[MS-OCDISCWS]:
Lync Autodiscover Web Service Protocol

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. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **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.

Revision Summary

Date	Revision History	Revision Class	Comments
01/20/2012	0.1	New	Released new document.
04/11/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	1.0	Major	Significantly changed the technical content.
02/11/2013	2.0	Major	Significantly changed the technical content.
07/30/2013	2.1	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

The Lync Autodiscover Web Service Protocol defines a set of operations that allows a client to discover its home server information.

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]:

fully qualified domain name (FQDN)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
UTF-8
XML namespace

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

byte order mark
JavaScript Object Notation (JSON)
Session Initiation Protocol (SIP)
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
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 technical 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.

[MS-OCAUTHWS] Microsoft Corporation, "OC Authentication Web Service Protocol".

[MS-OCSMP] Microsoft Corporation, "Microsoft Online Conference Scheduling and Management Protocol".

[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

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

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, June 2002, http://www.ietf.org/rfc/rfc3261.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC4627] Crockford, D., "The application/json Media Type for Javascript Object Notation (JSON)", RFC 4627, July 2006, http://www.ietf.org/rfc/rfc4627.txt

[WS-MetaDataExchange] Ballinger, K. et al., "Web Services Metadata Exchange (WS-MetadataExchange) Version 1.1", August 2006, http://specs.xmlsoap.org/ws/2004/09/mex/WS-MetadataExchange.pdf

[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., Beech, D., Maloney, M., Eds., 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., and Malhotra, A., Eds., "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-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

1.3 Overview

This protocol allows clients to discover web services on their home server and general topology information.

1.4 Relationship to Other Protocols

This protocol relies on **Hypertext Transfer Protocol (HTTP)** and **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** as the transmission protocols.

This protocol is dependent on [MS-OCAUTHWS] to authenticate users.

1.5 Prerequisites/Preconditions

Prior to the protocol execution, it is required that DNS entries be configured for both the internal and external Autodiscover services. The DNS entries are canonical name entries of the format lyncdiscover.<sipdomain> and lyncdiscoverinternal.<sipdomain> whose entries point to the primary Autodiscover service for the deployment's **Session Initiation Protocol (SIP)** domain. These DNS entries will allow the protocol client to discover the first hop Autodiscover service in the protocol execution. Further client details are described in section 3.2.

1.6 Applicability Statement

This protocol is applicable to clients that need to discover their home server information.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Protocol messages MUST be transported via HTTP, as specified in [RFC2616], or HTTPS, as specified in [RFC2818]. The service SHOULD be served on ports 80 and 443, but MAY be served on other ports.

Protocol messages are text-based and MUST be **UTF-8** encoded. Messages MUST NOT contain a **byte order mark**.

2.2 Message Syntax

This section contains common definitions used by this protocol. The syntax of the definitions uses the **XML schema** as defined in [XMLSCHEMA1] and [XMLSCHEMA2], and **JavaScript Object Notation (JSON)** as specified in [RFC4627].

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS]. Although this specification associates a specific XML namespace prefix for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

Prefix	Namespaces URI	Reference
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]

2.2.2 Custom HTTP Headers

The messages exchanged in this protocol use the following HTTP headers in addition to the existing set of standard HTTP headers.

Header	Description
Authorization	Specifies the client authentication header for requests to the OAuth Resource defined in section 3.1.5.5.
X-Ms-WebTicket	Specifies the client authentication header for requests to the User Resource defined in section 3.1.5.3.
X-Ms-WebTicketUrl	Specifies the Web Ticket Service location where a client can retrieve the web ticket. The Web Ticket Service is described in [MS-OCAUTHWS].

2.2.2.1 Authorization Header

The request to the OAuth resource MUST contain this authorization header. If the request does not include the authorization header, the request will be rejected with a 401 Unauthorized error code. The semantics regarding this authorization header are specified in [RFC2616] section 14.8 and more details about this authorization header are described in [MS-OCAUTHWS].

2.2.2.2 X-Ms-WebTicket Header

The request to the User resource MUST contain the X-Ms-WebTicket header. If the request does not include this header, the request will be rejected with a 401 Unauthorized error code. The semantics regarding this header are specified in [MS-OCAUTHWS].

