[MS-OXPROTO]:

Exchange Server Protocols System Overview

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.
- **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.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability.
4/25/2008	0.2	Minor	Revised and updated property names and other technical content.
6/27/2008	1.0	Major	Initial Release.
8/6/2008	1.01	Minor	Updated references to reflect date of initial release.
9/3/2008	1.02	Minor	Changed title of document.
12/3/2008	1.03	Minor	Revised and edited technical content.
2/4/2009	1.04	Minor	Revised and edited technical content.
3/4/2009	1.05	Minor	Revised and edited technical content.
4/10/2009	2.0	Major	Updated technical content for new product releases.
7/15/2009	3.0	Major	Revised and edited for technical content.
11/4/2009	4.0.0	Major	Updated and revised the technical content.
2/10/2010	4.0.0	None	Version 4.0.0 release
5/5/2010	5.0.0	Major	Updated and revised the technical content.
8/4/2010	5.1	Minor	Clarified the meaning of the technical content.
11/3/2010	6.0	Major	Significantly changed the technical content.
3/18/2011	7.0	Major	Significantly changed the technical content.
8/5/2011	8.0	Major	Significantly changed the technical content.
10/7/2011	9.0	Major	Significantly changed the technical content.
1/20/2012	10.0	Major	Significantly changed the technical content.
4/27/2012	10.1	Minor	Clarified the meaning of the technical content.
7/16/2012	10.2	Minor	Clarified the meaning of the technical content.
10/8/2012	11.0	Major	Significantly changed the technical content.
2/11/2013	12.0	Major	Significantly changed the technical content.
7/26/2013	12.1	Minor	Clarified the meaning of the technical content.
11/18/2013	13.0	Major	Significantly changed the technical content.
2/10/2014	13.0	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	13.0	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	13.0	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
10/30/2014	13.1	Minor	Clarified the meaning of the technical content.
3/30/2015	13.1	None	No changes to the meaning, language, or formatting of the technical content.
5/26/2015	14.0	Major	Significantly changed the technical content.
9/14/2015	14.1	Minor	Clarified the meaning of the technical content.
6/13/2016	14.2	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

Microsoft Exchange Server provides a rich set of interfaces with which messaging clients can interoperate. A messaging client can connect to a computer that is running Microsoft Exchange by using one or more of the available protocols and perform tasks by issuing requests to the server and processing server responses.

The technical requirements, limitations, dependencies, and Microsoft-specific protocol behavior of the protocols that are used in Microsoft Exchange are described by the Microsoft Exchange protocol documentation set. The documentation set can be broken down into functional groups as illustrated in the following diagram.

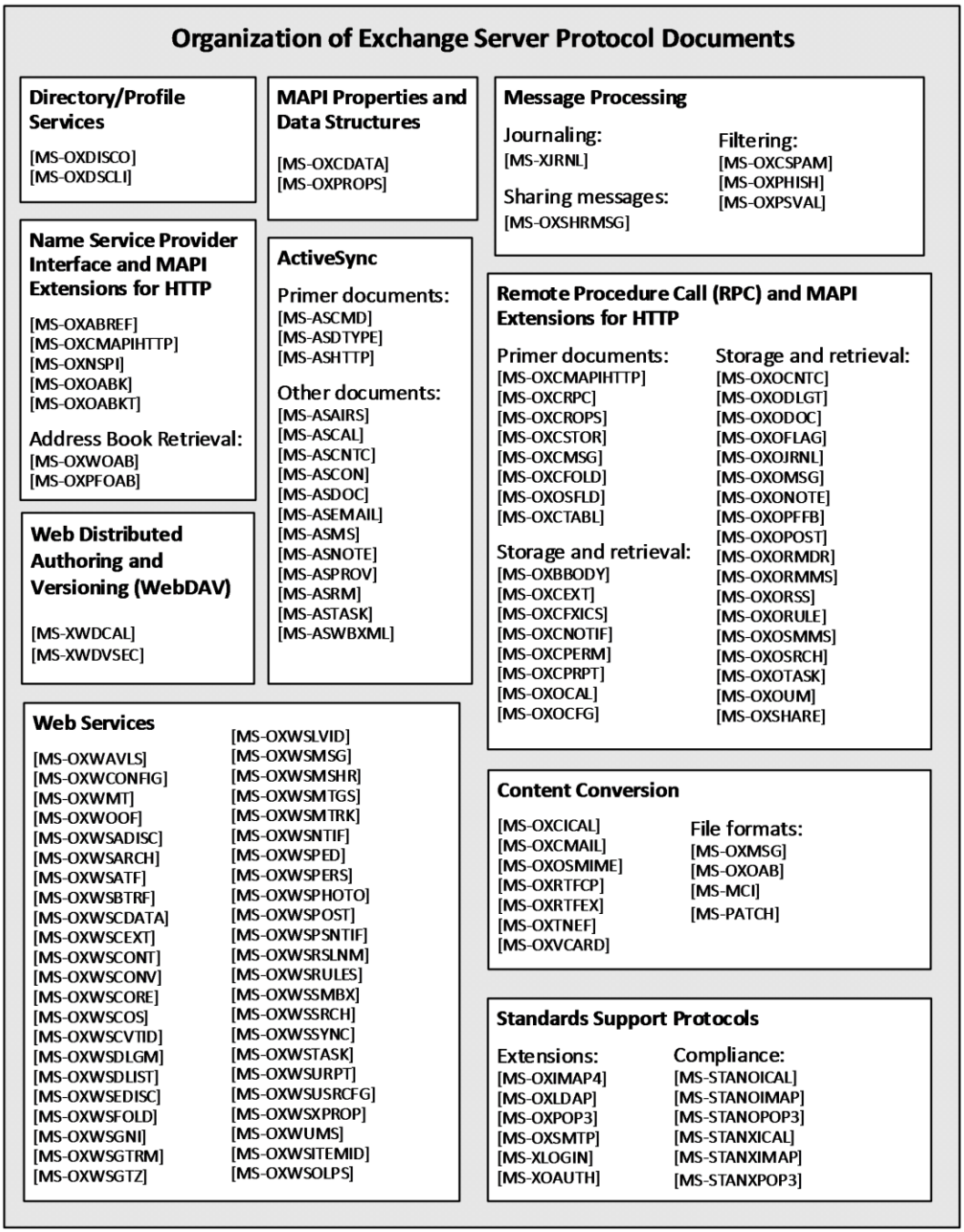

Figure 1: Organization of Microsoft Exchange protocol documents

For information about the protocols described by each of these documents, see section 2.2.

1.1 Glossary

This document uses the following terms:

address book: A collection of **Address Book objects**, each of which are contained in any number of address lists.

Address Book object: An entity in an **address book** that contains a set of attributes, each attribute with a set of associated values.

ambiguous name resolution (ANR): A search algorithm that permits a client to search multiple naming-related attributes (2) on objects by way of a single clause of the form "(anr=value)" in a **Lightweight Directory Access Protocol (LDAP)** search filter. This permits a client to query for an object when the client possesses some identifying material related to the object but does not know which attribute of the object contains that identifying material.

Appointment object: A **Calendar object** that has an organizer but no attendees.

Attachment object: A set of properties that represents a file, **Message object**, or structured storage that is attached to a Message object and is visible through the attachments table for a Message object.

Autodiscover client: A client that queries for a set of server locations where setup and configuration information for an [RFC2821]-compliant email address is stored.

Autodiscover server: A server in a managed environment that makes setup and configuration information available to **Autodiscover clients**. The location of Autodiscover servers is made available via the Autodiscover HTTP Service Protocol, as described in [MS-OXDISCO].

base64 encoding: A binary-to-text encoding scheme whereby an arbitrary sequence of bytes is converted to a sequence of printable ASCII characters, as described in [RFC4648].

Calendar folder: A Folder object that contains **Calendar objects**.

Calendar object: A **Message object** that represents an event, which can be a one-time event or a recurring event. The Calendar object includes properties that specify event details such as description, organizer, date and time, and status.

class: User-defined binary data that is associated with a key.

contact: (1) An object of the contact class that represents a company or person whom a user can contact.

(2) A person, company, or other entity that is stored in a directory and is associated with one or more unique identifiers and attributes (2), such as an Internet message address or login name.

contents table: A Table object whose rows represent the **Message objects** that are contained in a Folder object.

delegate: A user or resource that has permissions to act on behalf of another user or resource.

delegate access: The access that is granted by a delegator to a delegate and is used by the delegate to access the delegator's account.

delegator: A user or resource for which another user or resource has permission to act on its behalf.

directory service (DS): A service that stores and organizes information about a computer network's users and network shares, and that allows network administrators to manage users' access to the shares. See also Active Directory.

distinguished name (DN): In the Active Directory directory service, the unique identifier of an object in Active Directory, as described in [MS-ADTS] and [RFC2251].

Domain Name System (DNS): A hierarchical, distributed database that contains mappings of domain names (1) to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

Drafts folder: A **special folder** that is the default location for **Message objects** that have been saved but not sent.

event: Any significant occurrence in a system or an application that requires users to be notified or an entry to be added to a log.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

handle: Any token that can be used to identify and access an object such as a device, file, or a window.

Hypertext Markup Language (HTML): An application of the Standard Generalized Markup Language (SGML) that uses tags to mark elements in a document, as described in [HTML].

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

ICS state: A set of properties that determine the state of a local replica narrowed down to a specific synchronization scope.

Inbox folder: A **special folder** that is the default location for **Message objects** received by a user or resource.

Incremental Change Synchronization (ICS): A data format and algorithm that is used to synchronize folders and messages between two sources.

instant messaging: A method of real-time communication over the Internet in which a sender types a message to one or more recipients and the recipient immediately receives the message in a pop-up window.

Internet Message Access Protocol - Version 4 (IMAP4): A protocol that is used for accessing email and news items from mail servers, as described in [RFC3501].

Lightweight Directory Access Protocol (LDAP): The primary access protocol for Active Directory. Lightweight Directory Access Protocol (LDAP) is an industry-standard protocol, established by the Internet Engineering Task Force (IETF), which allows users to query and update information in a **directory service (DS)**, as described in [MS-ADTS]. The Lightweight Directory Access Protocol can be either version 2 [RFC1777] or version 3 [RFC3377].

mail add-in: An Office Add-in that enhances an email or appointment item.

mailbox: A **message store** that contains email, calendar items, and other **Message objects** for a single recipient.

Meeting object: A **Calendar object** that has both an organizer and attendees.

message body: (1) The content within an HTTP message, as described in [RFC2616] section 4.3.

(2) The main message text of an email message. A few properties of a **Message object** represent its message body, with one property containing the text itself and others defining its code page and its relationship to alternative body formats.

Message object: A set of properties that represents an email message, appointment, contact, or other type of personal-information-management object. In addition to its own properties, a Message object contains recipient properties that represent the addressees to which it is addressed, and an attachments table that represents any files and other Message objects that are attached to it.

message store: A unit of containment for a single hierarchy of Folder objects, such as a mailbox or public folders.

Multimedia Messaging Service (MMS): A communications protocol that is designed for messages containing text, images, and other multimedia content that is sent between mobile phones.

Multipurpose Internet Mail Extensions (MIME): A set of extensions that redefines and expands support for various types of content in email messages, as described in [RFC2045], [RFC2046], and [RFC2047].

NT LAN Manager (NTLM) Authentication Protocol: A protocol using a challenge-response mechanism for authentication (2) in which clients are able to verify their identities without sending a password to the server. It consists of three messages, commonly referred to as Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication). For more information, see [MS-NLMP].

offline address book (OAB): A collection of address lists that are stored in a format that a client can save and use locally.

Post object: A **Message object** that represents an entry in a discussion thread stored in a messaging store.

Post Office Protocol - Version 3 (POP3): A protocol that is used for accessing email from mail servers, as described in [RFC1939].

public folder: A Folder object that is stored in a location that is publicly available.

remote operation (ROP): An operation that is invoked against a server. Each ROP represents an action, such as delete, send, or query. A ROP is contained in a ROP buffer for transmission over the wire.

remote procedure call (RPC): A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

restriction: A filter used to map some domain into a subset of itself, by passing only those items from the domain that match the filter. Restrictions can be used to filter existing Table objects or to define new ones, such as search folder (2) or rule criteria.

retention policy: A policy that specifies the length of time during which data, documents, and other records must be available for recovery.

Rich Text Format (RTF): Text with formatting as described in [MSFT-RTF].

rights-managed email message: An email message that specifies permissions that are designed to protect its content from inappropriate access, use, and distribution.

S/MIME (Secure/Multipurpose Internet Mail Extensions): A set of cryptographic security services, as described in [RFC5751].

security descriptor: A data structure containing the security information associated with a securable object. A **security descriptor** identifies an object's owner by its security identifier (SID). If access control is configured for the object, its **security descriptor** contains a discretionary access control list (DACL) with SIDs for the security principals who are allowed or denied access. Applications use this structure to set and query an object's security status. The **security descriptor** is used to guard access to an object as well as to control which type of auditing takes place when the object is accessed. The **security descriptor** format is specified in [MS-DTYP] section 2.4.6; a string representation of **security descriptors**, called SDDL, is specified in [MS-DTYP] section 2.5.1.

service connection point: An object that is made available by a directory service and that clients can use to discover **Autodiscover servers**.

Short Message Service (SMS): A communications protocol that is designed for sending text messages between mobile phones.

Simple Mail Transfer Protocol (SMTP): A member of the TCP/IP suite of protocols that is used to transport Internet messages, as described in [RFC5321].

site mailbox: A repository comprised of a mailbox and a web-based collaboration environment that is presented to users as a mailbox in an email client. A site mailbox uses team membership to determine which users have access to the repository.

special folder: One of a default set of Folder objects that can be used by an implementation to store and retrieve user data objects.

Unified Messaging: A set of components and services that enable voice, fax, and email messages to be stored in a user's **mailbox** and accessed from a variety of devices.

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].

vCard: A format for storing and exchanging electronic business cards, as described in [RFC2426].

Web Distributed Authoring and Versioning Protocol (WebDAV): The Web Distributed Authoring and Versioning Protocol, as described in [RFC2518] or [RFC4918].

web service: A unit of application logic that provides data and services to other applications and can be called by using standard Internet transport protocols such as **HTTP**, **Simple Mail Transfer Protocol (SMTP)**, or File Transfer Protocol (FTP). Web services can perform functions that range from simple requests to complicated business processes.

Wireless Application Protocol (WAP) Binary XML (WBXML): A compact binary representation of **XML** that is designed to reduce the transmission size of XML documents over narrowband communication channels.

WSDL message: An abstract, typed definition of the data that is communicated during a WSDL operation [WSDL]. Also, an element that describes the data being exchanged between web service providers and clients.

XML: The Extensible Markup Language, as described in [XML1.0].

XML schema definition (XSD): The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining

the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring XML schemas.

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.

[MS-ASAIRS] Microsoft Corporation, "Exchange ActiveSync: AirSyncBase Namespace Protocol".

[MS-ASCAL] Microsoft Corporation, "Exchange ActiveSync: Calendar Class Protocol".

[MS-ASCMD] Microsoft Corporation, "Exchange ActiveSync: Command Reference Protocol".

[MS-ASCNTC] Microsoft Corporation, "Exchange ActiveSync: Contact Class Protocol".

[MS-ASCON] Microsoft Corporation, "Exchange ActiveSync: Conversations Protocol".

[MS-ASDOC] Microsoft Corporation, "Exchange ActiveSync: Document Class Protocol".

[MS-ASDTYPE] Microsoft Corporation, "Exchange ActiveSync: Data Types".

[MS-ASEMAIL] Microsoft Corporation, "Exchange ActiveSync: Email Class Protocol".

[MS-ASHTTP] Microsoft Corporation, "Exchange ActiveSync: HTTP Protocol".

[MS-ASMS] Microsoft Corporation, "Exchange ActiveSync: Short Message Service (SMS) Protocol".

[MS-ASNOTE] Microsoft Corporation, "Exchange ActiveSync: Notes Class Protocol".

[MS-ASPROV] Microsoft Corporation, "Exchange ActiveSync: Provisioning Protocol".

[MS-ASRM] Microsoft Corporation, "Exchange ActiveSync: Rights Management Protocol".

[MS-ASTASK] Microsoft Corporation, "Exchange ActiveSync: Tasks Class Protocol".

[MS-ASWBXML] Microsoft Corporation, "Exchange ActiveSync: WAP Binary XML (WBXML) Algorithm".

[MS-MCI] Microsoft Corporation, "Microsoft ZIP (MSZIP) Compression and Decompression Data Structure".

[MS-NSPI] Microsoft Corporation, "Name Service Provider Interface (NSPI) Protocol".

[MS-OXABREF] Microsoft Corporation, "Address Book Name Service Provider Interface (NSPI) Referral Protocol".

[MS-OXBBODY] Microsoft Corporation, "Best Body Retrieval Algorithm".

[MS-OXCDATA] Microsoft Corporation, "Data Structures".

[MS-OXCEXT] Microsoft Corporation, "Client Extension Message Object Protocol".

[MS-OXCFOLD] Microsoft Corporation, "Folder Object Protocol".

[MS-OXCFXICS] Microsoft Corporation, "Bulk Data Transfer Protocol".

[MS-OXCICAL] Microsoft Corporation, "iCalendar to Appointment Object Conversion Algorithm".

[MS-OXCMAIL] Microsoft Corporation, "RFC 2822 and MIME to Email Object Conversion Algorithm".

[MS-OXCMAPIHTTP] Microsoft Corporation, "Messaging Application Programming Interface (MAPI) Extensions for HTTP".

[MS-OXCMSG] Microsoft Corporation, "Message and Attachment Object Protocol".

[MS-OXCNOTIF] Microsoft Corporation, "Core Notifications Protocol".

[MS-OXCPERM] Microsoft Corporation, "Exchange Access and Operation Permissions Protocol".

[MS-OXCPRPT] Microsoft Corporation, "Property and Stream Object Protocol".

[MS-OXCROPS] Microsoft Corporation, "Remote Operations (ROP) List and Encoding Protocol".

[MS-OXCRPC] Microsoft Corporation, "Wire Format Protocol".

[MS-OXCSPAM] Microsoft Corporation, "Spam Confidence Level Protocol".

[MS-OXCSTOR] Microsoft Corporation, "Store Object Protocol".

[MS-OXCTABL] Microsoft Corporation, "Table Object Protocol".

[MS-OXDISCO] Microsoft Corporation, "Autodiscover HTTP Service Protocol".

[MS-OXDSCLI] Microsoft Corporation, "Autodiscover Publishing and Lookup Protocol".

[MS-OXIMAP4] Microsoft Corporation, "Internet Message Access Protocol Version 4 (IMAP4) Extensions".

[MS-OXLDAP] Microsoft Corporation, "Lightweight Directory Access Protocol (LDAP) Version 3 Extensions".

[MS-OXMSG] Microsoft Corporation, "Outlook Item (.msg) File Format".

[MS-OXNSPI] Microsoft Corporation, "Exchange Server Name Service Provider Interface (NSPI) Protocol".

[MS-OXOABKT] Microsoft Corporation, "Address Book User Interface Templates Protocol".

[MS-OXOABK] Microsoft Corporation, "Address Book Object Protocol".

[MS-OXOAB] Microsoft Corporation, "Offline Address Book (OAB) File Format and Schema".

[MS-OXOCAL] Microsoft Corporation, "Appointment and Meeting Object Protocol".

[MS-OXOCFG] Microsoft Corporation, "Configuration Information Protocol".

[MS-OXOCNTC] Microsoft Corporation, "Contact Object Protocol".

[MS-OXODLGT] Microsoft Corporation, "Delegate Access Configuration Protocol".

[MS-OXODOC] Microsoft Corporation, "Document Object Protocol".

[MS-OXOFLAG] Microsoft Corporation, "Informational Flagging Protocol".

[MS-OXOJRNL] Microsoft Corporation, "Journal Object Protocol".

[MS-OXOMSG] Microsoft Corporation, "Email Object Protocol".

[MS-OXONOTE] Microsoft Corporation, "Note Object Protocol".

[MS-OXOPFFB] Microsoft Corporation, "Public Folder-Based Free/Busy Protocol".

[MS-OXOPOST] Microsoft Corporation, "Post Object Protocol".

[MS-OXORMDR] Microsoft Corporation, "Reminder Settings Protocol".

[MS-OXORMMS] Microsoft Corporation, "Rights-Managed Email Object Protocol".

[MS-OXORSS] Microsoft Corporation, "RSS Object Protocol".

[MS-OXORULE] Microsoft Corporation, "Email Rules Protocol".

[MS-OXOSFLD] Microsoft Corporation, "Special Folders Protocol".

[MS-OXOSMIME] Microsoft Corporation, "S/MIME Email Object Algorithm".

[MS-OXOSMMS] Microsoft Corporation, "Short Message Service (SMS) and Multimedia Messaging Service (MMS) Object Protocol".

[MS-OXOSRCH] Microsoft Corporation, "Search Folder List Configuration Protocol".

[MS-OXOTASK] Microsoft Corporation, "Task-Related Objects Protocol".

[MS-OXOUM] Microsoft Corporation, "Voice Mail and Fax Objects Protocol".

[MS-OXPFOAB] Microsoft Corporation, "Offline Address Book (OAB) Public Folder Retrieval Protocol".

[MS-OXPHISH] Microsoft Corporation, "Phishing Warning Protocol".

[MS-OXPOP3] Microsoft Corporation, "Post Office Protocol Version 3 (POP3) Extensions".

[MS-OXPROPS] Microsoft Corporation, "Exchange Server Protocols Master Property List".

[MS-OXPSVAL] Microsoft Corporation, "Email Postmark Validation Algorithm".

[MS-OXRTFCP] Microsoft Corporation, "Rich Text Format (RTF) Compression Algorithm".

[MS-OXRTFEX] Microsoft Corporation, "Rich Text Format (RTF) Extensions Algorithm".

[MS-OXSHARE] Microsoft Corporation, "Sharing Message Object Protocol".

[MS-OXSHRMSG] Microsoft Corporation, "Sharing Message Attachment Schema".

[MS-OXSMTP] Microsoft Corporation, "Simple Mail Transfer Protocol (SMTP) Extensions".

[MS-OXTNEF] Microsoft Corporation, "Transport Neutral Encapsulation Format (TNEF) Data Algorithm".

[MS-OXVCARD] Microsoft Corporation, "vCard to Contact Object Conversion Algorithm".

[MS-OXWAVLS] Microsoft Corporation, "Availability Web Service Protocol".

[MS-OXWCONFIG] Microsoft Corporation, "Web Service Configuration Protocol".

[MS-OXWMT] Microsoft Corporation, "Mail Tips Web Service Extensions".

[MS-OXWOAB] Microsoft Corporation, "Offline Address Book (OAB) Retrieval File Format".

[MS-OXWOOF] Microsoft Corporation, "Out of Office (OOF) Web Service Protocol".

[MS-OXWSADISC] Microsoft Corporation, "Autodiscover Publishing and Lookup SOAP-Based Web Service Protocol".

[MS-OXWSARCH] Microsoft Corporation, "Archiving Web Service Protocol".

[MS-OXWSATT] Microsoft Corporation, "Attachment Handling Web Service Protocol".

[MS-OXWSBTRF] Microsoft Corporation, "Bulk Transfer Web Service Protocol".

[MS-OXWSCDATA] Microsoft Corporation, "Common Web Service Data Types".

[MS-OXWSCEXT] Microsoft Corporation, "Client Extension Web Service Protocol".

[MS-OXWSCONT] Microsoft Corporation, "Contacts Web Service Protocol".

[MS-OXWSCONV] Microsoft Corporation, "Conversations Web Service Protocol".

[MS-OXWSCORE] Microsoft Corporation, "Core Items Web Service Protocol".

[MS-OXWSCOS] Microsoft Corporation, "Unified Contact Store Web Service Protocol".

[MS-OXWSCVTID] Microsoft Corporation, "Convert Item Identifier Web Service Protocol".

[MS-OXWSDLGM] Microsoft Corporation, "Delegate Access Management Web Service Protocol".

[MS-OXWSDLIST] Microsoft Corporation, "Distribution List Creation and Usage Web Service Protocol".

[MS-OXWSEDISC] Microsoft Corporation, "Electronic Discovery (eDiscovery) Web Service Protocol".

[MS-OXWSFOLD] Microsoft Corporation, "Folders and Folder Permissions Web Service Protocol".

[MS-OXWSGNI] Microsoft Corporation, "Nonindexable Item Web Service Protocol".

[MS-OXWSGTRM] Microsoft Corporation, "Get Rooms List Web Service Protocol".

[MS-OXWSGTZ] Microsoft Corporation, "Get Server Time Zone Web Service Protocol".

[MS-OXWSITEMID] Microsoft Corporation, "Web Service Item ID Algorithm".

[MS-OXWSLVID] Microsoft Corporation, "Federated Internet Authentication Web Service Protocol".

