3.2.5.6 Server-to-Server Validation Criteria

The server application accepts a server-to-server security token that meets the following criteria:

- The server-to-server security token is signed with a trusted signing certificate from an STS that the server application trusts.
- The server-to-server security token contains an **iss** claim whose value shows that the security token is issued by an STS that the server application trusts.
- The server-to-server security token contains a **nameid** claim with the UPN value of the logged-on user.
- The server-to-server security token contains an outer token for which the value of the **iss** claim matches that of the **nameid** claim value in the inner token. The server application performs a case-sensitive comparison.
- The server-to-server security token contains an **aud** claim whose value meets the following criteria:
  - The **aud** claim value MUST contain three parts: **client_id**, **hostname**, and **realm**.
  - The value of the **client_id** part is the security principal identifier of a security principal (1) that the server application trusts. The server application performs a case-sensitive comparison.
  - The value of the **hostname** part is the host name of the server application. The server application performs a case-insensitive comparison to verify that it is the target of the request.
  - The value of the **realm** part is the source realm. The server application performs a case-sensitive comparison.

The STS uses the claims in the server-to-server security token to authenticate the caller.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Security Token Issued by STS

The following is an example of a security token issued by an STS. For more information about the claim values contained in this security token, see section 3.1.1.

```
actor:
{
    "typ":"JWT",
    "alg":"RS256",
    "x5t":"XqrnFEfsS55_vMBpHvF0pTnqeaM"
}.{
    "aud":"a0000003-0000-0ff1-ce00-000000000000/contoso.com@b84c5afe-7ced-4ce8-aa0b-
df0e2869d3c8",
    "iss":"00000001-0000-0000-c000-000000000000@b84c5afe-7ced-4ce8-aa0b-df0e2869d3c8",
    "nbf":"1323380070",
    "exp":"1323383670",
    "nameid":"00000002-0000-0ff1-ce00-000000000000@b84c5afe-7ced-4ce8-aa0b-df0e2869d3c8",
    "identityprovider":"00000001-0000-0000-c000-000000000000@b84c5afe-7ced-4ce8-aa0b-
df0e2869d3c8"
}
```

4.2 Security Token Self-Issued by Client Application

The following is an example of a security token self-issued by a client application. For more information about the claim values contained in this security token, see section 3.1.1.

```
actor:
{
    "typ":"JWT",
    "alg":"RS256",
    "x5t":"mH-TTlt-HAXC9-vjKVFtX6bAsR0"
}.{
    "aud":"a0000003-0000-0ff1-ce00-000000000000/contoso.com@EXHB-
88371dom.extest.contoso.com",
    "iss":"00000002-0000-0ff1-ce00-000000000000@EXHB-88371dom.extest.contoso.com",
    "nbf":"1323380605",
    "exp":"1323409405",
    "nameid":"00000002-0000-0ff1-ce00-000000000000@EXHB-88371dom.extest.contoso.com",
    "trustedfordelegation":"true"
}
```

4.3 Security Token Issued by STS with User Information Added by Client

The following is an example of a security token that is issued by an STS and contains user information added by the client. For more information about the claim values contained in this security token, see section 3.1.1.

```
{
    "typ":"JWT",
    "alg":"none"
}.{
```

```
    "aud":"00000003-0000-0ff1-ce00-000000000000/fabrikam.com@fabrikam-oauth-
929.extest.fabrikam.com",
    "iss":"00000001-0000-0ff1-ce00-000000000000@b84c5afe-7ced-4ce8-aa0b-df0e2869d3c8",
    "nbf":"1323380069",
    "exp":"1323408869",
    "nameid":"00000002-0000-0ff1-ce00-000000000000@BLID-EXHB-90232dom.extest.contoso.com"
}
```

4.4 Security Token Self-Issued By Client Application with User Information

The following is an example of a security token that is self-issued by a client application and contains user information. For more information about the claim values contained in this security token, see section 3.1.1.

```
{
    "typ":"JWT",
    "alg":"none"
}.{
    "aud":"a0000003-0000-0ff1-ce00-000000000000/contoso.com@EXHB-
88371dom.extest.contoso.com",
    "iss":"00000002-0000-0ff1-ce00-000000000000@EXHB-88371dom.extest.contoso.com",
    "nbf":"1323380605",
    "exp":"1323409405",
    "nameid":"ewsuser-55a83300@EXHB-88371dom.extest.contoso.com",
    "smtp":"ewsuser-55a83300@exhb-88371dom.extest.contoso.com",
    "sip":"ewsuser-55a83300@exhb-88371dom.extest.contoso.com",
    "msexchuid":"842e4c3a-0879-4973-83f9-495bb9863e18@exhb-88371dom.extest.contoso.com",
    "actort":"..actor.."
}
```

4.5 Security Token for Accessing a Third-Party Service with Extensions

The following is an example of a security token that is used to access a third-party service and contains user information. For more information about the claim values contained in this security token, see section 3.1.1.

```
{
    "aud":"00000002-0000-0ff1-ce00-000000000000/exhb-42702.exhb-
42702dom.extest.contoso.com@b84c5afe-7ced-4ce8-aa0b-df0e2869d3c8",
    "iss":"00000002-0000-0ff1-ce00-000000000000@b84c5afe-7ced-4ce8-aa0b-df0e2869d3c8",
    "nbf":"1323197012",
    "exp":"1323225812",
    "nameid":"https://www.fabrikam.com/weprintem",
    "appctx":"{"nameid":"ewsuser-cff3d495@BLID-EXHB-42702dom.extest.contoso.com",
    "smtp":"ewsuser-cff3d495@BLID-EXHB-42702dom.extest.contoso.com",
    "msexchuid":"842e4c3a-0879-4973-83f9-495bb9863e18@exhb-88371dom.extest.contoso.com"
}"
}
```

5 Security

5.1 Security Considerations for Implementers

The security considerations described in [MS-SPS2SAUTH] apply when implementing these extensions.

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:

- Microsoft® Exchange Server 15 Technical Preview
- Microsoft® SharePoint® Server 15 Technical Preview
- Microsoft® SharePoint® Foundation 15 Technical Preview

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 3.2.5.4: Exchange 15 Technical Preview does not return this parameter.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

T

V