2.2.2.3 X-Ms-WebTicketUrl Header

If the request to the User resource does not contain an X-Ms-WebTicket header, the 401 response will contain the X-Ms-WebTicketUrl header with the value being the **URL** of the Web Ticket Service, as described in [MS-OCAUTHWS]. The semantics regarding this header are specified as follows:

```
X-Ms-WebTicketUrl = "X-Ms-WebTicketUrl" ":" absolute-URI
```

The semantics regarding absolute-URI are specified in [RFC3986] section 4.3.

2.2.3 Common URI Parameters

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

URI parameter	Description
sipuri	The Session Initiation Protocol (SIP) URI of the user whose home server information is being discovered.

2.2.3.1 Sipuri URI Parameter

The request to the root resource MUST contain the *sipuri* parameter, as specified in [RFC3261] section 25.1. If the request does not contain this parameter, the client MAY eventually receive a 404 error code when proceeding with the rest of the protocol.

2.2.4 Complex Types

The following table summarizes the set of common XML schema complex type definitions defined by this specification. XML schema complex type definitions that are specific to a particular operation are described with the operation.

Complex Type	Description
AutodiscoverResponse	This is the type of the top level element **AutodiscoverResponse** that will always be returned in every Autodiscover response.
Domain	This type contains links that define topology information of the current server.
Link	This type contains an **href** attribute which indicates the URL to access the resource identified by its corresponding token attribute.
Root	This type contains a collection of links that identify resources exposed by the current server.
SipAccess	This type contains the **fully qualified domain name (FQDN)** and port of the Session Initiation Protocol (SIP) server.
User	This type contains topology information of the user's home server.

2.2.4.1 AutodiscoverResponse

The **AutodiscoverResponse** type is the type of the top level element **AutodiscoverResponse** that MUST be returned in every Autodiscover response where a body is present. This type MUST contain one and only one of the possible sub-elements: **Domain** (see the **Domain** type, section 2.2.4.2), **Root** (see the **Root** type, section 2.2.4.4), or **User** (see the **User** type, section 2.2.4.6).

The **AccessLocation** attribute MUST be present, as specified in section 2.2.5.1.

XML Schema:

```
<xs:complexType name="AutodiscoverResponse">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="Root" type=" Root"/>
    <xs:element minOccurs="0" maxOccurs="1" name="User" type=" User"/>
    <xs:element minOccurs="0" maxOccurs="1" name="Domain" type=" Domain"/>
  </xs:sequence>
  <xs:attribute name="AccessLocation" type="xs:string"/>
</xs:complexType>
```

JSON Schema:

```
AutodiscoverResponse
{
  "id": " AutodiscoverResponse",
  "type": "object",
  "properties": {
    "AccessLocation": {
      "required": true,
      "type": [
        "string",
        "null"
      ]
    },
    "Root": {
      "id": "Root",
      "required": true,
      "type": [
        "object",
        "null"
      ],
    },
    "User": {
      "id": "User",
      "required": true,
      "type": [
        "object",
        "null"
      ],
    },
    "Domain": {
      "id": "Domain",
      "required": true,
      "type": [
        "object",
```

```
        "null"
      ],
    }
}
```

2.2.4.2 Domain

The **Domain** type contains links that define topology information of the current server.

There is also a collection of links contained in the **Domain** type. These **Link** (see the **Link** type, section 2.2.4.3) elements indicate the web service URL for the corresponding tokens. The possible tokens are defined in section 3.1.5.4.

Information regarding the **SipAccess** type is in section 2.2.4.5.

XML Schema:

```
<xs:complexType name="Domain">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="SipServerInternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="1" name="SipClientInternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="1" name="SipServerExternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="1" name="SipClientExternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Link" type="Link"/>
  </xs:sequence>
</xs:complexType>
```

JSON Schema:

```
    "Domain": {
      "id": " Domain",
      "required": true,
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "SipServerInternalAccess": {
          "$ref": "SipAccess"
        },
        "SipClientInternalAccess": {
          "$ref": "SipAccess"
        },
        "SipServerExternalAccess": {
          "$ref": "SipAccess"
        },
        "SipClientExternalAccess": {
          "$ref": "SipAccess"
        },
        "Links": {
          "$ref": "Link[]"
        }
```

```
        }
      }
```

2.2.4.3 Link

The **Link** type is used to identify the URL to access a server resource. The URL which the **Link** element gives access to is present in the **href** attribute. The **token** attribute of the link is a unique string that identifies the type of resource to which the corresponding href will give access. The **token** (section 2.2.5) and **href** (section 2.2.5) attributes MUST be present in every **Link** element. For more details about **token**, see section 2.2.5.5.