[MS-OXWSMSG] Microsoft Corporation, "Email Message Types Web Service Protocol".

[MS-OXWSMSHR] Microsoft Corporation, "Folder Sharing Web Service Protocol".

[MS-OXWSMTGS] Microsoft Corporation, "Calendaring Web Service Protocol".

[MS-OXWSMTRK] Microsoft Corporation, "Message Tracking Web Service Protocol".

[MS-OXWSNTIF] Microsoft Corporation, "Notifications Web Service Protocol".

[MS-OXWSOLPS] Microsoft Corporation, "Online Personal Search Web Service Protocol".

[MS-OXWSPED] Microsoft Corporation, "Password Expiration Date Web Service Protocol".

[MS-OXWSPERS] Microsoft Corporation, "Persona Web Service Protocol".

[MS-OXWSPHOTO] Microsoft Corporation, "Photo Web Service Protocol".

[MS-OXWSPOST] Microsoft Corporation, "Post Items Web Service Protocol".

[MS-OXWSPSNTIF] Microsoft Corporation, "Push Notifications Web Service Protocol".

[MS-OXWSRSLNM] Microsoft Corporation, "Resolve Recipient Names Web Service Protocol".

[MS-OXWSRULES] Microsoft Corporation, "Inbox Rules Web Service Protocol".

[MS-OXWSSMBX] Microsoft Corporation, "Site Mailbox Web Service Protocol".

[MS-OXWSSRCH] Microsoft Corporation, "Mailbox Search Web Service Protocol".

[MS-OXWSSYNC] Microsoft Corporation, "Mailbox Contents Synchronization Web Service Protocol".

[MS-OXWSTASK] Microsoft Corporation, "Tasks Web Service Protocol".

[MS-OXWSURPT] Microsoft Corporation, "Retention Tag Web Service Protocol".

[MS-OXWSUSRCFG] Microsoft Corporation, "User Configuration Web Service Protocol".

[MS-OXWSXPROP] Microsoft Corporation, "Extended Properties Structure".

[MS-OXWUMS] Microsoft Corporation, "Voice Mail Settings Web Service Protocol".

[MS-PATCH] Microsoft Corporation, "LZX DELTA Compression and Decompression".

[MS-SMTPNTLM] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication: Simple Mail Transfer Protocol (SMTP) Extension".

[MS-STANOICAL] Microsoft Corporation, "Outlook iCalendar Standards Support Version 2".

[MS-STANOIMAP] Microsoft Corporation, "Outlook Internet Message Access Protocol (IMAP) Standards Support".

[MS-STANOPOP3] Microsoft Corporation, "Outlook Post Office Protocol Version 3 (POP3) Standards Support".

[MS-STANXICAL] Microsoft Corporation, "Exchange iCalendar Standards Support Version 2".

[MS-STANXIMAP] Microsoft Corporation, "Exchange Internet Message Access Protocol (IMAP) Standards Support".

[MS-STANXPOP3] Microsoft Corporation, "Exchange Post Office Protocol Version 3 (POP3) Standards Support".

[MS-WDVSE] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Protocol: Server Extensions".

[MS-XJRNL] Microsoft Corporation, "Journal Record Message File Format".

[MS-XLOGIN] Microsoft Corporation, "Simple Mail Transfer Protocol (SMTP) AUTH LOGIN Extension".

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

[MS-XWDCAL] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Extensions for Calendar Support".

[MS-XWDVSEC] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Protocol Security Descriptor Extensions".

[MSFT-RTF] Microsoft Corporation, "Rich Text Format (RTF) Specification", version 1.9.1, March 2008, http://www.microsoft.com/en-us/download/details.aspx?id=10725

[MSFT-SAP] Microsoft Corporation, "Security and Protection", http://technet.microsoft.com/en-us/library/aa996775(EXCHG.80).aspx

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987, http://www.ietf.org/rfc/rfc1034.txt

[RFC1823] Howes, T., and Smith, M., "The LDAP Application Program Interface", RFC 1823, August 1995, http://www.rfc-editor.org/rfc/rfc1823.txt

[RFC1939] Myers, J., and Rose, M., "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996, http://www.rfc-editor.org/rfc/rfc1939.txt

[RFC2045] Freed, N., and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, http://www.rfc-editor.org/rfc/rfc2045.txt

[RFC2046] Freed, N., and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996, http://www.rfc-editor.org/rfc/rfc2046.txt

[RFC2068] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997, http://www.ietf.org/rfc/rfc2068.txt

[RFC2445] Dawson, F., and Stenerson, D., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 2445, November 1998, http://www.rfc-editor.org/rfc/rfc2445.txt

[RFC2446] Silverberg, S., Mansour, S., Dawson, F., and Hopson, R., "iCalendar Transport-Independent Interoperability Protocol (iTIP) Scheduling Events, BusyTime, To-Dos, and Journal Entries", RFC 2446, November 1998, http://www.ietf.org/rfc/rfc2446.txt

[RFC2447] Dawson, F., Mansour, S., and Silverberg, S., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 2447, November 1998, http://www.rfc-editor.org/rfc/rfc2447.txt

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., et al., "HTTP Extensions for Distributed Authoring - WebDAV", RFC 2518, February 1999, http://www.ietf.org/rfc/rfc2518.txt

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

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001, http://www.ietf.org/rfc/rfc2821.txt

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt

[RFC2849] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000, http://www.ietf.org/rfc/rfc2849.txt

[RFC4315] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, December 2005, http://www.rfc-editor.org/rfc/rfc4315.txt

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[WSIBASIC] Ballinger, K., Ehnebuske, D., Gudgin, M., et al., Eds., "Basic Profile Version 1.0", Final Material, April 2004, http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

2 Functional Architecture

The Microsoft Exchange Server system consists of protocols (including extensions to industry-standard or other published protocols) that Microsoft Exchange uses to communicate with other products. The protocols enable the transfer of data between client and server and enable clients to access, interpret, and manipulate data in the **message store**.

2.1 Overview

The Microsoft Exchange system from a protocols perspective, where the server provides protocols for clients, is illustrated in the following figure. The clients that interoperate with the server perform messaging tasks, and ancillary entities provide essential supporting services.

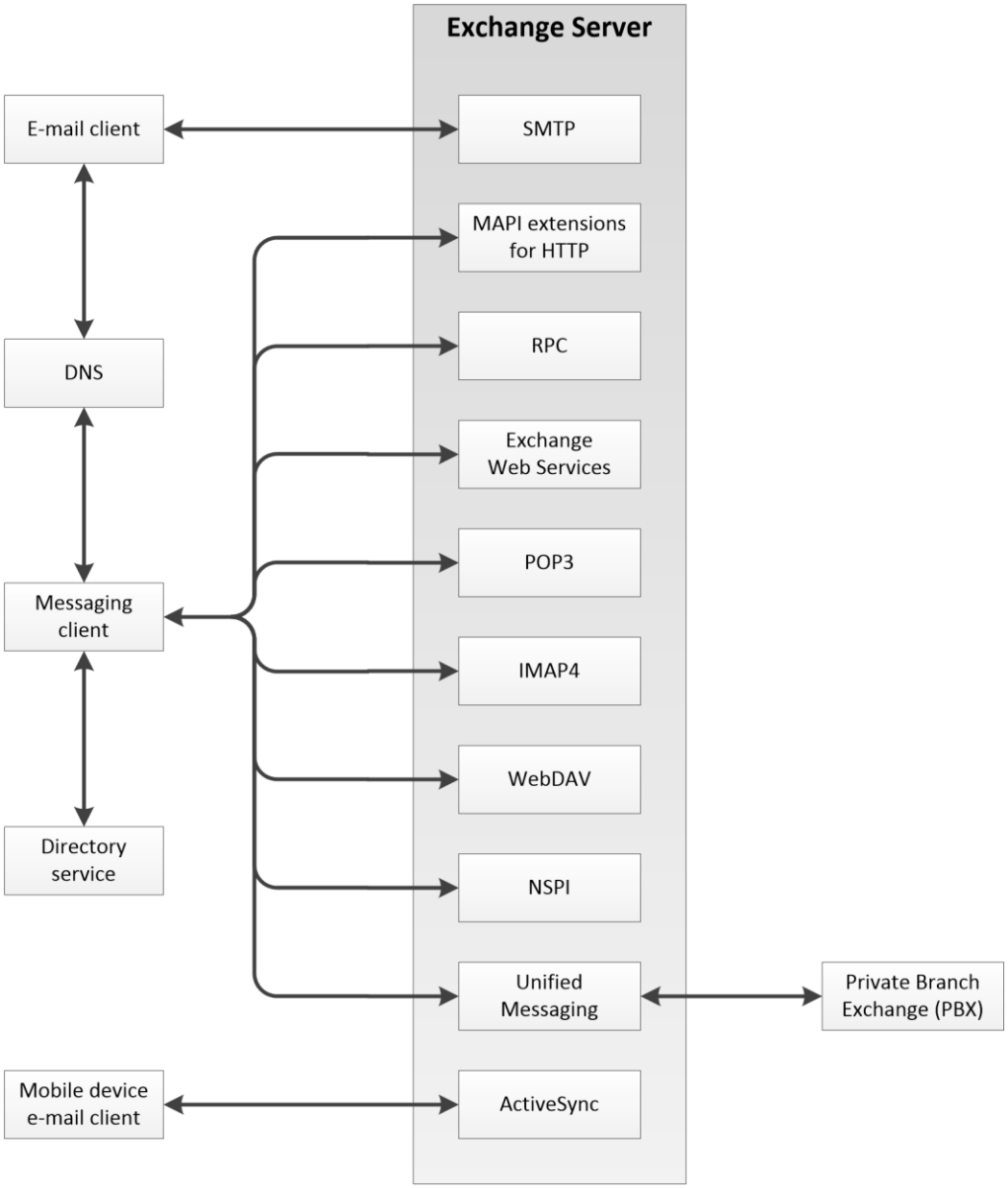

Figure 2: Functional architecture

Each protocol exposes a set of functionality that pertains to specific classes of operations. For example, the **Simple Mail Transfer Protocol (SMTP)**, the **Post Office Protocol - Version 3 (POP3)**, and the **Internet Message Access Protocol - Version 4 (IMAP4)** constitute a set of Internet Standard protocols that simple e-mail clients use to send, retrieve, and manage e-mail messages; Exchange Web Services offers a standardized interface for middle-tier applications to build value-added services; the **Web Distributed Authoring and Versioning Protocol (WebDAV)** provides a set of interfaces that caters to distributed authoring; and the remote operations (ROPs) along with either the **remote procedure call (RPC)** interface or the MAPI extensions for HTTP provide all of the above as well as direct access to storage and retrieval services.

In the simplest sense, the Exchange server operates under the common client-server architecture, where a messaging client connects to an Exchange server by using one or more of the available protocols. The client performs tasks by issuing a series of requests to the server and processing server responses. Behind the simplicity of the client-server architecture lies functionality from basic storage to accessing, updating, and synchronizing **address books**, appointments, and shared folders.

An Exchange server can be regarded as having two functional elements: a message store and a message processing system. These functions are explained in more detail in section 2.1.1 and section 2.1.2.

2.1.1 Message Store

The message store provides storage functionality for Exchange servers, as described in [MS-OXCSTOR]. From a functional point of view, the message store is a hierarchical storage system consisting of folders and messages. The message store also implements a wide range of methods to access, classify, render, and synchronize data between Exchange servers and clients.

2.1.2 Message Processing System

The message processing system consists of anything not directly related to storage, including the processing that happens when a message is in transit to and from storage. For example, when a new message is received, message processing determines whether the message needs to be placed into storage or whether and where it is routed. Similarly, when a new message is submitted for delivery, message processing retrieves the message from storage and determines whether content conversion is required and whether and where it is routed.

2.1.3 Communications within the System

2.1.3.1 Between an E-Mail Client and Exchange Servers

Communication between an e-mail client and Exchange servers implements SMTP or SMTP plus Exchange-specific extensions to SMTP, as described in [MS-OXSMTP], for e-mail transmission.

2.1.3.2 Between a Messaging Client and Exchange Servers

In the context of communication between a messaging client and Exchange servers, "messaging client" refers to any generic client that uses the Microsoft Exchange messaging system. A messaging client does not necessarily have to be an e-mail client. As illustrated in the figure in section 2.1, messaging clients have a variety of protocol options to communicate with Exchange servers: RPC, MAPI extensions for HTTP, POP3, IMAP4, WebDAV, Web Services, NSPI, and **Unified Messaging**.

2.1.3.3 Between a Mobile Client Device and Exchange Servers

Mobile client devices communicate with Exchange servers via the Exchange ActiveSync protocols.

2.1.3.4 Between a Messaging Client and Directory Service

Messaging clients communicate with the directory service to locate Autodiscover sites, as described in [MS-OXDISCO]. This communication uses the **Lightweight Directory Access Protocol (LDAP)**.

2.1.3.5 Between a Messaging Client and DNS

Messaging clients communicate with **Domain Name System (DNS)** servers to locate alternate Autodiscover sites. For information about Autodiscover, see [MS-OXDISCO].

2.2 Protocol Summary

The tables in the following sections provide a comprehensive list of the Member Protocols of the Microsoft Exchange System. The Member Protocols are grouped according to their primary purpose.

2.2.1 Microsoft Exchange Supplemental Documents

Protocols in this table enable consistency throughout the open specifications by providing data structures, terms, properties, and reference information that is broadly useful.

Protocol name	Description	Short name
Data Structures	Describes common data structures that are used in remote operations.	[MS-OXCDATA]
Exchange Server Protocols Master Property List	Lists all properties used for communication between clients and servers, provides summary information about each property, and provides links to the documents in which the property value ranges and semantics are specified.	[MS-OXPROPS]

2.2.2 ROP Primer/Storage and Retrieval Protocols

The ROP primer protocols enable the packaging and transmitting of data between clients and servers. The storage and retrieval protocols enable the storage and retrieval of messages related to calendars, tasks, and personal **contacts (2)**. The hierarchical relationships between the ROP storage and retrieval protocols are illustrated in the following figure, in which each protocol is represented by its specification short name.

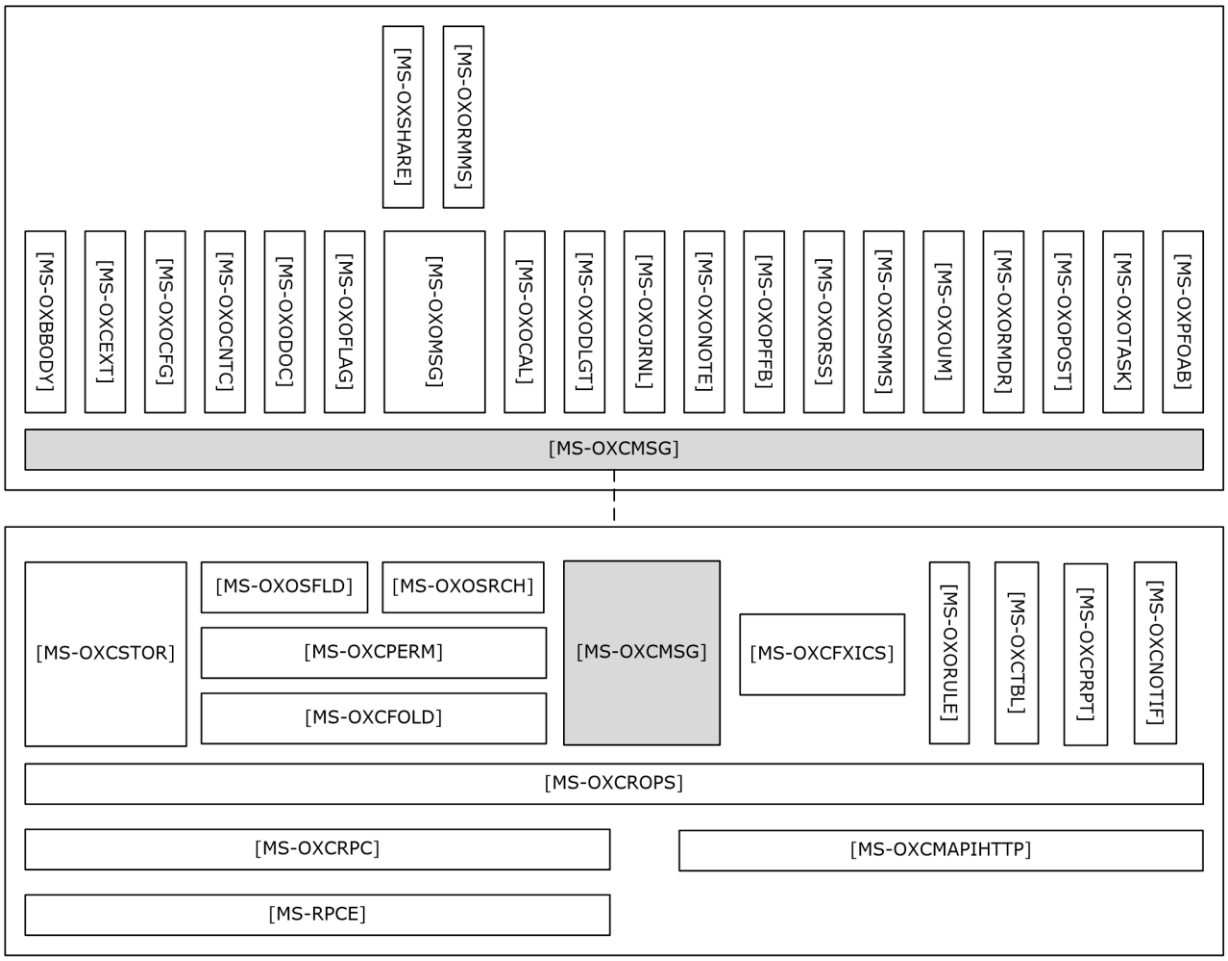

Figure 3: ROP primer/storage and retrieval protocols

2.2.2.1 ROP Primer Protocols

Protocols in this table enable the packaging and transmitting of messaging data between clients and servers.

Protocol name	Description	Short name
Messaging Application Programming Interface (MAPI) Extensions for HTTP	Enables a client to access personal messaging data on a server by sending HTTP requests and receiving responses returned on the same HTTP connection. This protocol extends HTTP and HTTPS.	[MS-OXCMAPIHTTP]
Wire Format Protocol	Serves as the transport basis for client/server communications over RPC.	[MS-OXCRPC]
Remote Operations (ROP) List and Encoding Protocol	Provides the remote operations used to access and modify **mailbox** information on the server.	[MS-OXCROPS]
Store Object Protocol	Used by clients to log on to a user mailbox or **public folders**, read and write mailbox-level properties for that user mailbox, perform various housekeeping tasks for that mailbox, and determine the availability of content for public folders.	[MS-OXCSTOR]

Protocol name	Description	Short name
Message and Attachment Object Protocol	Provides the methods used within the server for manipulating **Message objects**.	[MS-OXCMSG]
Folder Object Protocol	Enables a client to create a folder and to manipulate an existing folder and its contents, which can include messages and subfolders.	[MS-OXCFOLD]
Special Folders Protocol	Describes the default set of folders that an implementation supports, as well as other non-user-visible special folders for certain types of application data, such as reminders and views.	[MS-OXOSFLD]
Table Object Protocol	Used by clients to read and navigate through data that is retrieved in tabular format from the server.	[MS-OXCTABL]

2.2.2.2 ROP Storage and Retrieval Protocols

Protocols and other technologies listed in this table enable the storage and retrieval of messages related to calendars, tasks, and personal contacts (2).

Protocol or other technology name	Description	Short name
Best Body Retrieval Algorithm	Provides a mechanism for efficient storage of **message bodies (2)**.	[MS-OXBBODY]
Client Extension Message Object Protocol	Allows clients to access **mail add-in** data stored in a mailbox.	[MS-OXCEXT]
Bulk Data Transfer Protocol	Responsible for the order and data flow for that is used to transfer data between client and server.	[MS-OXCFXICS]
Core Notifications Protocol	Handles notifications that are sent to a client when specific server events occur.	[MS-OXCNOTIF]
Exchange Access and Operation Permissions Protocol	Used by clients to retrieve and manage the permissions on a folder. Extends the Folder Object Protocol, described in [MS-OXCFOLD]. Extends the Availability Web Service Protocol, described in [MS-OXWAVLS], if both the client and the server support the Availability Web Service Protocol.	[MS-OXCPERM]
Property and Stream Object Protocol	Enables a client to read, set, and delete the properties of an object.	[MS-OXCPRPT]
Appointment and Meeting Object Protocol	Extends the Message and Attachment Object Protocol, as described in [MS-OXCMSG], for use with calendaring.	[MS-OXOCAL]
Configuration Information Protocol	Allows a client to share overlapping application settings with a server. Where appropriate, it can also be used to change the configuration of a feature on the client from the server or vice versa.	[MS-OXOCFG]
Contact Object Protocol	Enables the handling of contacts (2) and personal distribution lists.	[MS-OXOCNTC]
Delegate Access Configuration Protocol	Enables a user to delegate the responsibility for his or her mailbox to another user.	[MS-OXODLGT]
Document Object Protocol	Enables representation of an ordinary file, such as a document generated by a word-processing application, in a mail folder for	[MS-

Protocol or other technology name	Description	Short name
	later retrieval. Extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].	OXODOC]
Informational Flagging Protocol	Allows a Message object to be marked for either follow-up or categorization. Extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].	[MS-OXOFLAG]
Journal Object Protocol	Used to track activity related to a meeting, task, contact (2), or application file. Extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].	[MS-OXOJRNL]
Email Object Protocol	Enables the representation of e-mail messages in a message store.	[MS-OXOMSG]
Note Object Protocol	Enables the representation of a brief note that functions as the electronic equivalent of a paper sticky note. Extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].	[MS-OXONOTE]
Public Folder-Based Free/Busy Protocol	Publishes the availability of a user or resource.	[MS-OXOPFFB]
Post Object Protocol	Enables the representation of a bulletin board post in a message store.	[MS-OXOPOST]
Reminder Settings Protocol	Enables a user to discover and act upon appointments, tasks, messages, or contacts (2) that have a deadline or for which follow-up is necessary.	[MS-OXORMDR]
Rights-Managed Email Object Protocol	Used by clients to create and consume **rights-managed e-mail messages**.	[MS-OXORMMS]
RSS Object Protocol	Enables the representation of an item that is from a news feed. Extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].	[MS-OXORSS]
Email Rules Protocol	Enables the manipulation of incoming e-mail messages on a server.	[MS-OXORULE]
Short Message Service (SMS) and Multimedia Messaging Service (MMS) Object Protocol	Enables clients to create, modify, and delete **Short Message Service (SMS)** and **Multimedia Messaging Service (MMS)** messages. Extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].	[MS-OXOSMMS]
Search Folder List Configuration Protocol	Enables clients to persist a user's search folders on the server.	[MS-OXOSRCH]
Task-Related Objects Protocol	Enables the representation of task-related Message objects in a message store.	[MS-OXOTASK]
Voice Mail and Fax Objects Protocol	Enables servers to create and send Unified Messaging objects.	[MS-OXOUM]
Offline Address Book (OAB) Public Folder Retrieval Protocol	Provides a method for delivering **offline address book (OAB)** data from server to client.	[MS-OXPFOAB]
Sharing Message Object Protocol	Shares mailbox folders between clients.	[MS-OXSHARE]

2.2.3 Content Conversion Protocols

Protocols and other technologies listed in this table enable clients and servers to convert from the standard-based formats into one or more of the Microsoft Exchange-supported formats.

Protocol or other technology name	Description	Short name
iCalendar to Appointment Object Conversion Algorithm	Converts data between iCalendar services, as described in [RFC2445], [RFC2446], [RFC2447], and **Appointment objects** and **Meeting objects**.	[MS-OXCICAL]
RFC 2822 and MIME to Email Object Conversion Algorithm	Converts data between the Internet standard e-mail format, as described in [RFC2822], and the Message object format.	[MS-OXCMAIL]
S/MIME Email Object Algorithm	Handles the conversion of arbitrary clear-signed messages and of **S/MIME** opaque-signed and encrypted messages.	[MS-OXOSMIME]
Rich Text Format (RTF) Compression Algorithm	Compresses and decompresses **RTF** data, as described in [MSFT-RTF], to or from one of the supported compression formats.	[MS-OXRTFCP]
Rich Text Format (RTF) Extensions Algorithm	Encapsulates additional content formats (such as **HTML**) within the RTF body property of messages and attachments.	[MS-OXRTFEX]
Transport Neutral Encapsulation Format (TNEF) Data Algorithm	Encodes and decodes Message objects and **Attachment objects** to an efficient stream representation.	[MS-OXTNEF]
vCard to Contact Object Conversion Algorithm	Converts data between a **vCard** and an object that represents a person.	[MS-OXVCARD]

2.2.3.1 Content Conversion File Formats

File formats in this table enable content conversion.