XML Schema:

```
<xs:complexType name="Link">
  <xs:attribute name="token" type="xs:string"/>
  <xs:attribute name="href" type="xs:string"/>
</xs:complexType>
```

JSON Schema:

```
          "items": {
            "id": "Link",
            "type": [
              "object",
              "null"
            ],
            "properties": {
              "token": {
                "required": true,
                "type": [
                  "string",
                  "null"
                ]
              },
              "href": {
                "required": true,
                "type": [
                  "string",
                  "null"
                ]
              }
            }
```

2.2.4.4 Root

The **Root** type contains links that define URLs that are served by the responding Autodiscover service. See the **Link** type in section 2.2.4.3 for the **Link** element. The corresponding tokens that identify these links are defined in section 3.1.5.2.

XML Schema:

```
<xs:complexType name="Root">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Link" type="Link"/>
```

```
    </xs:sequence>

  </xs:complexType>
```

JSON Schema:

```
"Root": {
      "id": "Root",
      "required": true,
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "Links": {
          "id": "Link[]",
          "required": true,
          "type": [
            "array",
            "null"
          ],
          "items": {
            "id": "Link",
            "type": [
              "object",
              "null"
            ],
            "properties": {
              "token": {
                "required": true,
                "type": [
                  "string",
                  "null"
                ]
              },
              "href": {
                "required": true,
                "type": [
                  "string",
                  "null"
                ]
              }
            }
          }
}
```

2.2.4.5 SipAccess

The **SipAccess** type reveals the fully qualified domain name (FQDN) and port of the Session Initiation Protocol (SIP) server. Protocol clients can use the **fqdn** and **port** attributes to connect to the SIP server front-end. See section 2.2.5 for the **fqdn** and **port** attributes. The **SipAccess** type is contained in the User and Domain resources under the elements. The type can be defined in four different elements that give different meanings to how the contents SHOULD be used. The behavior is described in the following table:

Type	Description
SipServerInternalAccess	This element reveals the SIP server FQDN and listening port to be used by SIP servers in the internal network.
SipServerExternalAccess	This element defines the SIP server FQDN and listening port to be used by SIP servers in the external network.
SipClientInternalAccess	This element defines the SIP server FQDN and listening port to be used by SIP clients in the internal network.
SipClientExternalAccess	This element defines the SIP server FQDN and listening port to be used by SIP clients in the external network.

XML Schema:

```
<xs:complexType name="SipAccess">
  <xs:attribute name="fqdn" type="xs:string" />
  <xs:attribute name="port" type="xs:string" />
</xs:complexType>
```

JSON Schema:

```
"properties": {
  "SipAccess": {
    "id": "SipAccess",
    "required": true,
    "type": [
      "object",
      "null"
    ],
    "properties": {
      "fqdn": {
        "required": true,
        "type": [
          "string",
          "null"
        ]
      },
      "port": {
        "required": true,
        "type": [
          "string",
          "null"
        ]
      }
    }
  }
```

2.2.4.6 User

The **User** type contains links that define topology information for the current user.

There is also a collection of links contained in the **User** type. These **link** (see the **Link** type in section 2.2.4.3) elements indicate the web service URL for the corresponding tokens. The possible tokens are defined in section 3.1.5.3.

Information regarding the **SipAccess** type is found in section 2.2.4.5.

XML Schema:

```
<xs:complexType name="User">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="SipServerInternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="1" name="SipClientInternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="1" name="SipServerExternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="1" name="SipClientExternalAccess" type="SipAccess"
/>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="link" type="link"/>
  </xs:sequence>
</xs:complexType>
```

JSON Schema:

```
    "User": {
      "id": "User",
      "required": true,
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "SipServerInternalAccess": {
          "id": "SipAccess",
          "required": true,
          "type": [
            "object",
            "null"
          ],
          "properties": {
            "fqdn": {
              "required": true,
              "type": [
                "string",
                "null"
              ]
            },
            "port": {
              "required": true,
              "type": [
                "string",
                "null"
              ]
            }
          }
        },
        "SipClientInternalAccess": {
          "$ref": "SipAccess"
        },
        "SipServerExternalAccess": {
          "$ref": "SipAccess"
        },
```

```
        "SipClientExternalAccess": {
          "$ref": "SipAccess"
        },
        "Links": {
          "$ref": "Link[]"
        }
      }
  }
```

2.2.5 Attributes

The following table summarizes the set of common XML schema attribute definitions defined by this specification. XML schema attributes that are specific to a particular operation are described with the operation.