File format name	Description	Short name
Outlook Item (.msg) File Format	Used to represent individual e-mail messages, appointments, contacts (2), tasks, and so on in the file system.	[MS-OXMSG]
Offline Address Book (OAB) File Format and Schema	Used for the local **Address Book objects** cache.	[MS-OXOAB]
Microsoft ZIP (MSZIP) Compression and Decompression Data Structure	Used for the encoding and decoding of MSZIP compressed data in cabinet files.	[MS-MCI]
LZX DELTA Compression and Decompression	A derivative of the Microsoft Cabinet LZX format with some modifications to facilitate efficient delta compression.	[MS-PATCH]

2.2.4 Exchange ActiveSync Protocols

Exchange ActiveSync protocols enable data to be shared and synchronized between a server and a mobile client device. The Exchange ActiveSync protocols also provide a notification mechanism that

allows clients to synchronize updates when changes occur on the server; for example, when a new e-mail message arrives The hierarchical relationships between the Exchange ActiveSync protocols are illustrated in the following figure, in which each protocol is represented by its specification short name.

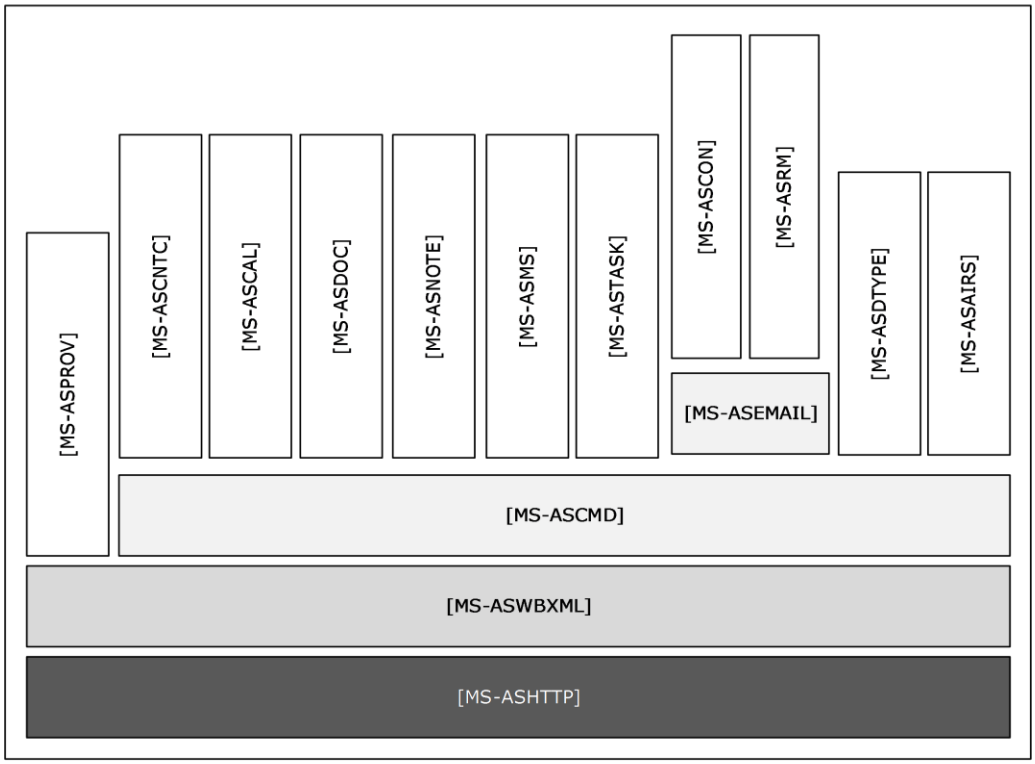

Figure 4: Exchange ActiveSync specifications

The Exchange ActiveSync protocols are listed in the following table.

Protocol or other technology name	Description	Short name
Exchange ActiveSync: AirSyncBase Namespace Protocol	Used by the Exchange ActiveSync commands to identify the size, type, and content of the data sent by and returned to the client.	[MS-ASAIRS]
Exchange ActiveSync: Calendar Class Protocol	Enables the interchange of calendar data between a server and a client device.	[MS-ASCAL]
Exchange ActiveSync: Command Reference Protocol	Enables the synchronization of e-mail, Short Message Service (SMS) messages, attachments, folders, contact (2) information, meetings, calendar data, tasks, notes, and documents.	[MS-ASCMD]
Exchange ActiveSync: Contact Class Protocol	Enables the interchange of contact (1) data between a server and a client device.	[MS-ASCNTC]
Exchange ActiveSync: Conversations Protocol	Improves the ways in which e-mails are displayed in the conversation view.	[MS-ASCON]
Exchange ActiveSync: Document Class Protocol	Communicates document data from the server to the client.	[MS-ASDOC]
Exchange ActiveSync: Data	Data types used by the Exchange ActiveSync **XML schema**	[MS-

Protocol or other technology name	Description	Short name
Types	**definitions (XSDs)**.	ASDTYPE]
Exchange ActiveSync: Email Class Protocol	An XML representation of e-mail data sent or received on mobile devices that communicate by using the Exchange ActiveSync protocols.	[MS-ASEMAIL]
Exchange ActiveSync: HTTP Protocol	Enables a client device to synchronize data with the data that is stored on the server.	[MS-ASHTTP]
Exchange ActiveSync: Short Message Service (SMS) Protocol	Provides the mechanisms for a mobile device to synchronize SMS messages with the server and for the server to send SMS messages through the mobile device.	[MS-ASMS]
Exchange ActiveSync: Notes Class Protocol	Provides the mechanisms that allow a mobile client to synchronize user notes with a server.	[MS-ASNOTE]
Exchange ActiveSync: Provisioning Protocol	Enables servers to communicate security policy settings to client devices.	[MS-ASPROV]
Exchange ActiveSync: Rights Management Protocol	Describes the actions allowed on an e-mail message and its attachments.	[MS-ASRM]
Exchange ActiveSync: Tasks Class Protocol	Enables the interchange of task data between a client and a server.	[MS-ASTASK]
Exchange ActiveSync: WAP Binary XML (WBXML) Algorithm	Enables **Wireless Application Protocol (WAP) Binary XML (WBXML)** encoding.	[MS-ASWBXML]

2.2.5 Directory/Profile Services Protocols

Protocols in this table enable clients to find and use mail server configuration information.

Protocol name	Description	Short name
Autodiscover HTTP Service Protocol	Extends the DNS and directory services to make the location and settings of mail servers available to clients.	[MS-OXDISCO]
Autodiscover Publishing and Lookup Protocol	Enables clients to locate the Autodiscover HTTP service.	[MS-OXDSCLI]

2.2.6 Address Book-related Protocols

Protocols in this table enable access to address book, user, group, and resource information via a directory service or provide address book retrieval format information.

Protocol name	Description	Short name
Address Book Name Service Provider Interface (NSPI) Referral Protocol	Redirects client address book requests to an appropriate address book server.	[MS-OXABREF]
Messaging Application Programming Interface	Enables a client to access directory services on a server by sending HTTP requests and receiving responses returned on	[MS-OXCMAPIHTTP]

Protocol name	Description	Short name
(MAPI) Extensions for HTTP	the same HTTP connection. This protocol extends HTTP and HTTPS.	
Exchange Server Name Service Provider Interface (NSPI) Protocol	Enables Messaging API (MAPI) clients to access the directory service.	[MS-OXNSPI]
Address Book Object Protocol	Describes the properties of various Address Book objects, and how the properties of Address Book objects interrelate.	[MS-OXOABK]
Address Book User Interface Templates Protocol	Describes the properties and operations that are permissible for address book templates.	[MS-OXOABKT]
Offline Address Book (OAB) Retrieval File Format	Enables clients to download full and incremental versions of the OAB.	[MS-OXWOAB]

2.2.7 Standards Support Protocol Extensions

Exchange servers support a number of different standard protocols for e-mail (POP3, SMTP, IMAP4, and WebDAV) and directory information (LDAP). The protocol extensions in this table describe extensions to these standards primarily for authentication and authorization.

Protocol extension name	Description	Short name
Internet Message Access Protocol Version 4 (IMAP4) Extensions	Provides an authentication mechanism based on the **NT LAN Manager (NTLM) Authentication Protocol**, a **delegate access** mechanism to allow a **delegate** to access a **delegator's** mailbox, and support for the IMAP UIDPLUS extension described in [RFC4315].	[MS-OXIMAP4]
Lightweight Directory Access Protocol (LDAP) Version 3 Extensions	Extends LDAPv3, which enables directory access.	[MS-OXLDAP]
Post Office Protocol Version 3 (POP3) Extensions	Extends POP3, as described in [RFC1939], which enables the listing and downloading of mail.	[MS-OXPOP3]
Simple Mail Transfer Protocol (SMTP) Extensions	Extends SMTP standards to facilitate authentication and negotiation between a client and a server and to enable the server to close connections that exceed configured thresholds.	[MS-OXSMTP]
SMTP Protocol: AUTH LOGIN Extension	Extends SMTP to support a simple **base64 encoding** authentication mechanism.	[MS-XLOGIN]
OAUTH 2.0 Authorization Protocol Extensions	Extends the OAuth 2.0 Authentication Protocol: SharePoint Extensions and the JSON Web Token (JWT) to enable server-to-server authentication.	[MS-XOAUTH]

2.2.7.1 Compliance-Related Standards-Based Protocols Support

The documents in this table describe the level of support for standards-based protocols.

Document name	Description	Short name
Outlook iCalendar Standards Support	The level of support provided by Microsoft Office Outlook 2007, Microsoft Outlook 2010, Microsoft Outlook 2013, and Microsoft Outlook 2016 for the Internet iCalendar Protocol (iCalendar), the iCalendar Transport-Independent Interoperability Protocol (iTIP), and the iCalendar Message-Based Interoperability Protocol (iMIP).	[MS-STANOICAL]
Outlook Internet Message Access Protocol (IMAP) Standards Support	The level of support provided by Office Outlook 2007, Outlook 2010, Outlook 2013, and Outlook 2016 for the Internet Message Access Protocol (IMAP).	[MS-STANOIMAP]
Outlook Post Office Protocol Version 3 (POP3) Standards Support	The level of support provided by Office Outlook 2007, Outlook 2010, Outlook 2013, and Outlook 2016 for POP3 service.	[MS-STANOPOP3]
Exchange iCalendar Standards Support	The level of support provided by Microsoft Exchange Server 2007, Microsoft Exchange Server 2010, Microsoft Exchange Server 2013, and Microsoft Exchange Server 2016 for the Internet iCalendar Protocol (iCalendar), the iCalendar Transport-Independent Interoperability Protocol (iTIP), and the iCalendar Message-Based Interoperability Protocol (iMIP).	[MS-STANXICAL]
Exchange Internet Message Access Protocol (IMAP) Standards Support	The level of support provided by Exchange 2007, Exchange 2010, Exchange 2013, and Exchange 2016 for the Internet Message Access Protocol (IMAP).	[MS-STANXIMAP]
Exchange Post Office Protocol Version 3 (POP3) Standards Support	The level of support provided by Exchange 2007, Exchange 2010, Exchange 2013, and Exchange 2016 for POP3 service.	[MS-STANXPOP3]

2.2.8 Message Processing Protocols

2.2.8.1 Journal Message Processing File Format

The file format in this table enables clients and servers to interpret metadata required to journal or archive e-mail messages.

File format name	Description	Short name
Journal Record Message File Format	Extends the protocols described in [RFC2045] and [RFC2046] to enable the formatting of information about an e-mail message that is sent through the server.	[MS-XJRNL]

2.2.8.2 Filter Message Processing Protocols

Protocols and other technologies listed in this table enable clients and servers to interpret metadata for spam, phishing, and postmark validation.

Protocol or other technology name	Description	Short name
Spam Confidence Level Protocol	Enables the handling of allow/block lists and the determination of junk e-mail messages.	[MS-OXCSPAM]
Phishing Warning Protocol	Identifies and marks e-mail messages that are designed to trick recipients into divulging sensitive information (such as passwords and/or other personal information) to a non-trustworthy source.	[MS-OXPHISH]
Email Postmark Validation Algorithm	Creates and validates computational puzzles to reduce the effectiveness of junk e-mail.	[MS-OXPSVAL]

2.2.8.3 Sharing Message Processing Schemas

The schema in this table enables clients and servers to interpret metadata for sharing message data.

Schema name	Description	Short name
Sharing Message Attachment Schema	Establishes a sharing relationship between two servers on behalf of client applications.	[MS-OXSHRMSG]

2.2.9 WebDAV Protocol Extensions

The Web Distributed Authoring and Versioning Protocol (WebDAV) is a set of methods, headers, and content types that extend the Hypertext Transport Protocol – HTTP/1.1, as described in [RFC2068]. WebDAV allows for the reading and writing of data to servers, as described in [RFC2518].

Protocol extensions in this table extend the WebDAV HTTP extensions described in [RFC2518] to provide additional functionality.

Protocol extension name	Description	Short name
Web Distributed Authoring and Versioning (WebDAV) Extensions for Calendar Support	Extends the WebDAV protocol to enable the creation and manipulation of **Calendar objects** by using WebDAV.	[MS-XWDCAL]
Web Distributed Authoring and Versioning (WebDAV) Protocol Security Descriptor Extensions	Extends the WebDAV protocol to enable clients to request and set the **security descriptor** by using the WebDAV methods **PROPFIND** and **PROPPATCH**.	[MS-XWDVSEC]

2.2.10 Web Service Protocols

Web services and HTTP/1.1 provide a standards-based layer upon which to build specific client-server protocols. Protocols and other technologies listed in this section are built on client and server implementations of HTTP/1.1, as specified in [RFC2616]. The WS-I Base Profile 1.0 [WSIBASIC] provides the reference infrastructure for protocols identified as Web Services.

The Web Services specifications can be grouped into functional areas as illustrated in the following diagram.

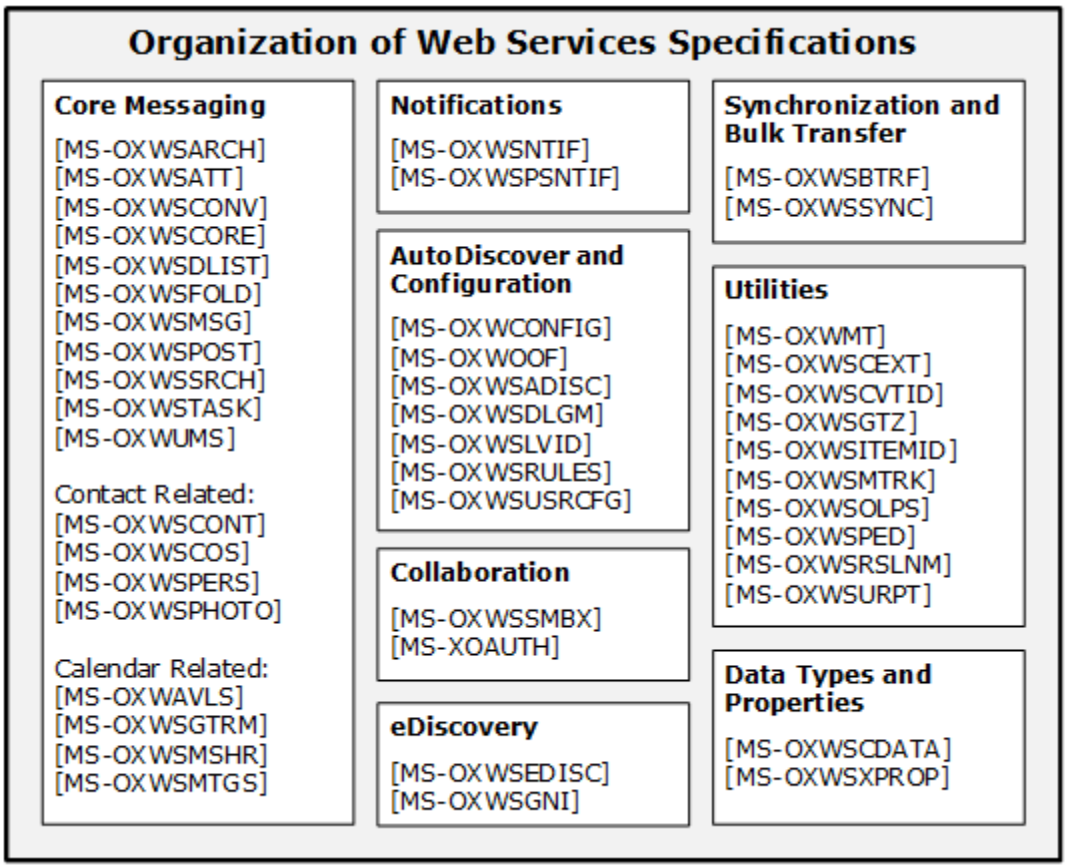

Figure 5: Web Services specifications

The Web Services protocols are listed in the following table.

Protocol or other technology name	Description	Short name
Availability Web Service Protocol	Enables clients to request availability information for users and/or resources.	[MS-OXWAVLS]
Web Service Configuration Protocol	Enables clients to retrieve organization policy configuration information for a mailbox.	[MS-OXWCONFIG]
Mail Tips Web Service Extensions	Extends the Mail Tips Web Service protocol, which enables clients to retrieve mail tips for a mailbox.	[MS-OXWMT]
Voice Mail Settings Web Service Protocol	Enables clients to read and change information about Unified Messaging properties, play voice mail, or record greetings over the telephone.	[MS-OXWUMS]
Out of Office (OOF) Web Service Protocol	Enables clients to configure server-based automatic e-mail responses.	[MS-OXWOOF]
Autodiscover Publishing and Lookup SOAP-Based Web Service Protocol	Enables clients to retrieve user configuration settings information.	[MS-OXWSADISC]
Attachment Handling Web Service Protocol	Enables clients to create, delete, and get attachments on items on the server.	[MS-OXWSATT]

Protocol or other technology name	Description	Short name
Bulk Transfer Web Service Protocol	Enables the export and upload of streamed item data between the server and the client.	[MS-OXWSBTRF]
Common Web Service Data Types	Describes Web service data types that are used by more than one Web service protocol.	[MS-OXWSCDATA]
Contacts Web Service Protocol	Enables clients to create, get, update, delete, move, and copy contact (2) items on the server.	[MS-OXWSCONT]
Conversations Web Service Protocol	Enables clients to find items in a conversation and apply actions to items in a conversation.	[MS-OXWSCONV]
Core Items Web Service Protocol	Enables clients to create, update, and delete items on the server.	[MS-OXWSCORE]
Convert Item Identifier Web Service Protocol	Enables clients to convert among the different identifier formats that can be used to locate items stored on the server.	[MS-OXWSCVTID]
Delegate Access Management Web Service Protocol	Enables clients to manage delegate access to mailbox information that is stored on a server.	[MS-OXWSDLGM]
Distribution List Creation and Usage Web Service Protocol	Enables clients to query the server for distribution lists; expand a distribution list into the constituent e-mail addresses; and create, delete, get, move, update, and copy distribution lists.	[MS-OXWSDLIST]
Folders and Folder Permissions Web Service Protocol	Enables clients to create, copy, move, delete, get, or empty folders and to modify folder permissions that are stored on the server.	[MS-OXWSFOLD]
Get Rooms List Web Service Protocol	Enables clients to retrieve information about meeting rooms from the server.	[MS-OXWSGTRM]
Get Server Time Zone Web Service Protocol	Enables clients to retrieve time zone information that is used by the server.	[MS-OXWSGTZ]
Web Service Item ID Algorithm	Describes how to create and process an item identifier.	[MS-OXWSITEMID]
Federated Internet Authentication Web Service Protocol	Describes the interaction between the server and standard Internet authentication protocols.	[MS-OXWSLVID]
Email Message Types Web Service Protocol	Enables clients to create, update, and delete e-mail items on the server.	[MS-OXWSMSG]
Folder Sharing Web Service Protocol	Enables clients to manage **Calendar folders** that are shared between users in separate organizations.	[MS-OXWSMSHR]
Calendaring Web Service Protocol	Enables clients to create, retrieve, update, move, copy, and delete calendar-related items (that is, appointments, meetings, meeting request messages, meeting response messages, and meeting cancellation messages) on the server.	[MS-OXWSMTGS]
Message Tracking Web Service Protocol	Enables clients to find and retrieve information about message delivery by the server.	[MS-OXWSMTRK]
Notifications Web Service Protocol	Enables clients to receive pull notifications from the server.	[MS-OXWSNTIF]
Online Personal Search Web Service Protocol	Enables a client to search a collection of items and return information about those items (if any) that match.	[MS-OXWSOLPS]

Protocol or other technology name	Description	Short name
Password Expiration Date Web Service Protocol	Enables client applications to query a server to determine when the password for an account will expire so that the application can warn that the password needs to be changed.	[MS-OXWSPED]
Post Items Web Service Protocol	Enables clients to create, retrieve, update, move, copy, and delete **Post objects** on the server.	[MS-OXWSPOST]
Push Notifications Web Service Protocol	Enables clients to receive subscribed event updates that are sent by the server.	[MS-OXWSPSNTIF]
Resolve Recipient Names Web Service Protocol	Enables clients with incomplete recipient identifying information to retrieve a list of matching and similar recipients that are known to the server.	[MS-OXWSRSLNM]
Inbox Rules Web Service Protocol	Enables clients to get Inbox rules and update Inbox rules for messages on the server.	[MS-OXWSRULES]
Mailbox Search Web Service Protocol	Enables clients to search the contents of a mailbox and retrieve the specified folders or items.	[MS-OXWSSRCH]
Mailbox Contents Synchronization Web Service Protocol	Enables clients to keep a local mailbox synchronized with the server mailbox.	[MS-OXWSSYNC]
Tasks Web Service Protocol	Enables clients to create, update, and delete task items on the server.	[MS-OXWSTASK]
User Configuration Web Service Protocol	Enables clients to create, get, update, and delete user configuration objects.	[MS-OXWSUSRCFG]
Extended Properties Structure	Enables clients to access custom properties on items and folders in a server's mailbox.	[MS-OXWSXPROP]
Archiving Web Service Protocol	Enables clients to use a **web service** to archive items in a mailbox.	[MS-OXWSARCH]
Client Extension Web Service Protocol	Enables clients to use a web service to retrieve and disable client extensions.	[MS-OXWSCEXT]
Unified Contact Store Web Service Protocol	Enables clients to use a web service to create, retrieve, update, and delete **instant messaging** contacts (2) and groups.	[MS-OXWSCOS]
Electronic Discovery (eDiscovery) Web Service Protocol	Enables clients to use a web service to implement legal compliance holds, get user hold settings, and search for mailboxes.	[MS-OXWSEDISC]
Nonindexable Item Web Service Protocol	Enables a client to use a web service to retrieve mailbox items that cannot be indexed.	[MS-OXWSGNI]
Persona Web Service Protocol	Enables clients to use a web service to find and retrieve linked contacts (2).	[MS-OXWSPERS]
Photo Web Service Protocol	Enables the transfer of a user photo from a mailbox to a client application that can authenticate and send an **HTTP GET** request.	[MS-OXWSPHOTO]
Site Mailbox Web Service Protocol	Enables clients to use a web service to set the lifecycle state of a **site mailbox** or unpin it from the client.	[MS-OXWSSMBX]
Retention Tag Web Service Protocol	Enables clients to use a web service to retrieve **retention policy** information for items in a mailbox.	[MS-OXWSURPT]

2.3 Environment

The following sections identify the context in which the system exists. This includes the systems that use the interfaces provided by this system of protocols, other systems that depend on this system, and, as appropriate, how elements of the system communicate.

2.3.1 Dependencies on This System

The Microsoft Exchange system supports a diverse set of protocols, ranging from standards-compliant protocols such as SMTP to rich, proprietary protocols such as the Exchange RPC protocols. Due to this flexibility, Exchange servers are able to interoperate with many different types of clients. This section describes the types of clients that rely on the Microsoft Exchange protocols.

2.3.1.1 RPC-Enabled Clients

Messaging clients that are RPC-enabled communicate with Exchange servers through the RPC protocol, which offers the most comprehensive set of functionality. In addition to sending and retrieving messages, RPC-enabled clients gain access to server-side features such as address books, contacts (2), appointments, and shared storage.

2.3.1.2 Non-RPC-Enabled Clients

Messaging clients that are not RPC-enabled communicate with Exchange servers through MAPI extensions for HTTP, which offers the same comprehensive set of functionality as communication through RPC.

Messaging clients that are not RPC-enabled can also communicate with Exchange servers through standards-based protocols such as SMTP, POP3, and IMAP4. However, these protocols offer only standard e-mail services, such as sending and retrieving e-mail messages. More advanced features, such as contacts (2) and appointments, are usually implemented on the client rather than on the server.