Attribute	Description
AccessLocation	An indicator of whether the client is connecting from inside or outside of the network.
Fqdn	The fully qualified domain name (FQDN) of the Session Initiation Protocol (SIP) server.
href	A URL that is relative to the service's root URL
Port	The listening port of the SIP server.
token	A classification that is used to infer the purpose and use of the attributed item. Somewhat analogous to a type in traditional, compiled programming languages

2.2.5.1 AccessLocation

This attribute MUST have values of either "internal" or "external". A value of "internal" indicates that the current server is hosting internal network traffic, while a value of "external" indicates that the current server is hosting external network traffic.

2.2.5.2 Fqdn

This attribute specifies the fully qualified domain name (FQDN) of the Session Initiation Protocol (SIP) server.

2.2.5.3 Href

This attribute specifies a URL that is relative to the service's root URL.

2.2.5.4 Port

This attribute specifies the listening port of the SIP server.

2.2.5.5 Token

The following table summarizes the set of possible tokens and their descriptions. For every **token** attribute, there MUST be a corresponding **href** attribute. The **href** attribute contains a URL that the client can use to access related services. Semantics on the entry points to each URL can be found in the following table, however no details are provided beyond the entry points.

Token	Description
"Internal/Autodiscover"	Identifies the Autodiscover Service's Root entry point that can be accessed from the internal network. See section 3.1.5.1 for semantics on this URL.
"External/Autodiscover"	Identifies the Autodiscover Service's Root entry point that can be accessed from the external network. See section 3.1.5.1 for semantics on this URL.
"Internal/AuthBroker"	Identifies the AuthBroker Service's entry point that can be accessed from the internal network. This service exposes a SOAP endpoint and publishes a MEX document. The Authbroker Service is specified as the Authentication Broker Service in [MS-OCAUTHWS]. MEX documents are specified in [WS-MetaDataExchange].
"External/AuthBroker"	Identifies the AuthBroker Service's entry point that can be accessed from the external network. This service exposes a SOAP endpoint and publishes a MEX document. The Authbroker Service is specified as the Authentication Broker Service in [MS-OCAUTHWS]. MEX documents are specified in [WS-MetaDataExchange].
"Internal/Ucwa"	Identifies the Ucwa Service's entry point that can be accessed from the internal network. This service exposes a REST endpoint that supports a GET operation. The Ucwa Service is defined in [MS-OCSMP].
"External/Ucwa"	Identifies the Ucwa Service's entry point that can be accessed from the external network. This service exposes a REST endpoint that supports a GET operation. The Ucwa Service is defined in [MS-OCSMP].
"Redirect"	Identifies an Autodiscover Service URL which is the Root entry point of the next hop in the discovery flow. See section 3.1.5.2 for semantics on this URL.
"User"	Identifies the Autodiscover Service URL for the User entry point. Semantics on this URL can be found in section 3.1.5.3.
"Domain"	Identifies the Autodiscover Service URL for the Domain entry point. Semantics on this URL can be found in section 3.1.5.4.
"OAuth"	Identifies the Autodiscover Service URL for the OAuth entry point. Semantics on this URL can be found in section 3.1.5.5.

3 Protocol Details

3.1 Server Details

This section specifies details that pertain to the protocol server behavior.

3.1.1 Abstract Data Model

None.

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

The following table lists all the exposed resources by the Autodiscover Service.

Resource	Description
Root	Service entry point that allows navigation to child resources.
User	Provides user home server information.
Domain	Provides general topology information about the current server.
OAuth	Provides user home server information.

3.1.5.1 Common Processing Details

The following table lists all the possible Accept headers handled by the Server, and the Content-Type returned in the response.

Accept Header Values	Returned Content-Type
No Accept header specified	application/vnd.microsoft.rtc.autodiscover+json;v=1
application/vnd.microsoft.rtc.autodiscover+xml;v=1	application/vnd.microsoft.rtc.autodiscover+xml;v=1
Application/vnd.microsoft.rtc.autodiscover+json;v=1	application/vnd.microsoft.rtc.autodiscover+json;v=1
/	application/vnd.microsoft.rtc.autodiscover+json;v=1

If none of the Accept headers listed in the table are sent with any request, the server will respond with a 406 response.

If the returned Content-Type is "application/vnd.microsoft.rtc.autodiscover+xml;v=1", then the response will be in XML format and MUST be de-serialized according to the schemas in section 2.2.4.

If the returned Content-Type is "application/vnd.microsoft.rtc.autodiscover+json;v=1", then the response will be in JSON format and MUST be de-serialized according to the schemas in section 2.2.4.

3.1.5.2 Root

The Root resource is the entry point to an instance of the Autodiscover Service. All requests to this resource MUST contain the *sipuri* query parameter defined in section 2.2.3.1.

The resource URL for this parameter is https://<fqdn>/autodiscover/autodiscoverservice.svc/root?sipuri=<sipuri> . The server MAY also listen on HTTP depending on the server configuration. If a request is made to the HTTP ROOT resource, the server MUST respond with a 200 OK and a **Root** element that contains a single **Link** element with token named "Redirect" and an **href** that points to the next hop URL that the client SHOULD request.

If the *sipuri* of the user cannot be processed by the current Autodiscover Service, the Autodiscover Service MUST respond with a 200 OK with a **Root** element that contains a **Link** with a token named "Redirect" and an **href** that points to the next hop URL that can handle the request.

If the current Autodiscover Service can handle the request, then the server MUST respond with a 200 OK and a **Root** element that contains a list of Links to child resources of the current Autodiscover Service. See section 2.2.4.4 for the schema of the **Root** element. If there is no "Redirect" link present in the **Root** element, then there MUST be "User" and "Domain" tokens present. The "OAuth" token MAY be present depending on the server version.

3.1.5.2.1 GET

This section details the Request and Response body for the supported GET HTTP operation to the Root resource.

3.1.5.2.1.1 Request Body

None.

3.1.5.2.1.2 Response Body

The Autodiscover Service MUST respond with an **AutodiscoverResponse** element that contains a **Root** element. These elements are defined in sections 2.2.4.1 and 2.2.4.4.

3.1.5.3 User

The User resource exposes topology information of a user's home server. The client MUST discover the User URL by parsing the href of the **Link** element in the GET response from the Root resource.

If the request to the User resource does not contain a valid X-Ms-WebTicket header as specified in section 2.2.2.2, the server MUST respond with a 401 Unauthorized response.

If a valid X-Ms-WebTicket header is provided and the user's home server information is unknown, the server MUST respond with a 404 response code and an empty body.

If a valid X-Ms-WebTicket header is provided and the user's home server information exists on a separate server, the server MUST respond with a 200 response code and a **User** element. The **User**

element MUST contain only one **Link** element with a "Redirect" Token. Semantics of the "Redirect" Token are defined in section 2.2.5.5.

If a valid X-Ms-WebTicket header is provided and the user's home server information exists on the current server, the server MUST respond with a 200 response code and a **User** element in the body. The **User** element MAY contain any of the following links depending on what is configured in the topology.

1. Internal/Autodiscover
2. External/Autodiscover
3. Internal/AuthBroker
4. External/AuthBroker
5. Internal/Ucwa
6. External/Ucwa

The **SipAccess** types MAY also be present in the response depending on what is configured in the topology.

3.1.5.3.1 GET

This section details the Request and Response body for the supported GET HTTP operation to the User resource.

3.1.5.3.1.1 Request Body

None.

3.1.5.3.1.2 Response Body

If the response code is 200 OK, then the response MUST contain an **AutodiscoverResponse** element that contains a **User** element. These elements are defined in sections 2.2.4.1 and 2.2.4.6.

If the response code is 401 Unauthorized, then there will be an HTML formatted body.

If the response code is 404 Not Found, then there will be an empty body.

3.1.5.4 Domain

The Domain resource exposes information about the topology information of the current server to the client. The client MUST discover the Domain URL by parsing the **Link** element in the GET response from the Root resource.

The Domain resource MUST respond with a 200 OK to every request and contain a **Domain** element in the body. The **Domain** element is defined in section 2.2.4.2.