2.3.1.3 Mobile Device Clients

Exchange ActiveSync provides mobile device clients with access to a subset of the server-side features that are available to RPC-enabled clients.

2.3.1.4 Document Sharing and Collaboration Services

The flexibility of Exchange servers enables storage and management of documents in the form of schematized messages. This ability to store and manage documents in the form of schematized messages, coupled with the fact that Exchange servers are message transport agents, allows document sharing and collaboration services to be built on top of the Exchange server's architecture.

2.3.1.5 Unified Messaging Clients

Exchange servers include the capability to integrate e-mail and related messaging systems with telephone-based communication. Unified Messaging provides APIs that enable software agents to integrate telephony functionality into a unified Inbox experience.

2.3.2 Dependencies on Other Systems/Components

The Microsoft Exchange system depends on other systems in order to function. The following sections outline these dependencies.

2.3.2.1 Domain Controller/Directory Service

Exchange servers depend on a domain controller to provide authentication services and security policies. This domain controller provides an LDAP-enabled directory service that stores messaging recipient information such as name and e-mail addresses. The directory service is used by the directory and profile services protocols.

2.3.2.2 DNS Service

DNS service is required so that mail servers can resolve host names to IP addresses and route mail accordingly. The DNS service also plays an integral role for outbound message routing and the Directory/Profile Services protocols.

2.4 Assumptions and Preconditions

The assumptions and preconditions that are described in this section apply to the Microsoft Exchange system overall. For information about assumptions and preconditions that apply to a specific protocol in this system, see the specification for that protocol.

- A domain controller is required to service the server domain and authentication requests and to handle management tasks.
- The Exchange servers are members of the domain.
- The Exchange server is reachable by external clients via an established IP address (or IP addresses).
- The appropriate MX DNS records are configured to map the mail domain to the public IP address(es) corresponding to the externally available Exchange server. The MX records are propagated to the extended private or public network(s) so that all intended clients can resolve the domain name.
- The Exchange server functional elements are started collectively, and the Exchange server accepts client requests.
- The **directory service (DS)** is accessible to the Exchange server. Any intermediate firewalls, routers, or connection points between elements of the system have all required ports and gateways open for communication between them.

2.5 Use Cases

This section presents a number of use cases that illustrate the key functionality of Exchange servers. Due to the complexity of the Microsoft Exchange system, the following use cases are not comprehensive. However, they are structured around the core types of activity that a typical client conducts with the system, and they provide a high-level interaction summary between clients and servers.

To further illustrate the flexibility of the Microsoft Exchange system in supporting different types of clients, some use cases are divided into subsections that show how to implement the functionality via protocols that support different types of clients.

The use cases provide high-level detail regarding the goals, actors, and interactions for each scenario and also point to the related documents that provide necessary detailed information.

The actors, actions, and targets of the use cases are intentionally abstract, using generic terms such as "client", "server", "message", and "folder" rather than terms such as "Outlook" or "**Inbox folder**". The examples in section 3 present a number of scenarios that illustrate how one or more use cases can work in conjunction to achieve specific results.

2.5.1 Server Information Discovery

2.5.1.1 Synopsis

The server information discovery use case describes how an **Autodiscover client** in a managed network (domain) locates a list of **URIs** for **Autodiscover servers**. This mechanism applies to clients using any protocol to access the Exchange server.

2.5.1.2 Builds on Use Case(s)

None.

2.5.1.3 References

- [RFC1823]
- [RFC2849]
- [RFC1034]
- [RFC2068]
- [MS-OXDISCO]

2.5.1.4 Requirements

- The client has to have a configured e-mail address.
- The client has to have a configured LDAP server.

2.5.1.5 Protocol-Specific Details

Using LDAP and DNS protocols.

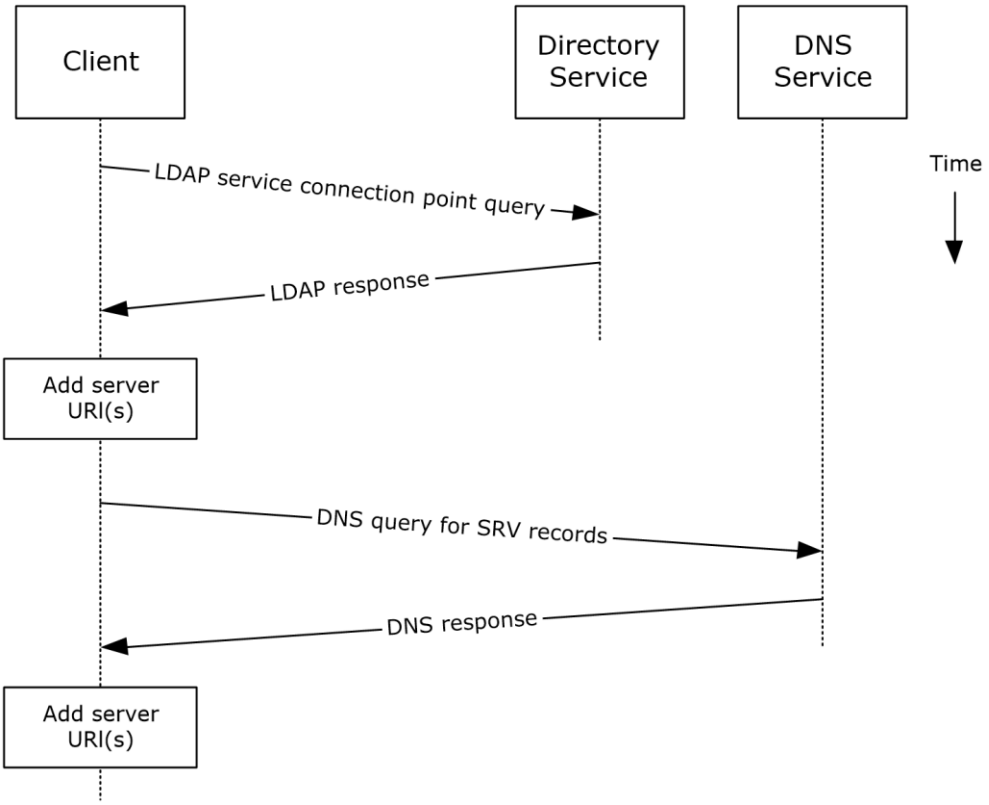

Figure 6: Server information discovery by using LDAP and DNS

1. The client contacts an LDAP server DS for **service connection point** objects via LDAP.
2. The LDAP server returns one or more service connection point objects, which reference one or more Autodiscover server URIs or another LDAP server. If the service connection point returns another LDAP server, repeat step 1 with the new LDAP server until URIs are returned for Autodiscover servers.
3. The client parses the URI and adds the appropriate Autodiscover server URIs to the list of possible Autodiscover server URIs.
4. The client executes a DNS search for SRV records that match the returned Autodiscover server URI.
5. If the DNS server responds with any SRV records, the corresponding Autodiscover server URI records are added to the list of possible Autodiscover server URIs on the client. It is not an error if the DNS server does not return any DNS SRV records in response to the DNS search.
6. The client uses the Autodiscover server URI to contact the Autodiscover server via **HTTP** to query server information.

2.5.2 Log On to a Mailbox

2.5.2.1 Synopsis

The log on to a mailbox use case describes how a messaging client logs on to a mailbox to gain access to its contents.

Note This use case applies only to clients that support RPC and Messaging Application Programming Interface (MAPI) extensions for HTTP.

2.5.2.2 Builds on Use Case(s)

Server information discovery, as described in section 2.5.1.

2.5.2.3 References

- [MS-OXCRPC]
- [MS-OXCMAPIHTTP]
- [MS-OXCDATA]
- [MS-OXCSTOR]
- [MS-OXCROPS]

2.5.2.4 Requirements

- The messaging client has to be able to determine the IP address for the Exchange server.
- The client has to have the **distinguished name (DN)** for a valid mailbox account.

2.5.2.5 Protocol-Specific Details

Using RPC

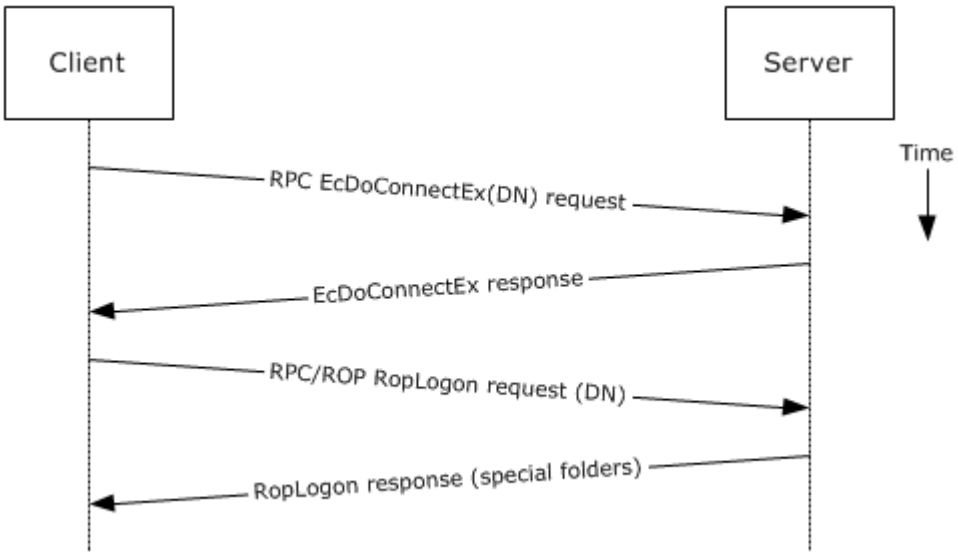

Figure 7: Logging on to a mailbox by using RPC

1. The client uses the discovery process from the use case described in section 2.5.1 to identify the appropriate server.
2. The client connects to the Exchange server via RPC and issues an **EcDoConnectEx** call, as described in [MS-OXCRPC] section 3.1.4.1, along with the client's version information.

3. The Exchange server accepts the connection request and responds with the server version and other connection information.

4. The client issues a **EcDoRpcExt2** call that includes the **RopLogon** **remote operation (ROP)** request ([MS-OXCROPS] section 2.2.3.1) to attempt to log on to the mailbox DN.

5. Upon successful logon, the Exchange server returns a list of special folder IDs, as described in [MS-OXCDATA] section 2.2.1.1, depending on the logon action requested by the client.

Using MAPI extensions for HTTP

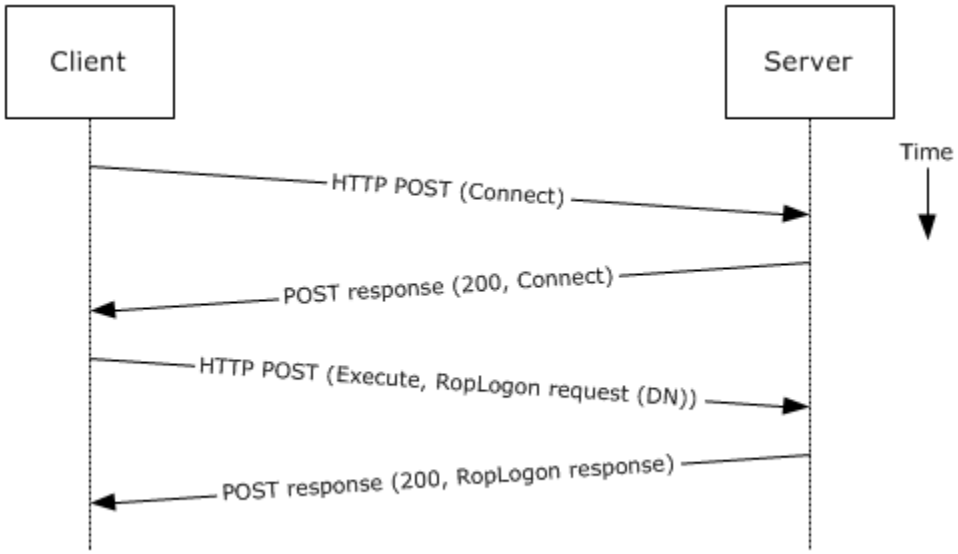

Figure 8: Logging on to a mailbox by using MAPI extensions for HTTP

1. The client uses the discovery process from the use case described in section 2.5.1 to identify the appropriate server.

2. The client connects to the Exchange server via MAPI extensions for HTTP and issues a **Connect** request type, as described in [MS-OXCMAPIHTTP] section 2.2.4.1, along with the client's version information.

3. The Exchange server accepts the connection request and responds with the server version and other connection information.

4. The client issues an **Execute** request type, as described in [MS-OXCMAPIHTTP] section 2.2.4.2, that includes the **RopLogon** remote operation (ROP) request ([MS-OXCROPS] section 2.2.3.1) to attempt to log on to the mailbox DN.

5. Upon successful logon, the Exchange server returns a list of special folder IDs, as described in [MS-OXCDATA] section 2.2.1.1, depending on the logon action requested by the client.

2.5.3 Create a Message

2.5.3.1 Synopsis

The create a message use case describes how a messaging client can create a new message.

2.5.3.2 Builds on Use Case(s)

Log on to a mailbox, as described in section 2.5.2

2.5.3.3 References

- [MS-OXCROPS]
- [MS-OXCMSG]
- [MS-OXCFOLD]
- [MS-OXWSMSG]
- [MS-ASDTYPE]
- [MS-ASHTTP]
- [MS-ASCMD]
- [MS-ASEMAIL]
- [RFC2068]

2.5.3.4 Requirements

The client knows the ID of the folder in which to create the message. The folder ID (FID), as described in [MS-OXCDATA] section 2.2.1.1, can be one of the **special folders** returned from the logon operation.

2.5.3.5 Protocol-Specific Details

The requests and responses below are separated to facilitate understanding conceptual communication flow and are for illustration purposes only. This same convention will be used for the use cases hereinafter.

Using remote operations (ROPs)

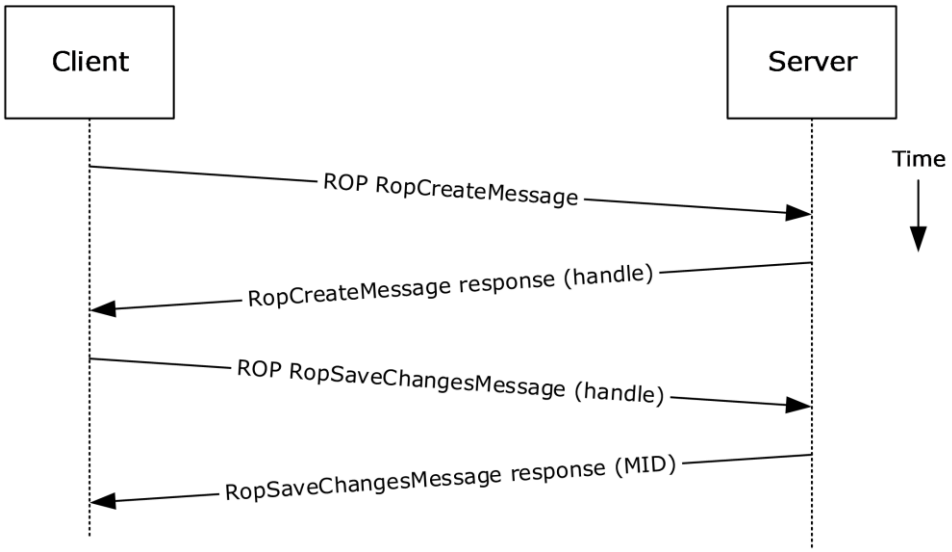

Figure 9: Creating a message by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client logs on to a mailbox per the use case described in section 2.5.2.
2. The client issues a **RopCreateMessage** ROP request ([MS-OXCROPS] section 2.2.6.2) to the Exchange server referencing the folder.
3. The Exchange server responds with a handle to the message.
4. Using the message object handle returned by the server, as described in [MS-OXCROPS] section 2.2.6.2.2, the client issues a **RopSaveChangesMessage** ROP request ([MS-OXCROPS] section 2.2.6.3) to the server to persist the message to storage.
5. The server returns the MID for the new message.

Using Exchange ActiveSync

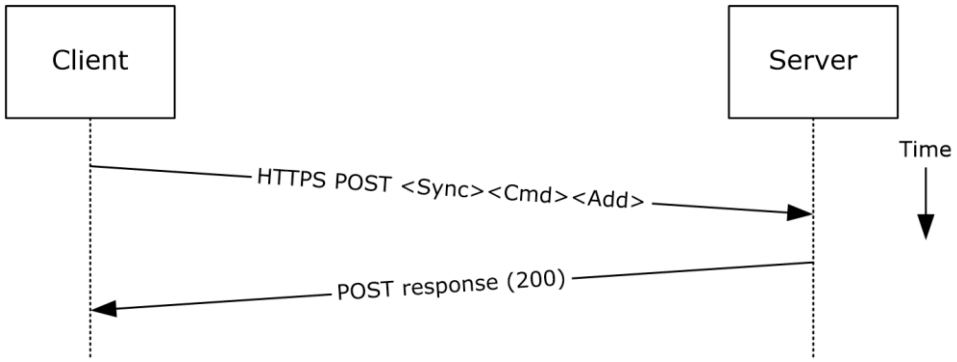

Figure 10: Creating a message by using Exchange ActiveSync

1. The client uses the **Sync** command request ([MS-ASCMD] section 2.2.2.21) with an **Add** element, as described in [MS-ASCMD] section 2.2.3.7.2, to upload/create new application data on the server. An **ApplicationData** element, as described in [MS-ASCMD] section 2.2.3.11, with class "E-mail" needs to contain the required XML schema elements, as described in [MS-ASEMAIL].
2. The Exchange server responds with HTTP status code 200 (OK).

Using Exchange Web Services

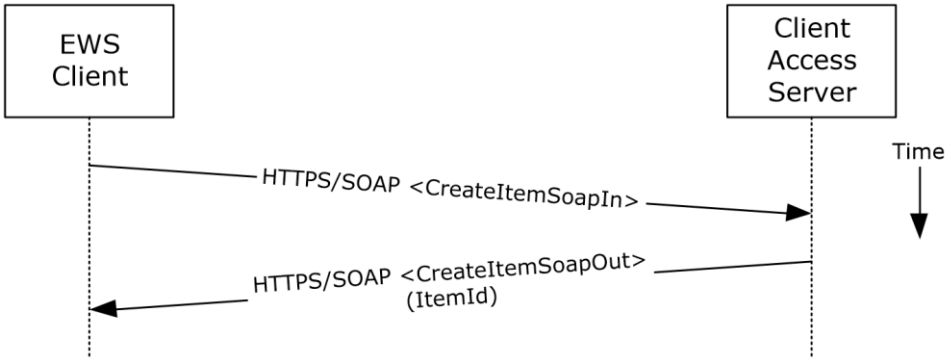

Figure 11: Creating a message by using Exchange Web Services

1. The client uses the HTTPS/SOAP **CreateItemSoapIn** request **WSDL message**, as described in [MS-OXWSMSG] section 3.1.4.2, to create a new message item in the specified folder.

2. The Client Access server responds with a **CreateItemSoapOut** response WSDL message, as described in [MS-OXWSMSG] section 3.1.4.2, which includes the **ResponseCode** element, indicating the status of the operation, and the **ItemId** element, whose value uniquely identifies the new message.

2.5.4 Create a Strongly Typed Message

2.5.4.1 Synopsis

The strongly typed message use case describes how a messaging client creates a new strongly typed message. A strongly typed message is associated with a specific message **class** and has specific requirements for the set of required properties.

2.5.4.2 Builds on Use Case(s)

Create a message, as described in section 2.5.3

2.5.4.3 References

- [MS-OXPROPS]
- [MS-OXOMSG]
- [MS-OXOCFG]
- [MS-OXOCNTC]
- [MS-OXODOC]
- [MS-OXOFLAG]
- [MS-OXOSMIME]
- [MS-OXORMMS]
- [MS-OXOCAL]
- [MS-OXODLGT]
- [MS-OXOJRNL]
- [MS-OXONOTE]
- [MS-OXOPFFB]
- [MS-OXORSS]
- [MS-OXOSMMS]
- [MS-OXORMDR]
- [MS-OXOPOST]
- [MS-OXOTASK]
- [MS-OXWSMSG]
- [MS-XWDCAL]
- [RFC2518]

2.5.4.4 Requirements

The kind of strongly typed message to be created needs to be predetermined.

2.5.4.5 Protocol-Specific Details

Using remote operations (ROPs)

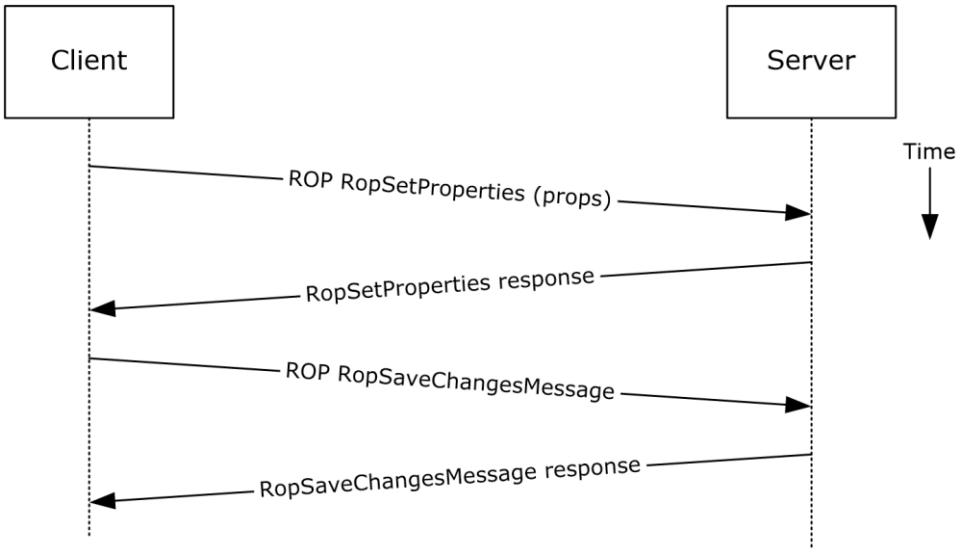

Figure 12: Creating a strongly typed message by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client creates a message per the use case described in section 2.5.3.
2. The client prepares a list of property-value pairs that will be set on the message. The property-value pairs include the message class and the required properties associated with the specific message class value.
3. The client issues a **RopSetProperties** ROP request ([MS-OXCROPS] section 2.2.8.6) to the Exchange server to set the property-value pairs on the message.
4. The Exchange server returns success or failure of the operation.
5. The client issues a **RopSaveChangesMessage** ROP request ([MS-OXCROPS] section 2.2.6.3) to the Exchange server to persist the changes.
6. The Exchange server returns success or failure of the save operation. This operation can fail when any of the required properties associated with the selected message class are not present.

Using Exchange ActiveSync

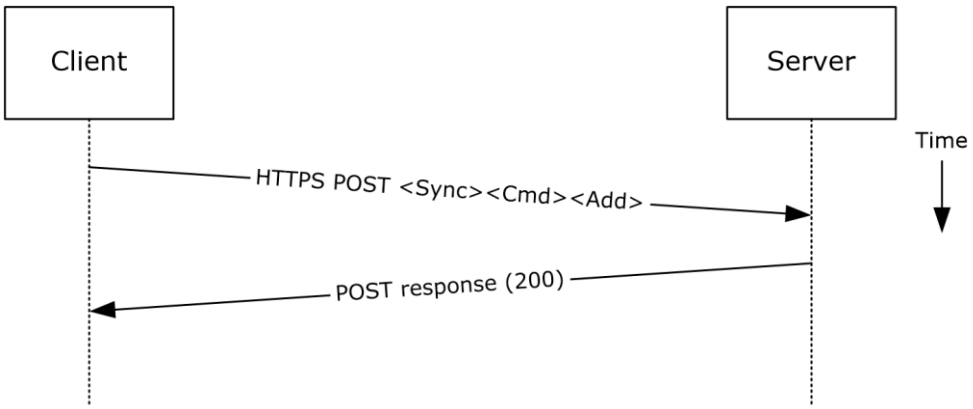

Figure 13: Creating a strongly typed message by using Exchange ActiveSync

1. The client creates a new message per the use use case described in section 2.5.3.
2. The client prepares a list of XML schema item-value pairs to be set on the message. The item-value pairs include the XML schema class name that corresponds to the kind of strongly typed message that is desired and the required items associated with the specific schema class.
3. The client uses the **Sync** command request ([MS-ASCMD] section 2.2.2.21) with an **Add** element, as described in [MS-ASCMD] section 2.2.3.7.2, to add the required schema items to the message.
4. The Exchange server responds with HTTP status code 200 (OK).

Using Exchange Web Services

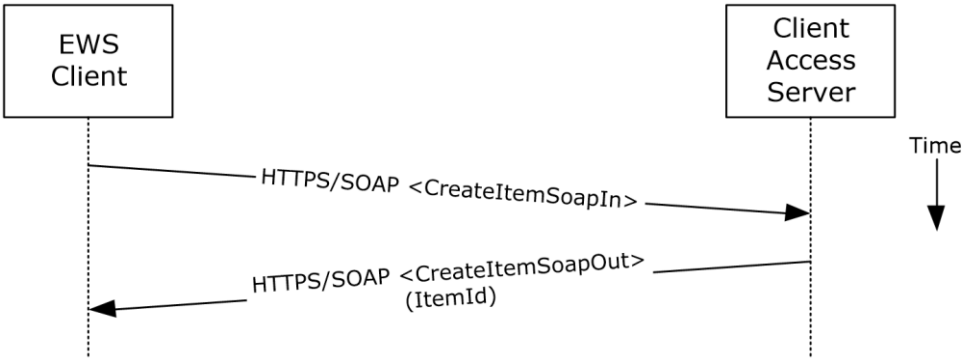

Figure 14: Creating a strongly typed message by using Exchange Web Services

1. The client uses the HTTPS/SOAP **CreateItemSoapIn** request WSDL message, as described in [MS-OXWSMSG] section 3.1.4.2, to create a new message item in the specified folder. The **Message** element contains the properties and child elements that specify the message class and the required properties associated with the specified message class.
2. The Client Access server responds with a **CreateItemSoapOut** response WSDL message, as described in [MS-OXWSMSG] section 3.1.4.2, which includes a **ResponseCode** element specifying the status of the operation and an **ItemId** element whose value uniquely identifies the new message.

2.5.5 Add an Attachment

2.5.5.1 Synopsis

The add an attachment use case describes how a client adds an attachment to a message.

2.5.5.2 Builds on Use Case(s)

Create a message, as described in section 2.5.3

2.5.5.3 References

- [MS-ASCMD]
- [MS-OXCMSG]
- [MS-OXWSATT]
- [MS-OXWSMSG]

2.5.5.4 Requirements

The attachment is available to the client.

2.5.5.5 Protocol-Specific Details

Using remote procedures (ROPs)

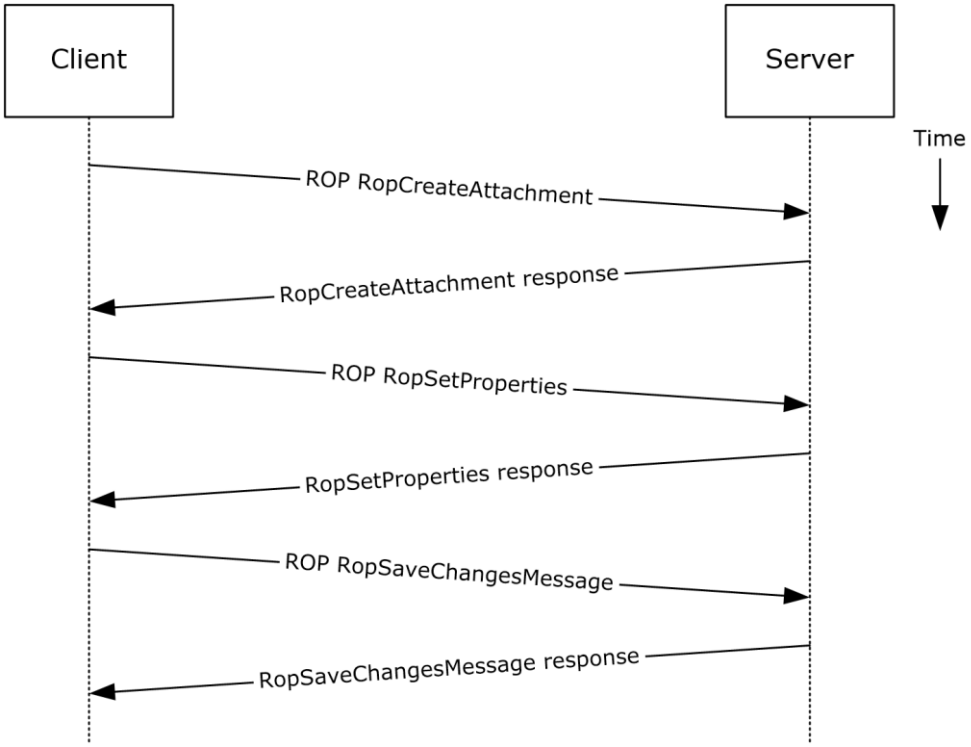

Figure 15: Adding an attachment by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client creates a message per the use case described in section 2.5.3.
2. The client issues a **RopCreateAttachment** ROP request ([MS-OXCROPS] section 2.2.6.13) to the Exchange server to create a new attachment associated with the message.
3. Upon success, the Exchange server returns a handle to the attachment.
4. The client prepares a list of required and optional property-value pairs to be set on the attachment. The attachment data is also set as one of the property values.
5. The client issues a **RopSetProperties** ROP request ([MS-OXCROPS] section 2.2.8.6) to the Exchange server to set the list of property-value pairs on the message.
6. The Exchange server returns the success or failure of the operation.
7. Using the attachment handle, the client issues a **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) attachment request to the Exchange server to persist the new attachment.
8. The Exchange server responds with HTTP status code 200 (OK). This operation can fail when any of the required properties for an attachment are not present.

Using WebDAV

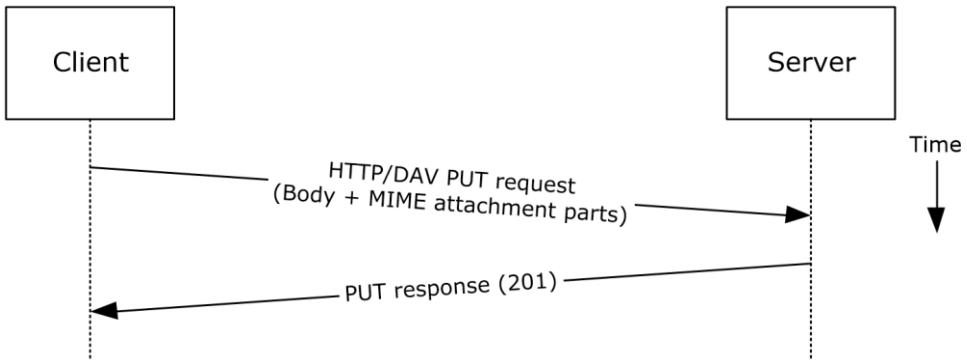

Figure 16: Adding an attachment by using WebDAV

1. The client creates a new message per the use case described in section 2.5.3.
2. The client uses the HTTP **PUT** request, as described in [RFC2068], to replace the existing message document resource, specifying a new message body (1) that contains the attachment data as embedded **MIME** body parts.
3. The Exchange server responds as described in [RFC2068] and Exchange-specific WebDAV extensions.

Using Exchange ActiveSync

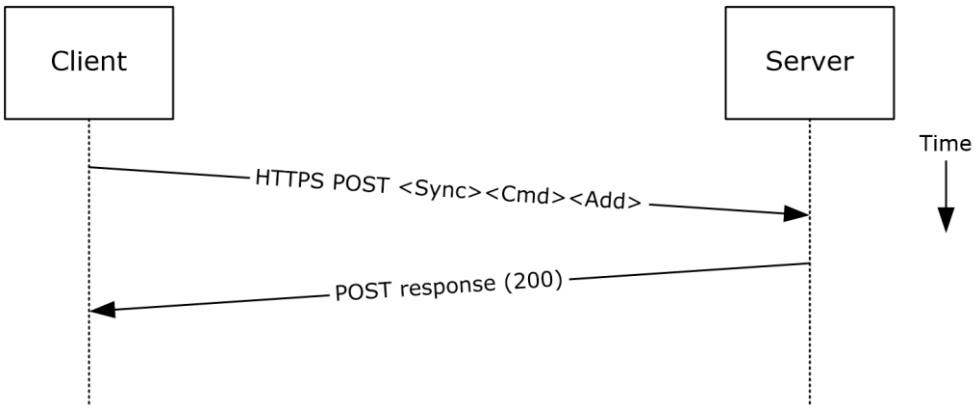

Figure 17: Adding an attachment by using Exchange ActiveSync

1. The client creates a new message per the use case described in section 2.5.3.
2. The client uses the **Sync** command request ([MS-ASCMD] section 2.2.2.21), with an Add element, as described in [MS-ASCMD] section 2.2.3.7.2, to upload/create new ApplicationData of class "Attachment" on the server. The attachment is added to a collection inside the message.
3. The Exchange server responds with HTTP status code 200 (OK) and returns a server ID that identifies the uploaded attachment.

Using Exchange Web Services

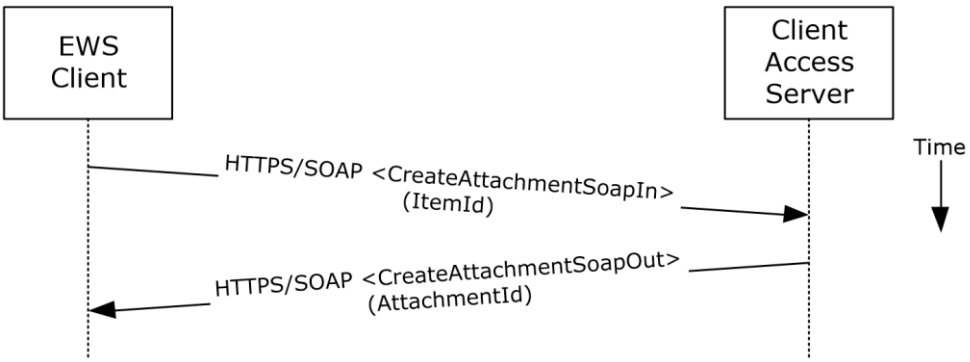

Figure 18: Adding an attachment by using Exchange Web Services

1. The client creates a new message item as described in section 2.5.3 and notes the value of the **ItemId** element for the new message.
2. The client uses the HTTPS/SOAP **CreateAttachmentSoapIn** request WSDL message, as described in [MS-OXWSATT] section 3.1.4.1.1.1, to create a new attachment. To associate the attachment with the message, the client includes the **ItemId** element, which sets the new message as the attachment's parent item.
3. The Client Access server responds with a **CreateAttachmentSoapOut** response WSDL message, as described in [MS-OXWSATT] section 3.1.4.1.1.2, which includes the **ResponseCode** element and the **AttachmentId** element, whose value uniquely identifies the new attachment.

2.5.6 Resolve a Recipient from an Address Book

2.5.6.1 Synopsis

The resolve a recipient from an address book use case describes how a client resolves a recipient and obtains associated information.

Note WebDAV requires the client to handle resolving recipient addresses directly and does not have the corresponding operation.

2.5.6.2 Builds on Use Case(s)

None.

2.5.6.3 References

- [MS-NSPI]
- [MS-OXOABK]
- [MS-ASCMD]
- [MS-OXWSRSLNM]

2.5.6.4 Requirements

The client has a specific recipient name to resolve.

2.5.6.5 Protocol-Specific Details

Using RPC

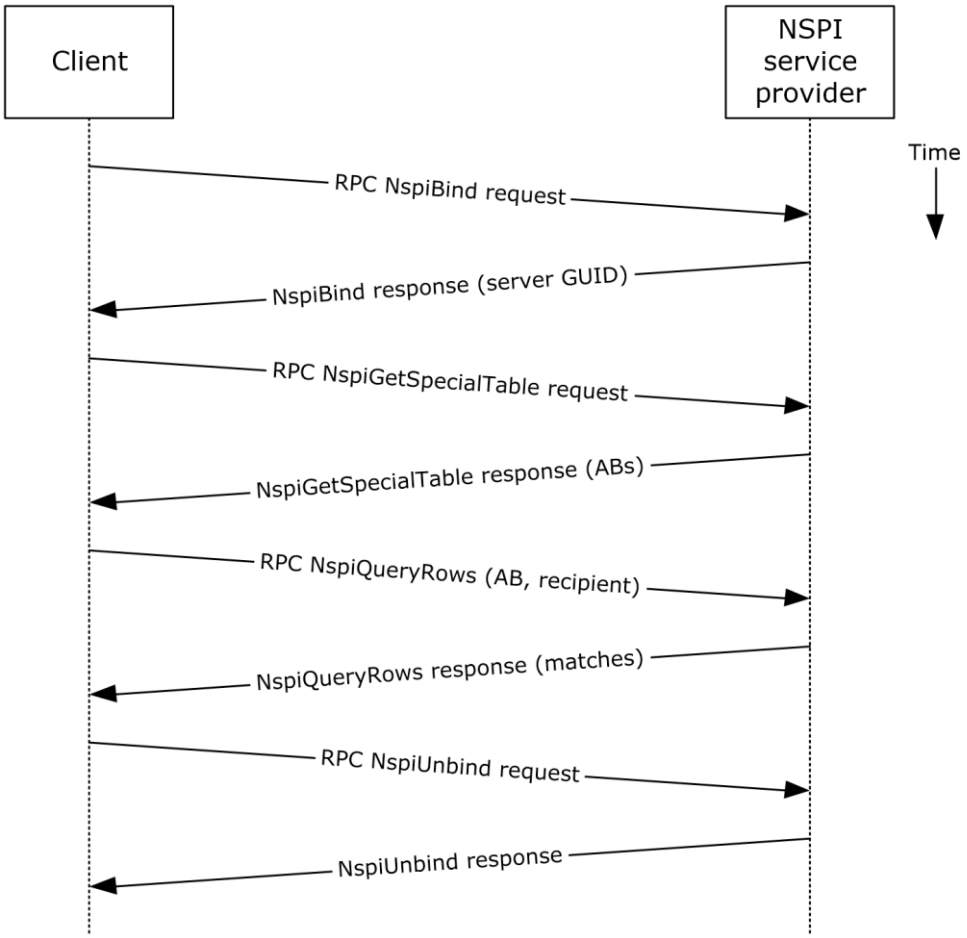

Figure 19: Resolving a recipient from an address book by using RPC

1. The client creates an RPC connection with NSPI and issues an **NspiBind** method request, as described in [MS-NSPI].
2. NSPI responds to the **NspiBind** method request and returns a server **GUID**.
3. The client then issues an **NspiGetSpecialTable** method request to obtain the hierarchy table.
4. NSPI returns a table of rows, where each row represents an address book container.
5. The client issues an **NspiQueryRows** method request to identify address book entries that match a specific recipient.
6. NSPI returns a table of zero or more rows, where each row represents the information for a matching address book entry. If no matches are found, no rows are returned.
7. The client issues an **NspiUnbind** method request to terminate the conversation.
8. The server acknowledges the termination by responding to the **NspiUnbind** method request.

> **Note** If no rows are returned, a match was not found for the recipient. If a single row is returned, an exact match was found. If more than one row is returned, the recipient name in question is considered to be ambiguous. It is the responsibility of the end user to determine the correct recipient from the list of ambiguous recipient names.

Using Exchange ActiveSync

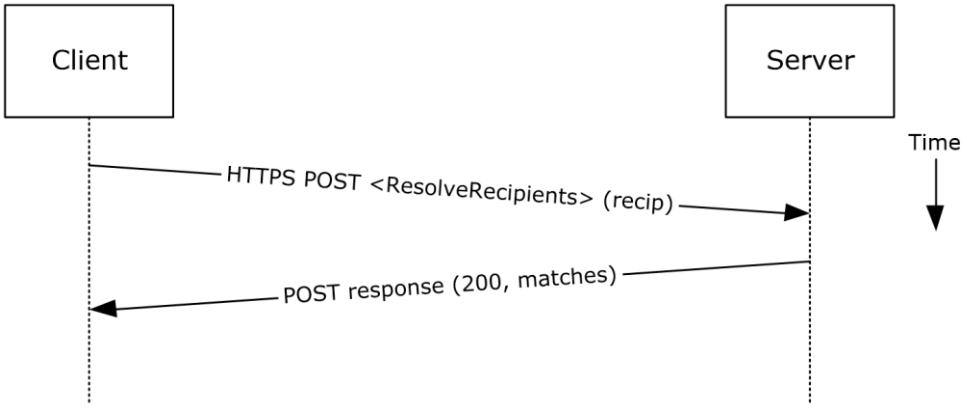

Figure 20: Resolving a recipient from an address book by using Exchange ActiveSync

1. The client issues a **ResolveRecipients** command request, as described in [MS-ASCMD] section 2.2.2.15, for the specific recipient to the server.

2. The Exchange server responds to the request by returning HTTP status code 200 (OK) and a sequence of complex elements, each representing a matching recipient in the address book. A sequence of zero elements indicates that a match for the recipient was not found.

Using Exchange Web Services

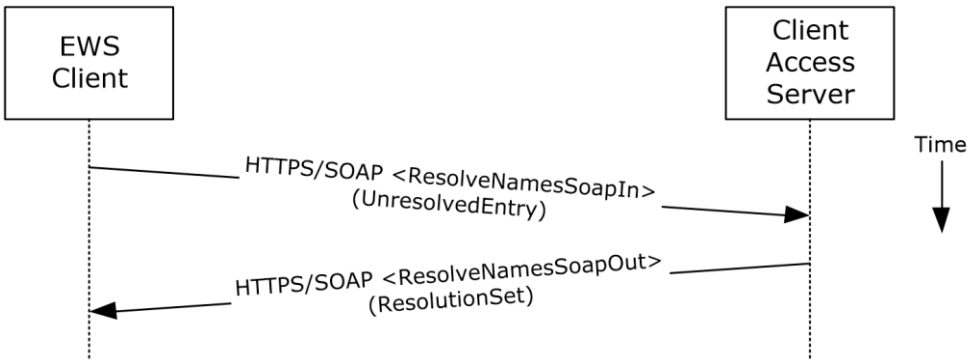

Figure 21: Resolving a recipient from an address book by using Exchange Web Services

1. The client uses the HTTPS/SOAP **ResolveNamesSoapIn** request message, as described in [MS-OXWSRSLNM] section 3.1.4.1.1.1, to resolve an unresolved entry.

2. The Client Access server responds with a **ResolveNamesSoapOut** response message, as described in [MS-OXWSRSLNM] section 3.1.4.1.1.2, which includes the **ResponseCode** element and the **ResolutionSet** element, as described in [MS-OXWSRSLNM] section 3.1.4.1.3.3, containing the list of matching names found in the resolution set.

2.5.7 Send a Message

2.5.7.1 Synopsis

The send a message use case describes how a messaging client can send a message (submit a message for delivery). SMTP-only clients cannot send messages using this mechanism. For more information about SMTP-based message submission, see [MS-OXSMTP] and [RFC2821].

2.5.7.2 Builds on Use Case(s)

- Create a message, as described in section 2.5.3
- Resolve a recipient from an address book, as described in section 2.5.6

2.5.7.3 References

- [MS-OXOMSG]
- [MS-OXSMTP]
- [RFC2821]
- [MS-OXWSMSG]
- [MS-OXWSCORE]

2.5.7.4 Requirements

The client is able to create a message, as described in section 2.5.3, and resolve a recipient, as described in section 2.5.6.

2.5.7.5 Protocol-Specific Details

Using remote operations (ROPs)

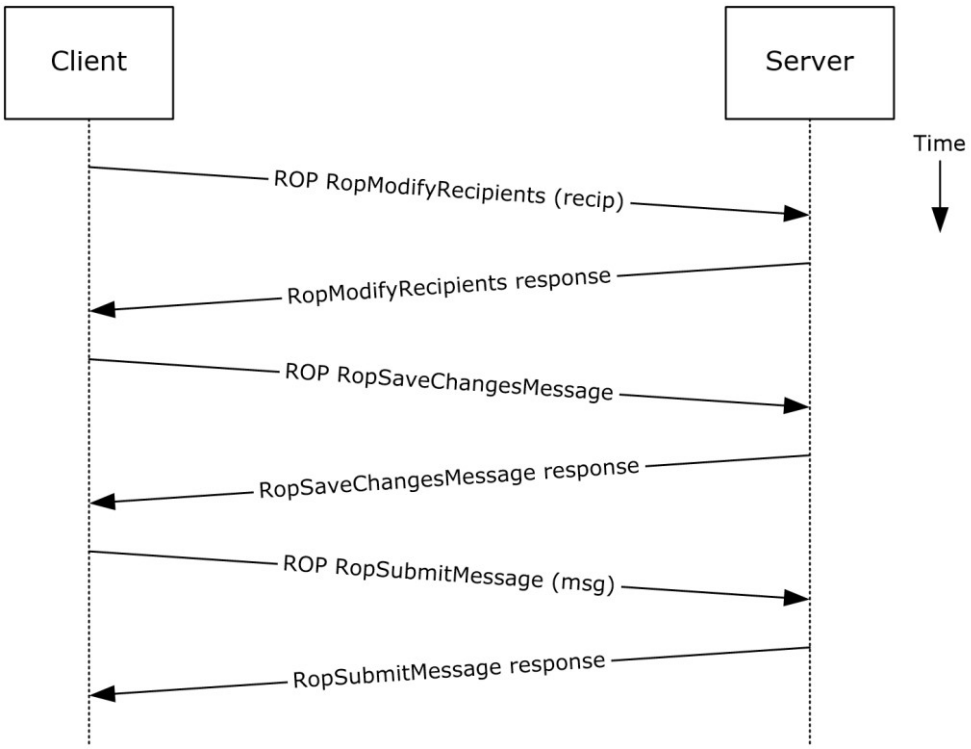

Figure 22: Sending a message by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client creates a message per the use case described in section 2.5.3.
2. The client resolves a recipient per the use case described in section 2.5.6.
3. The client issues a **RopModifyRecipients** ROP request ([MS-OXCROPS] section 2.2.6.5) to add the recipient to the recipient table for the message.
4. The Exchange server returns the success or failure of the operation.
5. The client issues a **RopSaveChangesMessage** ROP request ([MS-OXCROPS] section 2.2.6.3) to save the new recipient table.
6. The Exchange server returns the success or failure of the operation.
7. The client issues a **RopSubmitMessage** ROP request ([MS-OXCROPS] section 2.2.7.1) to submit the message for delivery.
8. The Exchange server returns the success or failure of the operation.

Using Exchange Web Services

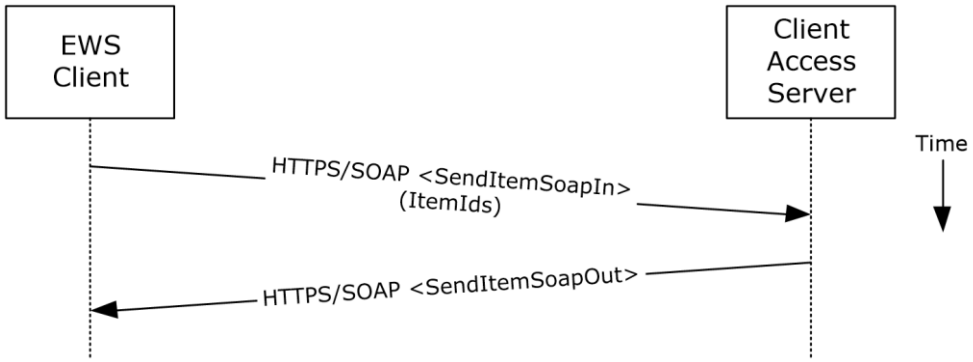

Figure 23: Sending a message by using Exchange Web Services

1. The client uses the HTTPS/SOAP **SendItemSoapIn** request WSDL message, as described in [MS-OXWSCORE] section 3.1.4.8.1.1, to specify a list of messages in the **ItemIds** element. It is assumed that that each item specified in the **ItemIds** element already contains the necessary sender and recipient information.

2. The Exchange Client Access server responds with a **SendItemSoapOut** response WSDL message, as described in [MS-OXWSCORE] section 3.1.4.8.1.2, which includes the **ResponseCode** element specifying the status of the operation.

2.5.8 Send a Message to a Remote Recipient

2.5.8.1 Synopsis

The send a message to a remote recipient use case describes how a messaging client sends a message to a recipient that is stored on a remote server. One of the intentions of this use case is to highlight how content conversion-related protocols are used in message processing.

2.5.8.2 Builds on Use Case(s)

Send a message, as described in section 2.5.7

2.5.8.3 References

- [MS-SMTPNTLM]
- [MS-OXCICAL]
- [MS-OXCMAIL]
- [MS-OXOSMIME]
- [MS-OXRTFCP]
- [MS-OXRTFEX]
- [MS-OXTNEF]
- [MS-OXVCARD]

2.5.8.4 Requirements

The recipient in this case is not stored on the same server as the sender of the message.