The **Domain** element MAY contain any of the following links, depending on what is configured in the topology. The links are identified by tokens for which semantics are defined in section 2.2.5.5.

1. Internal/Autodiscover
2. External/Autodiscover

3. Internal/AuthBroker

4. External/AuthBroker

5. Internal/Ucwa

6. External/Ucwa

The **SipAccess** types MAY also be present in the response, depending on what is configured in the topology.

3.1.5.4.1 GET

This section details the Request and Response body for the supported GET HTTP operation to the Domain resource.

3.1.5.4.1.1 Request Body

None.

3.1.5.4.1.2 Response Body

The response MUST contain an **AutodiscoverResponse** element that contains a **Domain** element. These elements are defined in sections 2.2.4.1 and 2.2.4.2.

3.1.5.5 OAuth

The OAuth resource exposes topology information of a user's home server. The client MUST discover the OAuth URL by parsing the href of the **Link** element in the GET response from the Root resource.

If the request to the OAuth resource does not contain an Authorization header as specified in section 2.2.2.1, the server MUST respond with a 401 Unauthorized response.

If the request to the OAuth resource contains an invalid Authorization header as specified in section 2.2.2.1, the server MUST respond with a 403 Forbidden response.

If a valid Authorization header is provided and the user's home server information is unknown, the server MUST respond with a 404 response code and an empty body.

If a valid Authorization header is provided and the user's home server information exists on a separate server, the server MUST respond with a 200 response code and a **User** element. The **User** element MUST contain only one link with a "Redirect" token. Semantics of the "Redirect" Token are in section 2.2.5.5.

If a valid Authorization header is provided and the user's home server information exists on the current server, the server MUST respond with a 200 response code and a **User** element in the body. The **User** element MAY contain any of the following links depending on what is configured in the topology.

1. Internal/Autodiscover

2. External/Autodiscover

3. Internal/AuthBroker

4. External/AuthBroker

5. Internal/Ucwa

6. External/Ucwa

The **SipAccess** types MAY also be present in the response depending on what is configured in the topology

3.1.5.5.1 GET

This section details the Request and Response body for the supported GET HTTP operation to the OAuth resource.

3.1.5.5.1.1 Request Body

None.

3.1.5.5.1.2 Response Body

If the response code is 200 OK, then the response MUST contain an **AutodiscoverResponse** element that contains a **User** element. These elements are defined in sections 2.2.4.1 and 2.2.4.6.

If the response code is 401 Unauthorized, then there will be an HTML formatted body.

If the response code is 404 Not Found, then there will be an empty body.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

This section specifies details that pertain to protocol client behavior.

3.2.1 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.

3.2.1.1 Discovery of Autodiscover

The client will usually not have the initial Autodiscover service URL to send the first HTTP request and initiate the protocol flow. The client MUST construct and send requests to as many as four URLs to discover the initial Autodiscover service. The URLs constructed are based off of the client's Session Initiation Protocol (SIP) domain and MUST contain the following:

1. http://lyncdiscoverinternal.<sipdomain>

2. https://lyncdiscoverinternal.<sipdomain>

3. http://lyncdiscover.<sipdomain>

4. https://lyncdiscover.<sipdomain>

The client MAY execute requests 1 and 2 in parallel, and MAY execute requests 3 and 4 in parallel. The client MUST wait until getting a response from both request 1 and 2 before sending a request to 3 or 4.

The client MUST process the result of the first successful request and MAY cancel any pending requests after the receiving a successful response. The body of the response to any of the previously listed requests MUST contain an **AutodiscoverResponse** element which contains a **Root** element.

3.2.1.2 Redirect Detection

This protocol will result in many redirects. To prevent any redirect loops, the client MUST not allow more than 10 redirects in the entirety of the flow, or the client MUST implement redirect loop detection.

3.2.1.3 Internal External Network Switching

The final response of the OAuth, User, and Domain resources will contain an **AutodiscoverResponse** element with the **AccessLocation** attribute present, as well as internal and external links of the web services. The internal and external links are both provided in the same response to assist the client in switching between internal and external web services for times when the client is transition from inside to outside the network or vice versa. If the AccessLocation has the value "internal", the client SHOULD use the internal links. If the AccessLocation has the value "external", the client SHOULD use the external links.