2.5.8.5 Protocol-Specific Details

Using remote operations (ROPs)

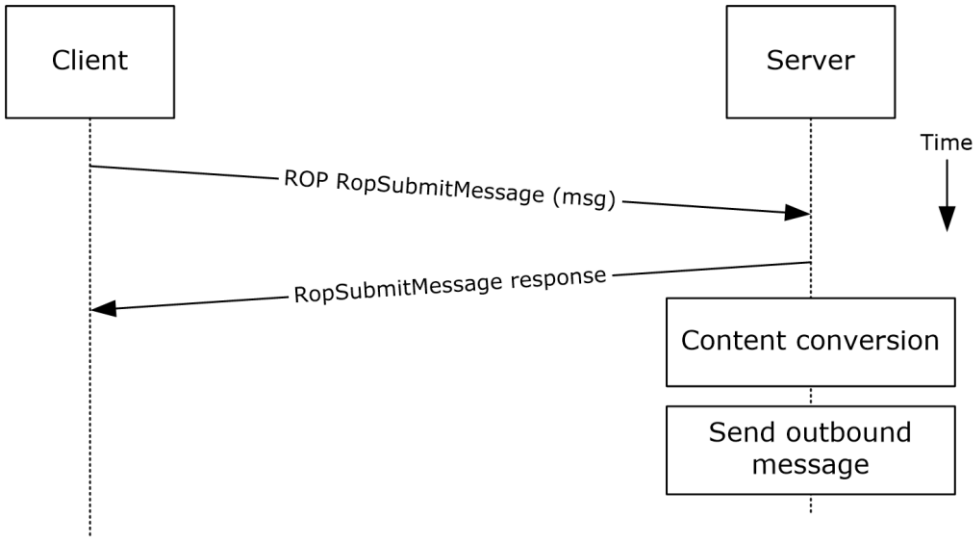

Figure 24: Sending a message to a remote recipient by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The messaging client submits a message for delivery by using a **RopSubmitMessage** ROP request ([MS-OXCROPS] section 2.2.7.1), per the use case described in section 2.5.7, except that the recipient in this case is stored on a different server than the sender.
2. The server determines that the recipient is not stored on the current server and locates the next hop for message delivery.
3. The server invokes content conversion to convert the message into Internet-transmittable form.
4. The server connects to the destination server and transmits the message via SMTP. For information regarding SMTP mail submission and Exchange-specific SMTP extensions, see [RFC2821] and [MS-OXSMTP].

2.5.9 Open a Folder

2.5.9.1 Synopsis

The open a folder use case describes how a client opens a folder.

Note This use case applies only to clients using the RPC protocol or the MAPI extensions for HTTP.

2.5.9.2 Builds on Use Case(s)

Log on to a mailbox, as described in section 2.5.2

2.5.9.3 References

- [MS-OXCFOLD]

- [MS-OXPROPS]

2.5.9.4 Requirements

The client knows the FID, as described in [MS-OXCDATA] section 2.2.1.1, or path for the folder that is to be opened.

2.5.9.5 Protocol-Specific Details

Using remote operations (ROPs)

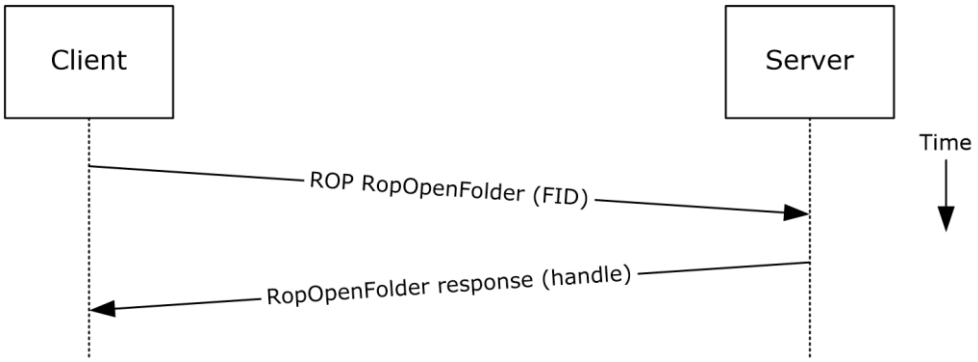

Figure 25: Opening a folder by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client logs on to a mailbox per the use case described in section 2.5.2.
2. The messaging client issues a **RopOpenFolder** request ([MS-OXCROPS] section 2.2.4.1) to the Exchange server to open the specified folder.
3. The server responds with a handle to the folder.

2.5.10 Find Items in a Folder That Match Search Criteria

2.5.10.1 Synopsis

The find items in a folder that match search criteria use case describes how a client finds items in a folder that match specified search criteria.

2.5.10.2 Builds on Use Case(s)

Open a folder, as described in section 2.5.9

2.5.10.3 References

- [MS-OXCDATA]
- [MS-OXCFOLD]
- [MS-OXPROPS]
- [MS-ASCMD]

- [MS-OXWSSRCH]

2.5.10.4 Requirements

The client knows the FID, as described in [MS-OXCDATA] section 2.2.1.1, or path for the folder that is to be searched.

2.5.10.5 Protocol-Specific Details

Using remote operations (ROPs)

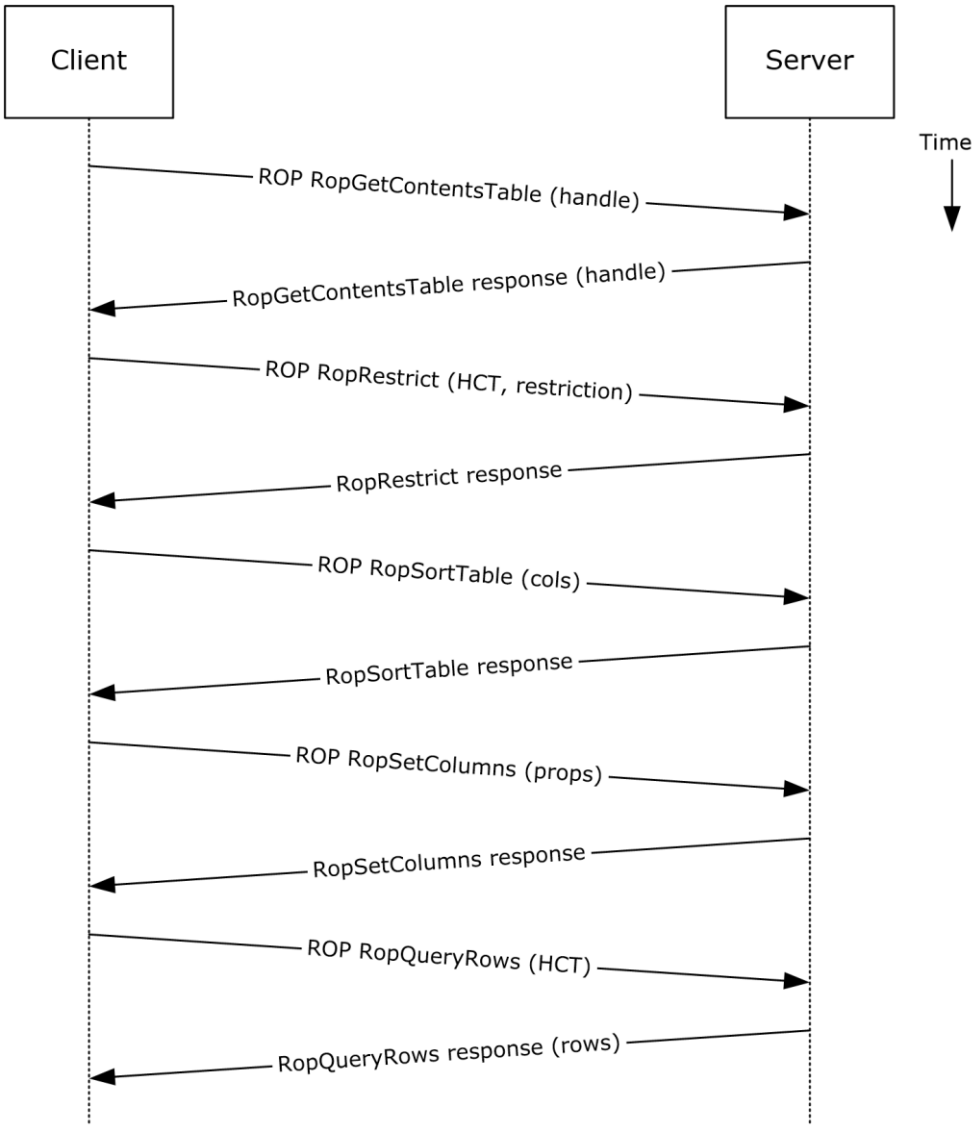

Figure 26: Finding folder items by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client opens the specified folder per the use case described in section 2.5.9.

2. The client issues a **RopGetContentsTable** ROP request ([MS-OXCROPS] section 2.2.4.14) with a **handle** to the folder for which to open the **contents table**.

3. The Exchange server responds with a handle to the contents table.

4. The client builds the search criteria by constructing a **restriction**, as described in [MS-OXCDATA].

5. The client issues a **RopRestrict** ROP request ([MS-OXCROPS] section 2.2.5.3) with the constructed restriction to establish the search criteria.

6. The Exchange server responds to the **RopRestrict** ROP request.

7. The client can optionally issue a **RopSortTable** ROP request ([MS-OXCROPS] section 2.2.5.2) to specify a series of sort columns in the resulting table.

8. The Exchange server responds to the **RopSortTable** ROP request.

9. The client prepares a list of desired properties to retrieve and issues a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) to indicate the desired property columns.

10. The Exchange server responds to the **RopSetColumns** ROP request.

11. The client issues a **RopQueryRows** ROP request ([MS-OXCROPS] section 2.2.5.4) with a contents table handle (HCT) to retrieve rows from the contents table.

12. The Exchange server responds with a table of rows, where each row represents a message in the folder that matches the search criteria from step 4, and each column corresponds to the properties indicated in step 8.

Using WebDAV

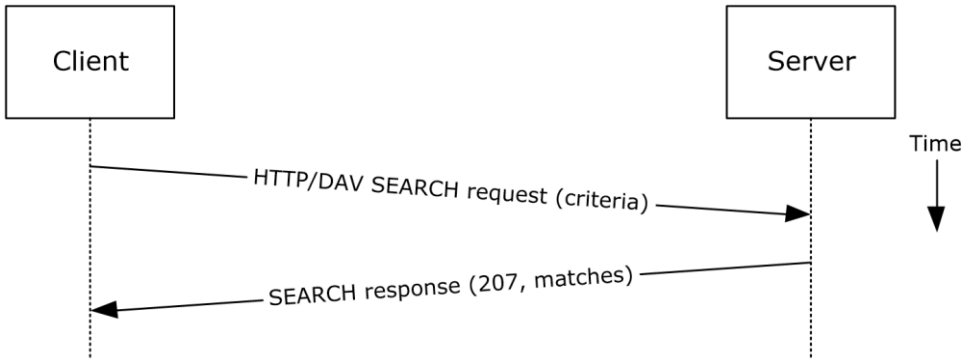

Figure 27: Finding folder items by using WebDAV

1. The client issues a **SEARCH** method request, as described in [MS-WDVSE], to the Exchange server as the Request-URI, referencing the desired folder path to perform the search. The search criteria are expressed in the XML body of the request.

2. The Exchange server responds with HTTP status code 207 (Multi-Status) and a series of responses, where each response corresponds to a matching entry in the folder.

Using Exchange ActiveSync

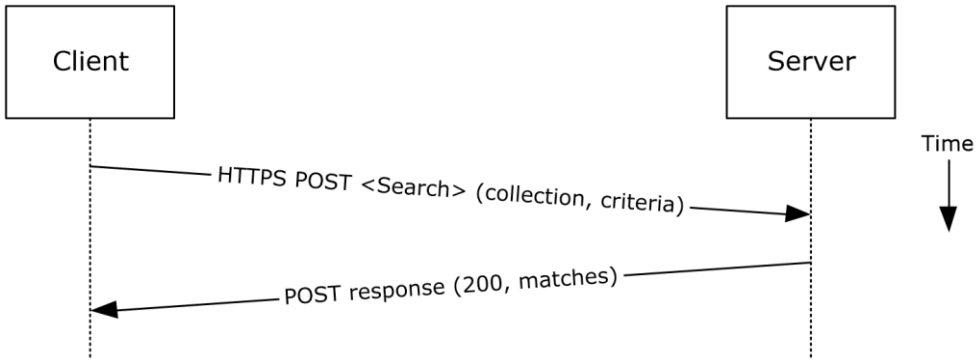

Figure 28: Finding folder items by using Exchange ActiveSync

1. The client issues a **Search** command request, as described in [MS-ASCMD] section 2.2.2.16, by specifying the name of the folder to be searched, and an XML query that represents the search criteria.

2. The Exchange server responds with HTTP status code 200 (OK) and a collection of results, where each result corresponds to an item in the folder that matches the search criteria.

Using Exchange Web Services

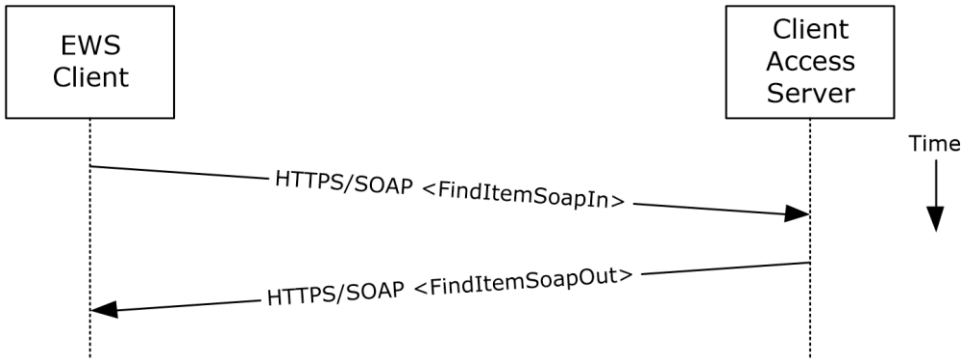

Figure 29: Finding folder items by using Exchange Web Services

1. The client uses the HTTPS/SOAP **FindItemSoapIn** request message, as described in [MS-OXWSSRCH] section 3.1.4.2.1.1, to find specific items from one or more folders. The client can specify the list of folders to search, the list of properties to return, the search criteria, and the sort order of the results, among other options.

2. The Client Access server responds with a **FindItemSoapOut** response message, as described in [MS-OXWSSRCH] section 3.1.4.2.1.2, which includes the **ResponseCode** element specifying the status of the operation and the set of items that match the search criteria.

2.5.11 Delete Message(s)

2.5.11.1 Synopsis

The delete message use case describes how a messaging client deletes messages.

2.5.11.2 Builds on Use Case(s)

Open a folder, as described in section 2.5.9

2.5.11.3 References

- [MS-OXCDATA]
- [MS-OXCFOLD]
- [MS-OXPROPS]
- [MS-ASCMD]
- [MS-OXWSMSG]

2.5.11.4 Requirements

- The client knows the FID, as described in [MS-OXCDATA] section 2.2.1.1, or path for the folder in which the message(s) reside.
- The client knows the MID(s), as described in [MS-OXCDATA] section 2.2.1.2, or name(s) for the message(s) that are to be deleted.

2.5.11.5 Protocol-Specific Details

Using remote operations (ROPs)

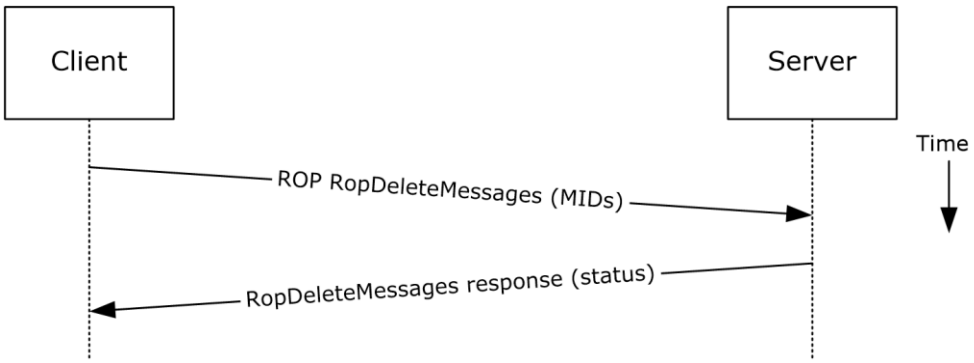

Figure 30: Deleting a message by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client opens the specified folder per the use case described in section 2.5.9.
2. The client issues a **RopDeleteMessages** ROP request ([MS-OXCROPS] section 2.2.4.11) with the list of MIDs to be deleted.
3. The Exchange server returns the success or failure of the operation.

Using Exchange ActiveSync

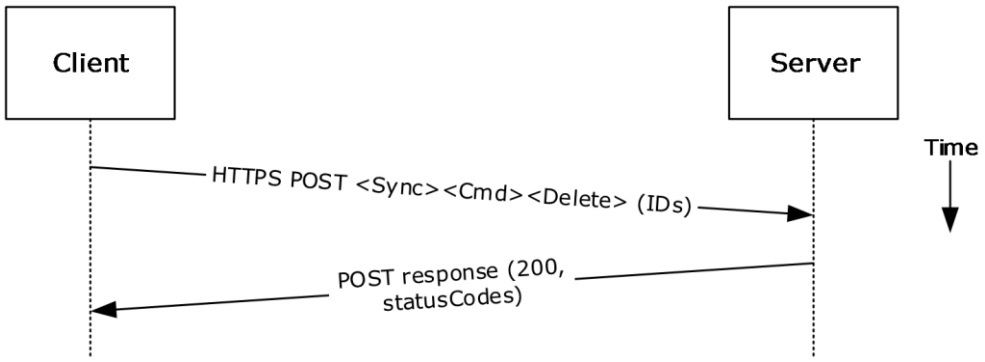

Figure 31: Deleting a message by using Exchange ActiveSync

1. The client issues a **Sync** command request ([MS-ASCMD] section 2.2.2.21), with a **Delete** element, as described in [MS-ASCMD] section 2.2.3.42.2, to the server, where each item to be deleted is listed in the schematized XML request body.

2. The Exchange server responds with HTTP status code 200 (OK) and returns a series of sync status codes, where each corresponds to the deletion status for a message in the deletion list.

Using Exchange Web Services

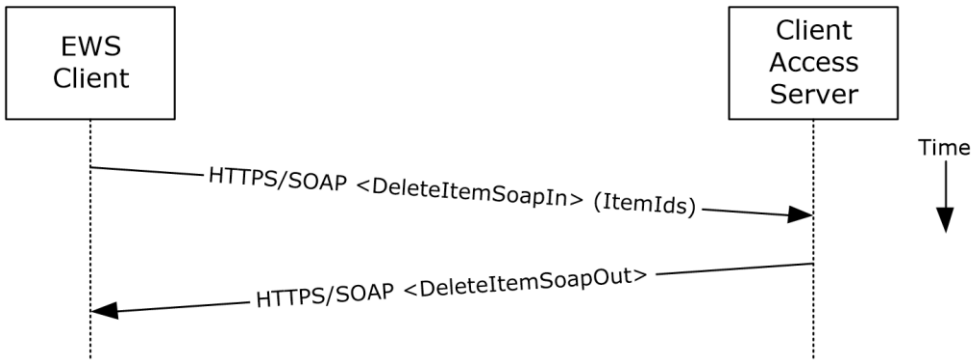

Figure 32: Deleting a message by using Exchange Web Services

1. The client uses the HTTPS/SOAP **DeleteItemSoapIn** request WSDL message, as described in [MS-OXWSCORE] section 3.1.4.3.1.1, to delete items specified in the **ItemIds** element.

2. The Exchange Client Access server responds with a **DeleteItemSoapOut** response WSDL message, as described in [MS-OXWSCORE] section 3.1.4.3.1.2, which includes a **ResponseCode** element for the deletion status of each item.

2.5.12 Synchronize Item(s)

2.5.12.1 Synopsis

The synchronize items use case describes how a client synchronizes the contents of a folder with an Exchange server.

2.5.12.2 Builds on Use Case(s)

Open a folder, as described in section 2.5.9

2.5.12.3 References

- [MS-OXCFXICS]
- [MS-OXCMSG]
- [MS-OXCFOLD]
- [MS-OXWSSYNC]

2.5.12.4 Requirements

The client knows the FID, as described in [MS-OXCDATA] section 2.2.1.1, or path for the folder to synchronize.

2.5.12.5 Protocol-Specific Details

Using remote operations (ROPs)

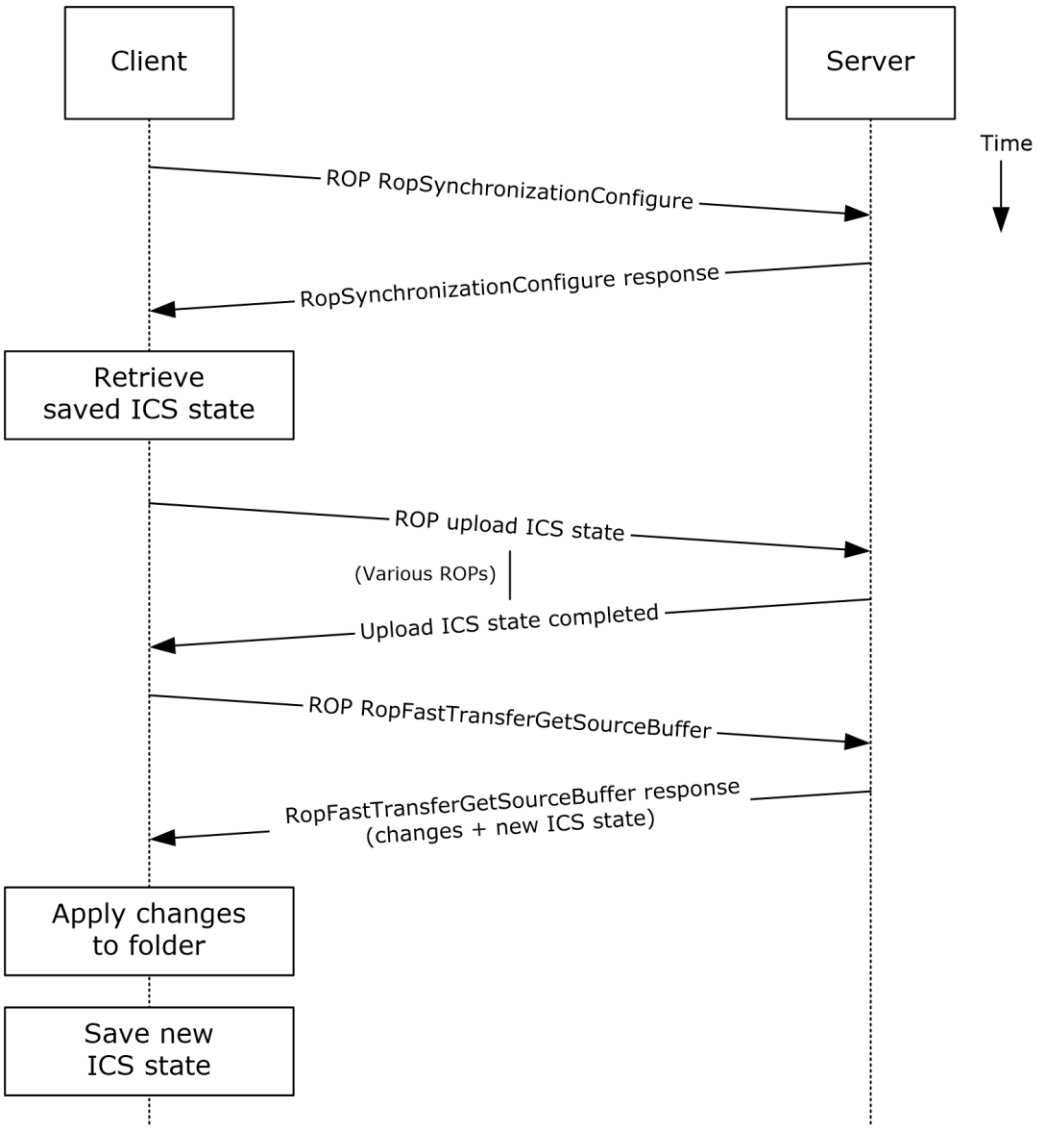

Figure 33: Synchronizing items by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client opens the specified folder per the use case described in section 2.5.9.
2. The client issues a **RopSynchronizationConfigure** ROP request ([MS-OXCROPS] section 2.2.13.1) by using the handle to the folder to initiate the synchronization.
3. The Exchange server responds with a handle to the synchronization.
4. The client uses a series of **RopSynchronizationUploadStateStreamBegin** ([MS-OXCROPS] section 2.2.13.9), **RopSynchronizationUploadStateStreamContinue** ([MS-OXCROPS] section 2.2.13.10), and **RopSynchronizationUploadStateStreamEnd** ([MS-OXCROPS] section 2.2.13.11) ROP requests to upload the client's **Incremental Change Synchronization (ICS)** state, as described in [MS-OXCFXICS], to the Exchange server, which prepares the list of changed items for the client to download based on the uploaded **ICS state**.
5. The client issues a **RopFastTransferSourceGetBuffer** ROP request ([MS-OXCROPS] section 2.2.12.3) to obtain the list of changes from the Exchange server.
6. The Exchange server responds to the request with the full list of changes as well as the new ICS state.
7. The client applies the changes to the folder items and saves the new ICS state for subsequent synchronizations.

Using Exchange ActiveSync

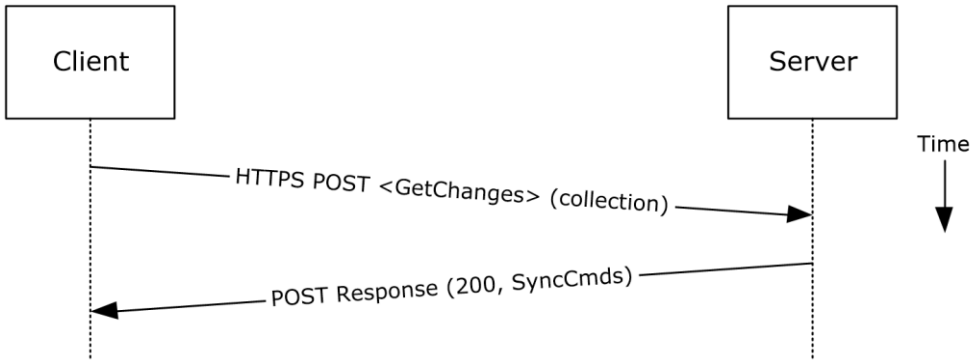

Figure 34: Synchronizing items by using Exchange ActiveSync

1. The client issues a **Sync** command request, as described in [MS-ASCMD] section 2.2.2.21, with a **GetChanges** element, as described in [MS-ASCMD] section 2.2.3.83, to the Exchange server requesting a list of all changes that have occurred in the specified collection (folder) since the last successful synchronization.
2. The Exchange server responds with a series of **Sync** command responses, as described in [MS-ASCMD] section 2.2.2.21, that the client processes to synchronize with the Exchange server.

Using Exchange Web Services

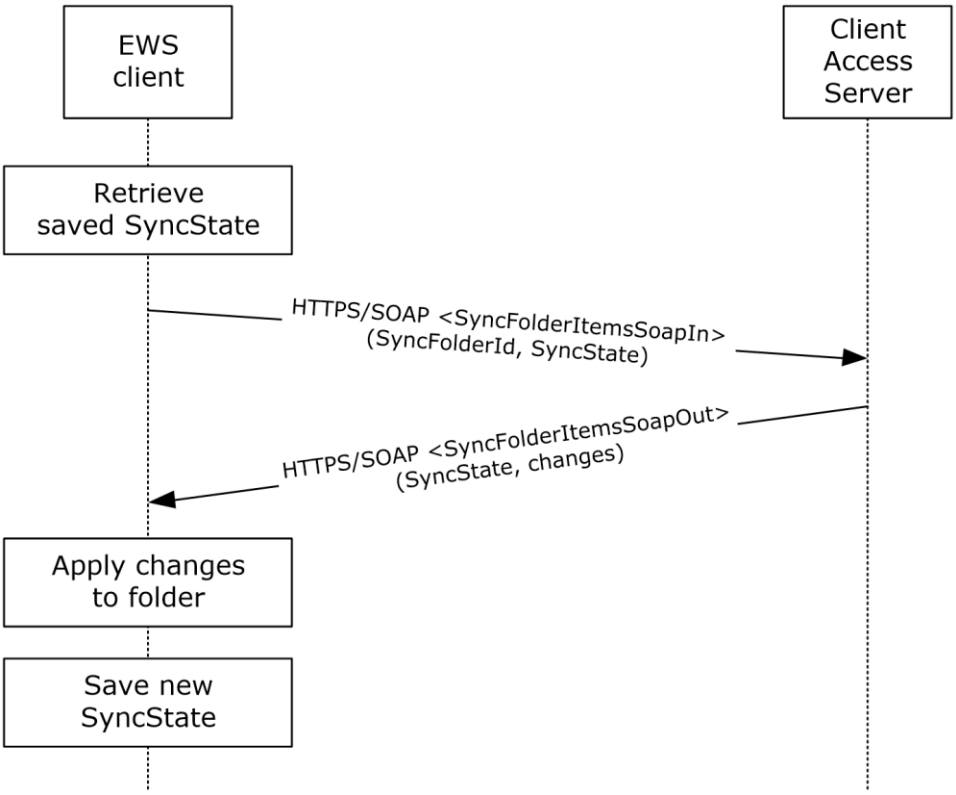

Figure 35: Synchronizing items by using Exchange Web Services

1. The client uses the HTTPS/SOAP **SyncFolderItemsSoapIn** request message, as described in [MS-OXWSSYNC] section 3.1.4.2.1.1, to synchronize the changes to items in the folder specified in the **SyncFolderId** element. The **SyncState** element provides the state of the last synchronization and acts as a starting marker for synchronizing new changes.

2. The Exchange Client Access server responds with a **SyncFolderItemsSoapOut** response message, as described in [MS-OXWSSYNC] section 3.1.4.2.1.2, which includes a **ResponseCode** element, a new **SyncState** marker, and a list of changes.

3. The client applies changes to the folder by using information in the **SyncFolderItemsSoapOut** response, and saves the **SyncState** value that was returned in the **SyncFolderItemsSoapOut** response.

2.5.13 Register For and Receive Notifications

2.5.13.1 Synopsis

The register for and receive notifications use case describes how a client registers for server notifications for a particular folder and how to determine whether a notification event has occurred using a passive mechanism.

2.5.13.2 Builds on Use Case(s)

Log on to a mailbox, as described in section 2.5.2

2.5.13.3 References

- [MS-OXCROPS]
- [MS-OXCNOTIF]
- [MS-ASCMD]
- [MS-OXWSNTIF]

2.5.13.4 Requirements

The client knows the FID, as described in [MS-OXCDATA] section 2.2.1.1, or path for the folder that will be used to register server notifications.

2.5.13.5 Protocol-Specific Details

Using remote operations (ROPs)

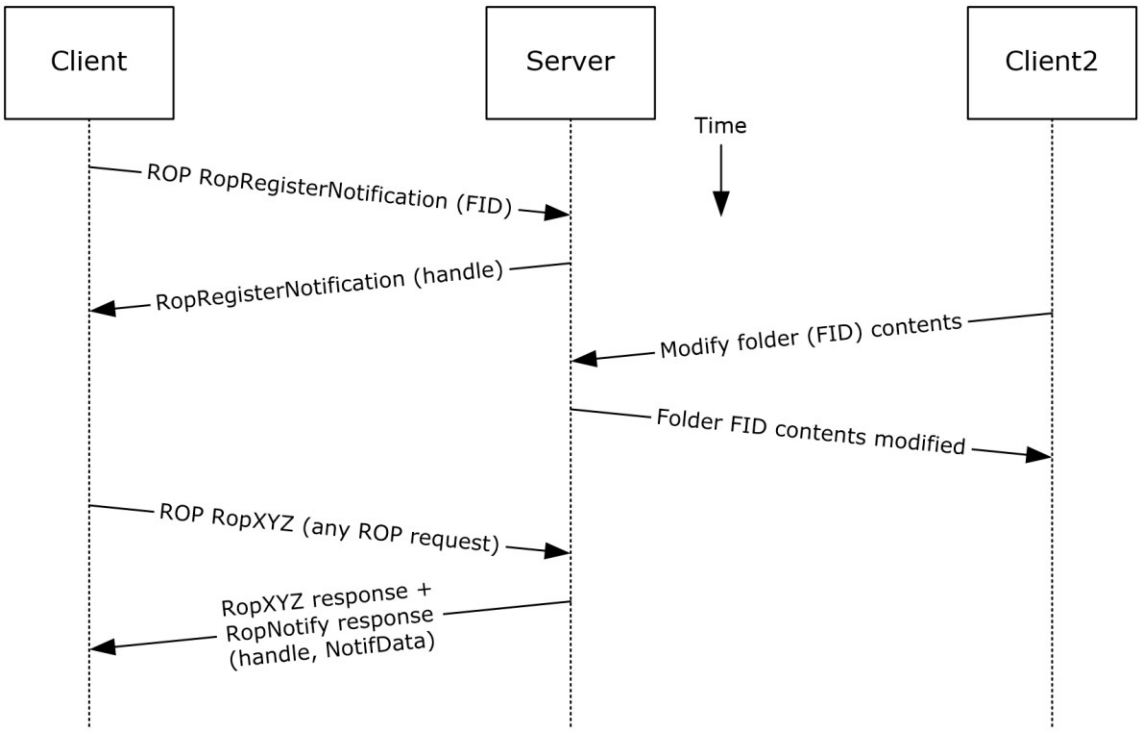

Figure 36: Registering for and receiving notifications by using ROPs

Note The ROPs are sent via RPC or MAPI extensions for HTTP. Several ROPs can be batched into a single request, as described in [MS-OXCROPS].

1. The client logs on to the mailbox per the use case described in section 2.5.2.
2. The client issues a **RopRegisterNotification** ROP request ([MS-OXCROPS] section 2.2.14.1) to register for notifications. The FID, as described in [MS-OXCDATA] section 2.2.1.1, of the folder to be monitored is included in the request.
3. The Exchange server responds with a handle to the notification.

4. The next time that the client issues a ROP request to the Exchange server, the server adds a **RopNotify** ROP response ([MS-OXCROPS] section 2.2.14.2) and a result code of **RopPending** is returned for the overall ROP response.

5. The client examines the information in the **RopNotify** ROP response and triggers an **event** on the client. For example, a UI notification is triggered in response to a link that was added to the body of a meeting invite.

Using Exchange ActiveSync

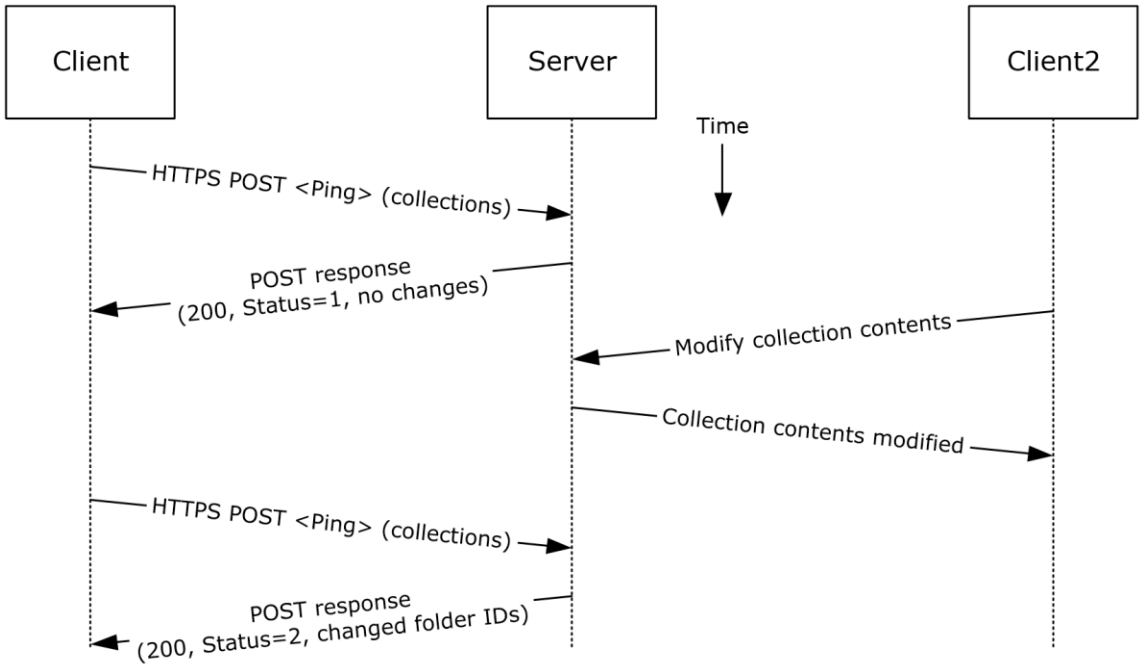

Figure 37: Registering for and receiving notifications by using Exchange ActiveSync

1. The client issues a ping request to the Exchange server to check for events that have occurred in the specified list of containers (folders).

2. If no events have occurred since the last ping, the Exchange server returns HTTP status code 200 (OK) and a status code of 1, as described in [MS-ASCMD]. If events have occurred, the server returns HTTP status code 200 (OK) and a status code of 2, followed by a list of FIDs that need to be synchronized.

Using Exchange Web Services

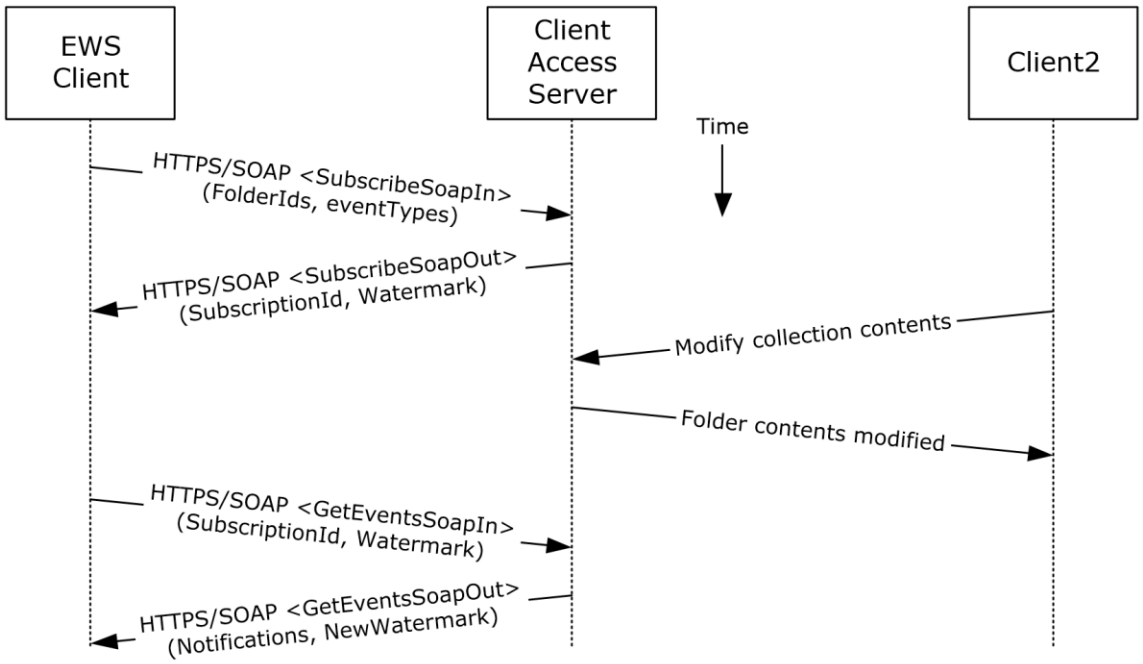

Figure 38: Registering for and receiving notifications by using Exchange Web Services

1. The client uses the HTTPS/SOAP **SubscribeSoapIn** request, as described in [MS-OXWSNTIF] section 3.1.4.3.1.1, to subscribe to pull notifications for the types of events indicated in the **EventTypes** element on a list of folders specified by the **FolderIds** element.

2. The Client Access server responds with a **SubscribeSoapOut** response, as described in [MS-OXWSNTIF] section 3.1.4.3.1.2, which includes a **ResponseCode** element, a **SubscriptionId** element that identifies this subscription, and a **Watermark** element that indicates the current notification state.

3. A second client modifies the contents of a folder that has an active subscription. This causes the server to create a pending notification associated with the subscription.

4. The first client uses the HTTPS/SOAP **GetEventsSoapIn** request, as described in [MS-OXWSNTIF] section 3.1.4.1.1.1, to query the server for any new notifications. The value of the **Watermark** element allows the server to determine which new notifications, if any, to return to the client.

5. The Exchange Client Access server responds with a **GetEventsSoapOut** response, as described in [MS-OXWSNTIF] section 3.1.4.1.1.2, which includes a **Notifications** element listing the notifications that were pending since the value of the **Watermark** element specified by the client. A **Watermark** element is also returned to the client to indicate the new notification state for the next **GetEventsSoapIn** call.

2.5.14 Provision a Mobile Client Device

2.5.14.1 Synopsis

The provision a mobile client device use case describes the multiphase provisioning process for a mobile client device using Exchange ActiveSync.

Note This use case is specific to Exchange ActiveSync.

2.5.14.2 Build on Use Case(s)

None.

2.5.14.3 References

- [MS-ASHTTP]
- [MS-ASCMD]
- [MS-ASPROV]

2.5.14.4 Requirements

The mobile client is configured to communicate with an Exchange ActiveSync service provider.

2.5.14.5 Protocol-Specific Details

Using Exchange ActiveSync

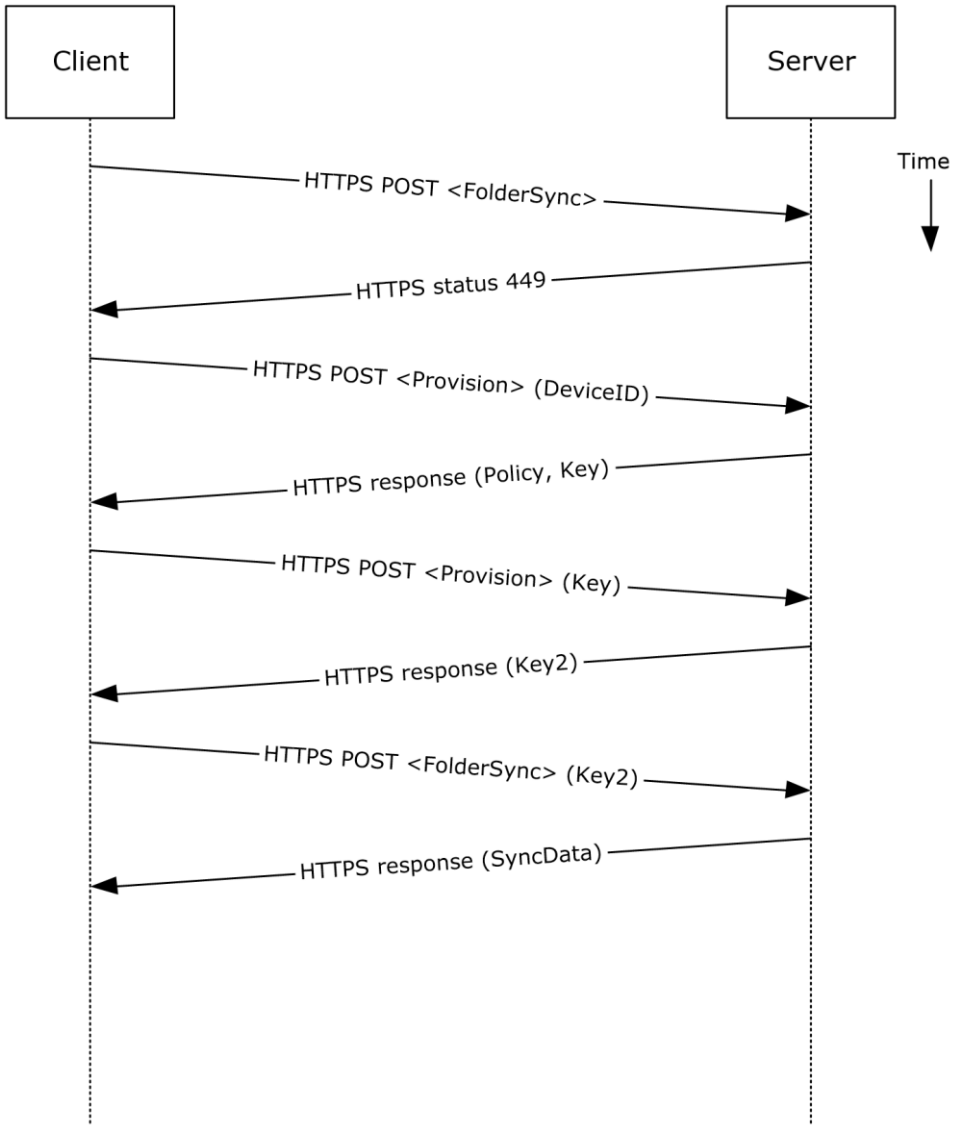

Figure 39: Provisioning a mobile client device by using Exchange ActiveSync

1. A client without a current policy attempts to synchronize with an Exchange server (in this case, by issuing a **FolderSync** command request, as described in [MS-ASCMD] section 2.2.2.5).
2. The Exchange server replies with HTTP status code 449 (Need Provisioning) to indicate that the client's policy is not current.
3. The client issues a **Provision** command request ([MS-ASCMD] section 2.2.2.14) to download the latest policy from the Exchange server, referencing its own device ID for identification.
4. The Exchange server responds with an **XML** document representation of the current policy plus a temporary policy key.
5. The client issues a second **Provision** command request to acknowledge receipt of policy settings in the initial server response.
6. The Exchange server responds with a policy key that can be used in subsequent command requests.

7. The client issues a **FolderSync** command request to the Exchange server, including the most recent policy key in the HTTP headers, as described in [MS-ASPROV].

8. The Exchange server acknowledges the policy key and returns the **FolderSync** information.

2.6 Versioning, Capability Negotiation, and Extensibility

Several mechanisms exist to allow the exchange of version information between a client and the Exchange server. The exact mechanism used depends on the protocol used by the requesting client. The following sections outline several of the most commonly used mechanisms.

2.6.1 Version Negotiation Using RPC

When a client attempts to connect with an Exchange server via the RPC protocol, as described in [MS-OXCRPC], the client sends its version as part of the **EcDoConnectEx** connection request. If the server returns a value of **Success** (0x00000000) or **ecVersionMismatch** (0x80040110) to the connection request, the server also returns two version numbers: the server version, as described by the **rgwServerVersion** parameter, and the "best" version, as described by the **rgwBestVersion** parameter. The **rgwServerVersion** parameter contains the actual version of the server. The value of the **rgwBestVersion** parameter depends on whether **Success** or **ecVersionMismatch** was returned by the **EcDoConnectEx** method. If the server returns a value of **ecVersionMismatch**, the **rgwBestVersion** parameter contains the minimal client version the client needs to support to connect to the server. If the server returned a value of **Success**, the **rgwBestVersion** parameter contains the client version provided by the client during the request. If the server returns any other error code, the value of the **rgwBestVersion** parameter is undefined.

Upon exchange of this information, the client can determine the level of functionality offered by that Exchange server and the most appropriate functionality to provide to the end user.

The server cannot perform any client protocol version negotiation.

2.6.2 Version Negotiation Using MAPI Extensions for HTTP

When a client attempts to connect with an Exchange server via the MAPI extensions for HTTP, as described in [MS-OXCMAPIHTTP], the client sends its version in the **X-ClientApplication** header of the **POST** request. The server returns its own version back to the client in the **X-ServerApplication** header of the **POST** response.

Upon exchange of this information, the client can determine the level of functionality offered by that Exchange server and the most appropriate functionality to provide to the end user.

The server cannot perform any client protocol version negotiation.

2.6.3 Version Negotiation Using Exchange Web Services

Web services clients contact the Exchange server by using the industry-standard SOAP 1.1 over HTTP protocol, as described in [SOAP1.1] and [RFC2616]. The server responds with a **ServerVersionInfo** XML structure ([MS-OXWSCDATA] section 2.2.3.12) with the server version details in the SOAP response header. Based on this information, the client can determine the server's capabilities.

2.6.4 Version Negotiation Using Exchange ActiveSync

An ActiveSync client can indicate its version via the **MS-ASProtocolVersion** header or as part of a query string encoded with base64 encoding in the HTTP request URI, as described in [MS-ASHTTP]. Similar to WebDAV, the client can use the HTTP **OPTIONS** command, as described in [MS-ASHTTP], to obtain the server version and a comprehensive list of supported ActiveSync versions and keywords.

2.7 Error Handling

The following sections provide an overview of the impact of failure for each element in the figure in section 2.1. In addition, the following sections describe the impact of DNS or the directory service becoming unavailable.

2.7.1 SMTP

SMTP is the key gateway for receiving inbound mail from external servers and non-RPC clients. If inbound SMTP is unavailable, messages cannot be received from other e-mail servers or from SMTP-based e-mail clients. In addition, Unified Messaging relies on SMTP to inject new messages, so some of its functionality can be impacted.

If outbound SMTP is unavailable, the Exchange server cannot route outbound e-mail. Messages will accumulate in the delivery queue, and in extreme cases, this can cause inbound SMTP to reject incoming messages.

2.7.2 Remote Operations (ROPs)

Due to Microsoft Exchange-specific implementation details, the remote operations (ROPs) are at the heart of the Exchange message store. Almost all other protocols are constructed around the ROPs and are dependent on it. If the ROPs become unavailable, then other protocols can experience service outages related to accessing storage items.