The client also has the ability to probe the Autodiscover service to determine whether it is internal or external without executing the full protocol. After the client obtains the internal and external Autodiscover links, the client MAY cache them and probe the Root resource and determine its network location based on the AccessLocation returned in the response. If the client loses network connectivity for a brief period of time, the client MAY send a single GET request to the internal and external Autodiscover sites and then connect to the corresponding URL stored in the cache based on the returned AccessLocation.

The client SHOULD cache the returned web service URLs until the client receives an error connecting to one of the web services.

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

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Discover Home Server

In this scenario, the client follows this protocol to discover its home server URLs.

The client constructs and sends an HTTP and HTTPS request to the Autodiscover service based on its Session Initiation Protocol (SIP) URI, john@contoso.com.

```
GET http://lyncdiscover.contoso.com/?sipuri=sip:john@contoso.com HTTP/1.1
Host: contoso.com
Accept: application/vnd.microsoft.rtc.autodiscover+xml;v=1

GET https://lyncdiscoverinternal.contoso.com HTTP/1.1
Host: contoso.com
Accept: application/vnd.microsoft.rtc.autodiscover+xml;v=1
```

The HTTPS request succeeds first with the following result.

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/vnd.microsoft.rtc.autodiscover+xml;v=1

<?xml version="1.0" encoding="utf-8"?>
<AutodiscoverResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" AccessLocation="Internal">
  <Root>
    <Link href="https://contoso.com/Autodiscover/AutodiscoverService.svc/root/domain"
token="Domain"/>
    <Link href="https://contoso.com/Autodiscover/AutodiscoverService.svc/root/user"
token="User"/>
    <Link href="https://contoso.com/Autodiscover/AutodiscoverService.svc/root/oauth/user"
token="OAuth"/>
  </Root>
</AutodiscoverResponse>
```

The client parses the returned links and sends an HTTP GET to the href of the link with the "OAuth" token. The client provides the JSON token in the Authorization header.

```
GET https://contoso.com/autodiscover/autodiscoverservice.svc/root/oauth/user HTTP/1.1
Host: contoso.com
Authorization: Bearer Y4LBFDjzqXHPxYMHiJ9snKNWiw0lyShf+i/GU9B+scVnYB3T5BDp6w==
Accept: application/vnd.microsoft.rtc.autodiscover+xml;v=1
```

The server authenticates the client and returns a Redirect response indicating the home server location of the client.

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/vnd.microsoft.rtc.autodiscover+xml;v=1

<?xml version="1.0" encoding="utf-8"?>
<AutodiscoverResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" AccessLocation="Internal">
```

```
  <User>
    <Link href="https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root"
token="Redirect"/>
  </User>
</AutodiscoverResponse>
```

The client sends an HTTPS request to redirect the link.

```
GET https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root HTTP/1.1
Host: pool1.contoso.com
Accept: application/vnd.microsoft.rtc.autodiscover+xml;v=1
```

The HTTPS request succeeds with the following result.

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/vnd.microsoft.rtc.autodiscover+xml;v=1

<?xml version="1.0" encoding="utf-8"?>
<AutodiscoverResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" AccessLocation="Internal">
  <Root>
    <Link href="https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root/domain"
token="Domain"/>
    <Link href="https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root/user"
token="User"/>
    <Link
href="https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root/oauth/user"
token="OAuth"/>
  </Root>
</AutodiscoverResponse>
```

The client sends an HTTPS request to OAuth link with Authentication header.

```
GET https://pool1.contoso.com/autodiscover/autodiscoverservice.svc/root/oauth/user HTTP/1.1
Host: contoso.com
Authorization: Bearer Y4LBFDjzqXHPxYMHiJ9snKNWiw0lyShf+i/GU9B+scVnYB3T5BDp6w==
Accept: application/vnd.microsoft.rtc.autodiscover+xml;v=1
```

The HTTPS request succeeds with the following result.

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/vnd.microsoft.rtc.autodiscover+xml;v=1

<?xml version="1.0" encoding="utf-8"?>
<AutodiscoverResponse AccessLocation="Internal" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <User>
```

```
    <Link token="Internal/Autodiscover"
  href="https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root"/>
    <Link token="Internal/AuthBroker" href="https://pool1.contoso.com/Reach/sip.svc"/>
    <Link token="Internal/Ucwa" href="https://pool1.contoso.com/Ucwa/discovery"/>
    <Link token="External/Autodiscover"
  href="https://pool1external.contoso.com/Autodiscover/AutodiscoverService.svc/root"/>
    <Link token="External/AuthBroker"
  href="https://pool1external.contoso.com/Reach/sip.svc"/>
    <Link token="External/Ucwa" href="https://pool1external.contoso.com/Ucwa/discovery"/>

  </User>
</AutodiscoverResponse>
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full XML Schema

For ease of implementation, the following is the full XML schema for this protocol.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="AutodiscoverResponse" nillable="true" type="AutodiscoverResponse" />
  <xs:complexType name="AutodiscoverResponse">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Root" type="Root" />
      <xs:element minOccurs="0" maxOccurs="1" name="User" type="User" />
      <xs:element minOccurs="0" maxOccurs="1" name="Domain" type="Domain" />
    </xs:sequence>
    <xs:attribute name="AccessLocation" type="xs:string" />
  </xs:complexType>
  <xs:complexType name="Root">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Link" type="Link" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Link">
    <xs:attribute name="token" type="xs:string" />
    <xs:attribute name="href" type="xs:string" />
  </xs:complexType>
  <xs:complexType name="User">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="SipServerInternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="SipClientInternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="SipServerExternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="SipClientExternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Link" type="Link" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SipAccess">
    <xs:attribute name="fqdn" type="xs:string" />
    <xs:attribute name="port" type="xs:string" />
  </xs:complexType>
  <xs:complexType name="Domain">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="SipServerInternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="SipClientInternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="SipServerExternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="SipClientExternalAccess" type="SipAccess"
/>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Link" type="Link" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

7 Appendix B: Full JSON Schema

For ease of implementation, the following is the full JSON schema for this protocol.

```
{
  "id": "AutodiscoverResponse",
  "type": "object",
  "properties": {
    "AccessLocation": {
      "required": true,
      "type": [
        "string",
        "null"
      ]
    },
    "Root": {
      "id": "Root",
      "required": true,
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "Links": {
          "id": "Link[]",
          "required": true,
          "type": [
            "array",
            "null"
          ],
          "items": {
            "id": "Link",
            "type": [
              "object",
              "null"
            ],
            "properties": {
              "token": {
                "required": true,
                "type": [
                  "string",
                  "null"
                ]
              },
              "href": {
                "required": true,
                "type": [
                  "string",
                  "null"
                ]
              }
            }
          }
        }
      }
    },
    "User": {
```

```
    "id": "User",
    "required": true,
    "type": [
      "object",
      "null"
    ],
    "properties": {
      "SipServerInternalAccess": {
        "id": "SipAccess",
        "required": true,
        "type": [
          "object",
          "null"
        ],
        "properties": {
          "fqdn": {
            "required": true,
            "type": [
              "string",
              "null"
            ]
          },
          "port": {
            "required": true,
            "type": [
              "string",
              "null"
            ]
          }
        }
      },
      "SipClientInternalAccess": {
        "$ref": "SipAccess"
      },
      "SipServerExternalAccess": {
        "$ref": "SipAccess"
      },
      "SipClientExternalAccess": {
        "$ref": "SipAccess"
      },
      "Links": {
        "$ref": "Link[]"
      }
    }
  },
  "Domain": {
    "id": "Domain",
    "required": true,
    "type": [
      "object",
      "null"
    ],
    "properties": {
      "SipServerInternalAccess": {
        "$ref": "SipAccess"
      },
      "SipClientInternalAccess": {
        "$ref": "SipAccess"
      },
```

```
          "SipServerExternalAccess": {
            "$ref": "SipAccess"
          },
          "SipClientExternalAccess": {
            "$ref": "SipAccess"
          },
          "Links": {
            "$ref": "Link[]"
          }
        }
      }
    }
  }
```

8 Appendix C: 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 Lync Server 2013
- Microsoft Lync 2013

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.

9 Change Tracking

This section identifies changes that were made to the [MS-OCDISCWS] protocol document between the February 2013 and July 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

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 language and 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 or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

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.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.4.1 AutodiscoverResponse	Updated xml schema.	N	Content updated.
2.2.4.2 Domain	Updated xml schema.	N	Content updated.
2.2.4.4 Root	Updated xml schema.	N	Content updated.
2.2.4.6 User	Updated xml schema.	N	Content updated.
4.1 Discover Home Server	Updated the example.	N	Content updated.

10 Index