2.7.3 Exchange Web Services

If Exchange Web Services becomes unavailable, Web services clients will not be able to access the Exchange server.

2.7.4 POP3

If POP3 becomes unavailable, e-mail clients will not be able to access their mailbox by using POP3.

2.7.5 IMAP4

If IMAP4 becomes unavailable, e-mail clients will not be able to access their mailbox by using IMAP4.

2.7.6 WebDAV

If WebDAV becomes unavailable, e-mail clients will not be able to communicate with the Exchange server by using WebDAV.

2.7.7 Address Book

If the address book server becomes unavailable, operations related to address book lookup and recipient resolution will be affected. However, clients using an offline address book (OAB), as described in [MS-OXOAB], will still be able to perform recipient lookups and resolutions for the contacts (2) listed in the local OAB.

2.7.8 Unified Messaging

If Unified Messaging becomes unavailable, Unified Messaging-specific features will not be available.

2.7.9 Exchange ActiveSync

Mobile clients using Exchange ActiveSync will be impacted if it becomes unavailable. However, many clients capable of using Exchange ActiveSync are able to take alternate routes (for example, Web Access) to access their mailboxes until Exchange ActiveSync is available again.

2.7.10 DNS

If DNS becomes unavailable, outbound e-mail routing will be impacted because the remote domain address cannot be resolved. Since Microsoft Exchange Autodiscover, as described in [MS-OXDISCO], relies on DNS as an alternate mechanism, this will cause Autodiscover problems when the system is configured to use DNS as a discovery mechanism.

2.7.11 Directory Service

The directory service is responsible for storing mail recipient information and is a primary mechanism for Autodiscover, as described in [MS-OXDISCO]. If the directory service becomes unavailable, Autodiscover will be unavailable.

2.8 Coherency Requirements

This system has no special coherency requirements.

2.9 Security

For a comprehensive and current discussion about security and protection for Microsoft Exchange, see [MSFT-SAP].

2.10 Additional Considerations

There are no additional considerations.

3 Examples

Each of the following examples illustrate how one or more of the use cases described in section 2.5 can be combined to achieve specific end-user results. The examples show how the Microsoft Exchange Server system can be used for specific messaging tasks that map to real-life end-user scenarios. These examples are not meant to be exhaustive. However, many of these examples, once understood, can be applied to other similar real-life scenarios.

To reduce the complexity of the examples, each example is demonstrated using the protocol (for example, RPC or Exchange ActiveSync) that is most commonly used for the scenario in that example. In some cases, the same outcome can be achieved using other supported protocols as indicated in the use cases section, but these alternate methods will not be included in the example.

The following system-level examples are provided in this document:

- Display the most recent message in the inbox
- Compose and send an e-mail message with an attachment
- Set up and display new mail notifications
- Create an appointment request using free-busy data
- Provision and synchronize a mobile client device for the first time

3.1 Example 1: Display the Most Recent Message in the Inbox

3.1.1 Synopsis

This example illustrates how the Microsoft Exchange system can be used to access an Inbox folder and display the newest message in it.

Note This example uses the RPC protocol.

3.1.2 Use Cases

- Server information discovery (section 2.5.1)
- Log on to a mailbox (section 2.5.2)
- Open a folder (section 2.5.9)
- Find items in a folder that match search criteria (section 2.5.10)

3.1.3 Entities Involved

- An Exchange server (server)
- An RPC-enabled client (client)

3.1.4 Preconditions

- The Exchange server is available to accept client requests.
- The client has credentials for a valid mailbox account.

3.1.5 Details

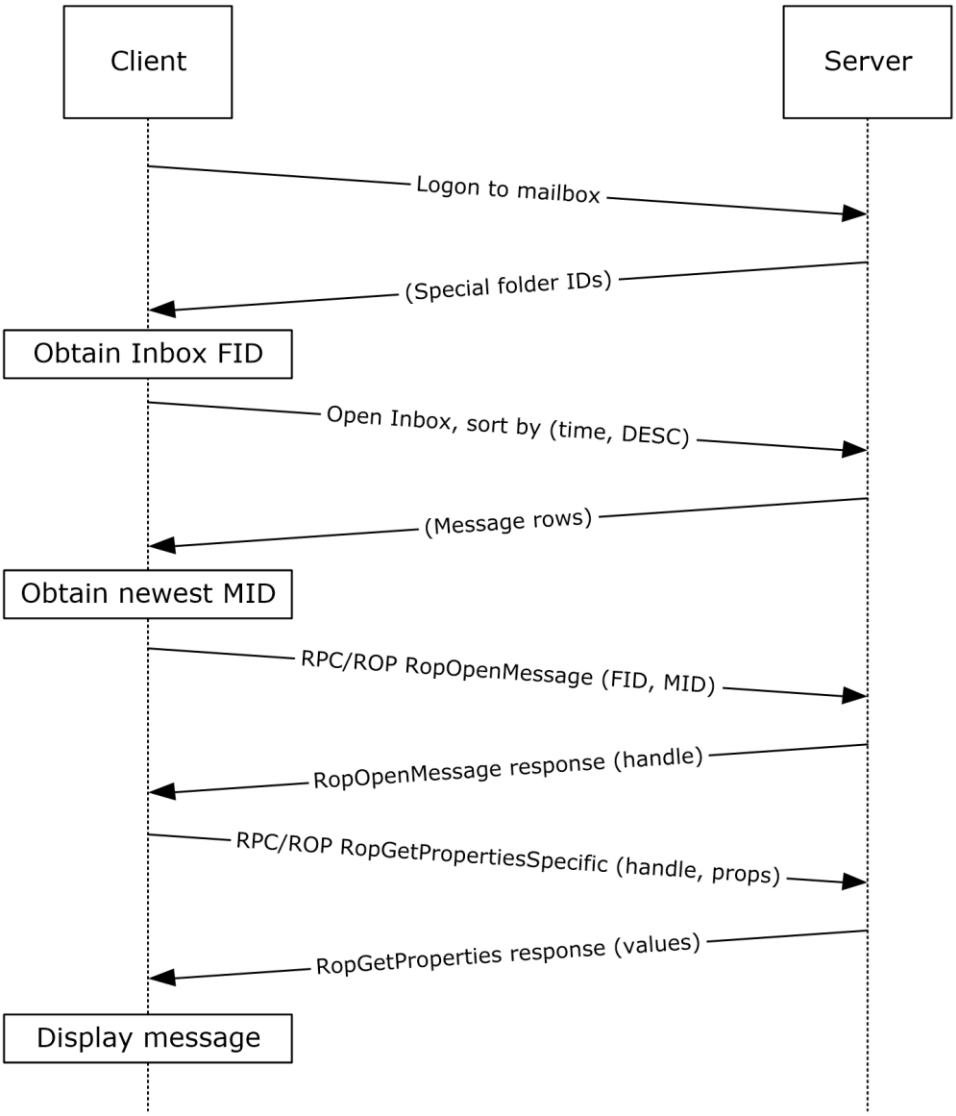

Figure 40: Display message steps

1. The client logs on to the mailbox per the use case described in section 2.5.2.
2. The client obtains the FID for the Inbox folder from the server as a result of successful logon.
3. Based on the use case described in section 2.5.10, the client queries the message rows in the Inbox folder and sorts the rows by message date (the **PidTagMessageDeliveryTime** property, as described in [MS-OXOMSG] section 2.2.3.9) in descending order. The resulting table contains a column for the MID (the **PidTagMid** property, as described in [MS-OXCFXICS] section 2.2.1.2.1).
4. The first row in the Inbox folder contents table contains information about the newest message. The client extracts the MID of the newest message from the **PidTagMid** property column.

5. The client issues a **RopOpenMessage** ROP request ([MS-OXCROPS] section 2.2.6.1) with the inbox FID and the MID returned. The Exchange server returns a handle to the opened message.

6. The client issues a **RopGetPropertiesSpecific** ROP request ([MS-OXCROPS] section 2.2.8.3) with the message handle to retrieve the body of the message (the **PidTagBody** property, as described in [MS-OXCMSG] section 2.2.1.56.1) and other properties required for displaying the message.

7. The client displays the message using the property values returned by the Exchange server.

3.2 Example 2: Compose and Send an E-Mail Message with an Attachment

3.2.1 Synopsis

This example illustrates how the Microsoft Exchange system can enable an end user to compose and send an e-mail message with an attachment.

Note This example uses the RPC protocol.

3.2.2 Use Cases

- Server information discovery (section 2.5.1)
- Log on to a mailbox (section 2.5.2)
- Create a message (section 2.5.3)
- Add an attachment (section 2.5.5)
- Resolve a recipient from an address book (section 2.5.6)
- Send a message (section 2.5.7)

3.2.3 Entities Involved

- An Exchange server (server)
- An RPC-enabled client (client)

3.2.4 Preconditions

- The Exchange server is available to accept client requests.
- The client has credentials for a valid mailbox account.
- The client has established a **Drafts folder** to store unsent messages.
- The intended mail recipient is listed in an address book.

3.2.5 Details

1. The client logs on to the mailbox per the use case described in section 2.5.2.
2. The client creates a new message in the drafts folder per the use case described in section 2.5.3.

3. An end user enters the name of the message recipient by using the client user interface and chooses to resolve the recipient.

4. The client resolves the recipient per the use case described in section 2.5.6. The Exchange server returns the successful matches (or none if there are no matches).

5. If the Exchange server returns more than a single match, the end user is presented with a list of matches and is asked to select the correct recipient from the list. This is known as **ambiguous name resolution (ANR)**. For more information about ANR, see [MS-NSPI].

6. The end user adds an attachment to the message.

7. The client loads the attachment data and creates a new attachment per the use case described in section 2.5.5.

8. The end user enters a body for the message and sends the message.

9. The client submits the message to the Exchange server per the use case described in section 2.5.7.

3.3 Example 3: Set Up and Display New Mail Notifications

3.3.1 Synopsis

This example illustrates how a client can use the Microsoft Exchange system to register itself for new mail notifications.

Note This example uses the RPC protocol.

3.3.2 Use Cases

- Server information discovery (section 2.5.1)
- Log on to a mailbox (section 2.5.2)
- Register for and receive notifications (section 2.5.13)

3.3.3 Entities Involved

- An Exchange server (server)
- An RPC-enabled client (client)

3.3.4 Preconditions

- The Exchange server is available to accept client requests.
- The client has credentials for a valid mailbox account.

3.3.5 Details

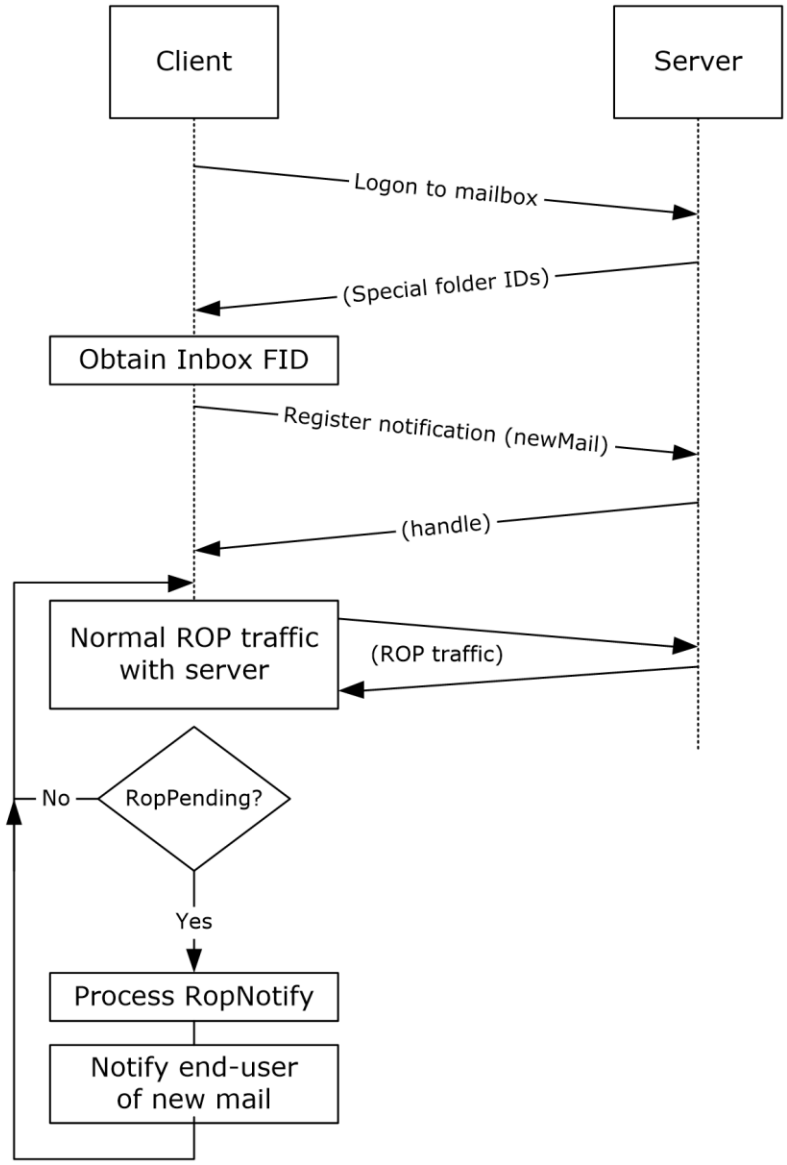

Figure 41: Setting up and displaying new mail notifications

1. The client logs on to the mailbox per the use case described in section 2.5.2.
2. The client obtains the FID for the Inbox folder from the Exchange server as a result of successful logon.
3. The client registers for notifications for new mail, as described in [MS-OXCNOTIF], by referencing the Inbox folder's FID per the use case described in section 2.5.13.
4. The client continuously monitors for **RopPending** ROP ([MS-OXCROPS] section 2.2.14.3) result codes from ROP responses as well as **RopNotify** ROP responses ([MS-OXCROPS] section 2.2.14.2) in the ROP response payload for notifications. The client will take appropriate action (for example, play a sound) when a new mail notification is received.

3.4 Example 4: Create an Appointment Request Using Free-Busy Data

3.4.1 Synopsis

This example illustrates how the Microsoft Exchange system can be used to enable an end user to examine an invitee's free-busy data and send out an appointment request.

Note This example uses the RPC protocol.

3.4.2 Use Cases

- Server discovery information (section 2.5.1)
- Log on to a mailbox (section 2.5.2)
- Create a message (section 2.5.3)
- Create a strongly typed message (section 2.5.4)
- Resolve a recipient from an address book (section 2.5.6)
- Send a message (section 2.5.7)
- Open a folder (section 2.5.9)
- Find items in a folder that match search criteria (section 2.5.10)

3.4.3 Entities Involved

- An Exchange server (server)
- An RPC-enabled client (client)
- A valid mailbox account to receive the appointment request (invitee)

3.4.4 Preconditions

- The Exchange server is available to accept client requests.
- The client has credentials for a valid mailbox account with access to the public folders, as described in [MS-OXCSTOR], and a drafts folder to store unsent messages.
- The invitee is listed in an address book.

3.4.5 Details

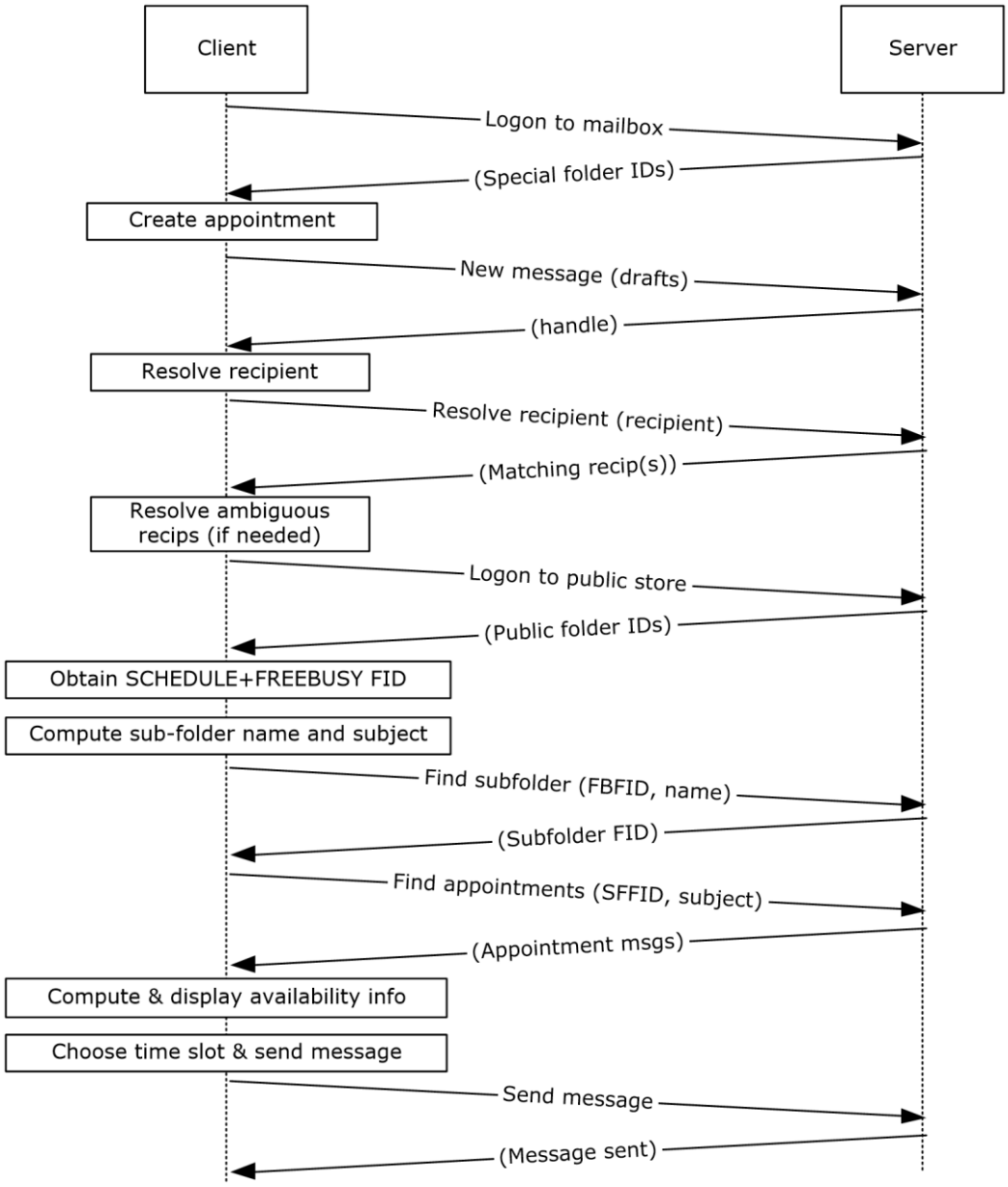

Figure 42: Creating an appointment using free-busy data

1. The client logs on to the mailbox per the use case described in section 2.5.2.
2. An end user creates a new appointment.
3. The client creates a new message in the drafts folder per the use case described in section 2.5.3. The Exchange server returns a handle to the new message.
4. The end user enters the name of the invitee and chooses to resolve the recipient.
5. The client resolves the recipient per the use case described in section 2.5.6. The server returns the successful matches (or none if there are no matches).

6. If the Exchange server returns more than a single match, the end user is presented with a list of matches and is asked to select the correct recipient from the list.

7. The client logs on to the public message store, as described in [MS-OXCSTOR] (using steps similar to the use case described in section 2.5.2).

8. The Exchange server returns a list of IDs for special folders upon successful logon. The client obtains the FID of the "SCHEDULE+FREEBUSY" folder, as described in [MS-OXOPFFB], from the list of special FIDs.

9. Based on the address book information of the resolved invitee, the client computes the name of the subfolder that contains the invitee's free-busy information and the subject for the appointment message, as described in [MS-OXOPFFB].

10. The client performs a folder search in the "SCHEDULE+FREEBUSY" public folder to find the FID of the subfolder that matches the computed subfolder name per the use case described in section 2.5.9.

11. The client performs another folder search in the subfolder to find all appointment messages whose subject (the **PidTagSubject** property, as described in [MS-OXPROPS] section 2.1030) matches the computed subject in step 9 per the use case described in section 2.5.9.

12. The client scans the appointment messages to calculate the invitee's availability and presents the availability to the end user.

13. The end user chooses a time slot based on the free-busy information.

14. The end-user enters some message text and sends the invite.

15. The client submits the message per the use case described in section 2.5.7.

3.5 Example 5: Provision and Synchronize a Mobile Client Device for the First Time

3.5.1 Synopsis

This example illustrates the process by which a mobile client device is provisioned and synchronized for the first time.

Note This example uses the Exchange ActiveSync protocol.

3.5.2 Use Cases

- Synchronize item(s) (section 2.5.12)
- Provision a mobile device (section 2.5.14)

3.5.3 Entities Involved

- An Exchange server (server)
- An Exchange ActiveSync–enabled mobile client device (client)

3.5.4 Preconditions

- The Exchange server is available to accept client requests.
- The client has credentials for a valid mailbox account.

3.5.5 Details

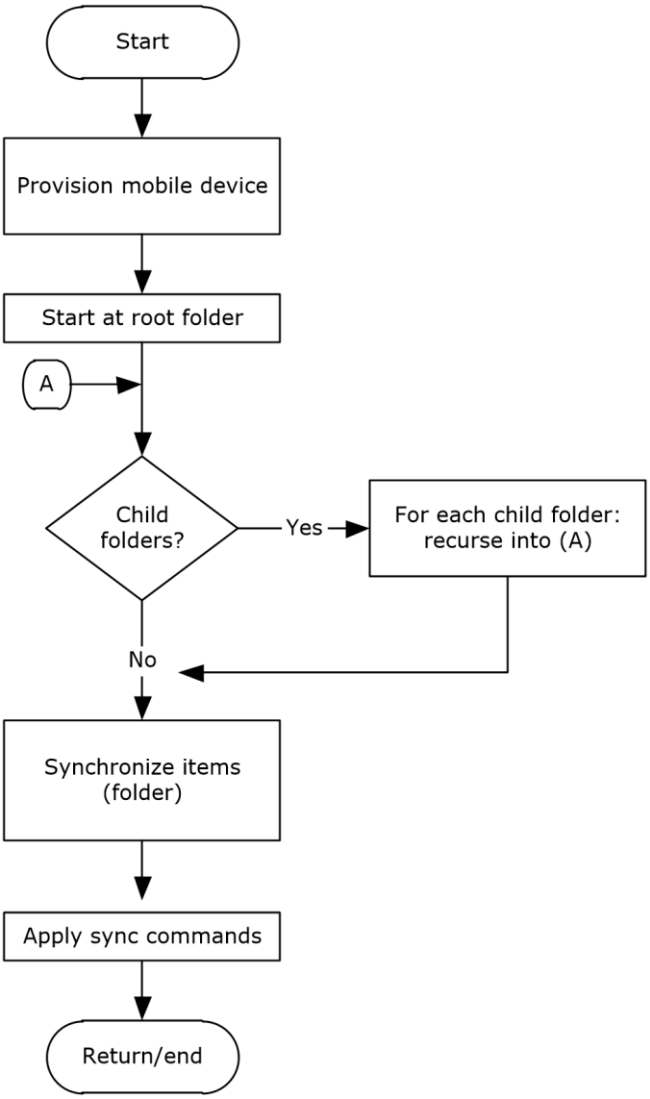

Figure 43: Provisioning and synchronizing a mobile device

1. The client undergoes the provisioning process per the use case described in section 2.5.14 to obtain the current policy and policy key. At this point, the **FolderSync** command, as described in [MS-ASCMD] section 2.2.2.5, would have caused the folder hierarchy to be created on the client. However, the contents in the folders are not synchronized yet.

2. The client recursively walks the folder hierarchy and performs a per-folder synchronization to download the folder contents per the use case described in section 2.5.12.

4 Microsoft Implementations

There are no variations in the behavior of the Microsoft Exchange Server system in different versions of Microsoft Exchange beyond those described in the specifications of the protocols supported by the system, as listed in section 2.2.

The information in this specification is applicable to the following versions of Microsoft Exchange and Microsoft Outlook:

- Microsoft Exchange Server 2003
- Microsoft Exchange Server 2007
- Microsoft Exchange Server 2010
- Microsoft Exchange Server 2013
- Microsoft Exchange Server 2016
- Microsoft Office Outlook 2003
- Microsoft Office Outlook 2007
- Microsoft Outlook 2010
- Microsoft Outlook 2013
- Microsoft Outlook 2016

Exceptions, if any, are noted in the following section.

4.1 Product Behavior

5 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Added new protocol MS-OXWSOLPS to the Web Services diagram.	N	New content added.
2.2.10 Web Service Protocols	Added new protocol MS-OXWSOLPS to the table and the diagram.	N	New content added.

6 Index

E

S

T

U

V

W