[MS-OXPROTO]:
Exchange Server Protocols System Overview

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Updated references to reflect date of initial release.
09/03/2008	1.02		Changed title of document.
12/03/2008	1.03		Revised and edited technical content.
02/04/2009	1.04		Revised and edited technical content.
03/04/2009	1.05		Revised and edited technical content.
04/10/2009	2.0		Updated technical content for new product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	4.0.0	None	Version 4.0.0 release
05/05/2010	5.0.0	Major	Updated and revised the technical content.
08/04/2010	5.1	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

This document provides a system overview for the protocols in the Exchange Server Protocols system. It is intended for use in conjunction with the Microsoft protocol technical specifications, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A protocol system document does not require the use of Microsoft programming tools or programming environments in order to implement the protocols in the system. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Sections 2.1 and 2.2 list definitions, protocol stacks, client configurations, and functional areas referred to throughout the rest of this overview.

1.1 Glossary

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

Address Book object
Attachment object
base64 encoding
folder
Folder object
message
Message object
property (1)
public folder
search folder
special folder
Unified Messaging

1.2 References

[MS-OXGLOS] Microsoft Corporation, "Exchange Server Protocols Master Glossary", April 2008.

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

[MS-RPCH] Microsoft Corporation, "Remote Procedure Call over HTTP Protocol Specification", July 2006, http://msdn.microsoft.com/en-us/library/cc243950.aspx

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

[MSFT-SAP] TechNet, Microsoft Corporation, Microsoft Exchange Server 2007 Security and Protection, January, 2009, http://go.microsoft.com/fwlink/?LinkId=64880.

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

2 Functional Architecture

The Microsoft Exchange Server Protocols documentation set provides detailed technical specifications for Microsoft proprietary protocols (including extensions to industry-standard or other published protocols) that are used by Microsoft Exchange Server to communicate with other Microsoft products.

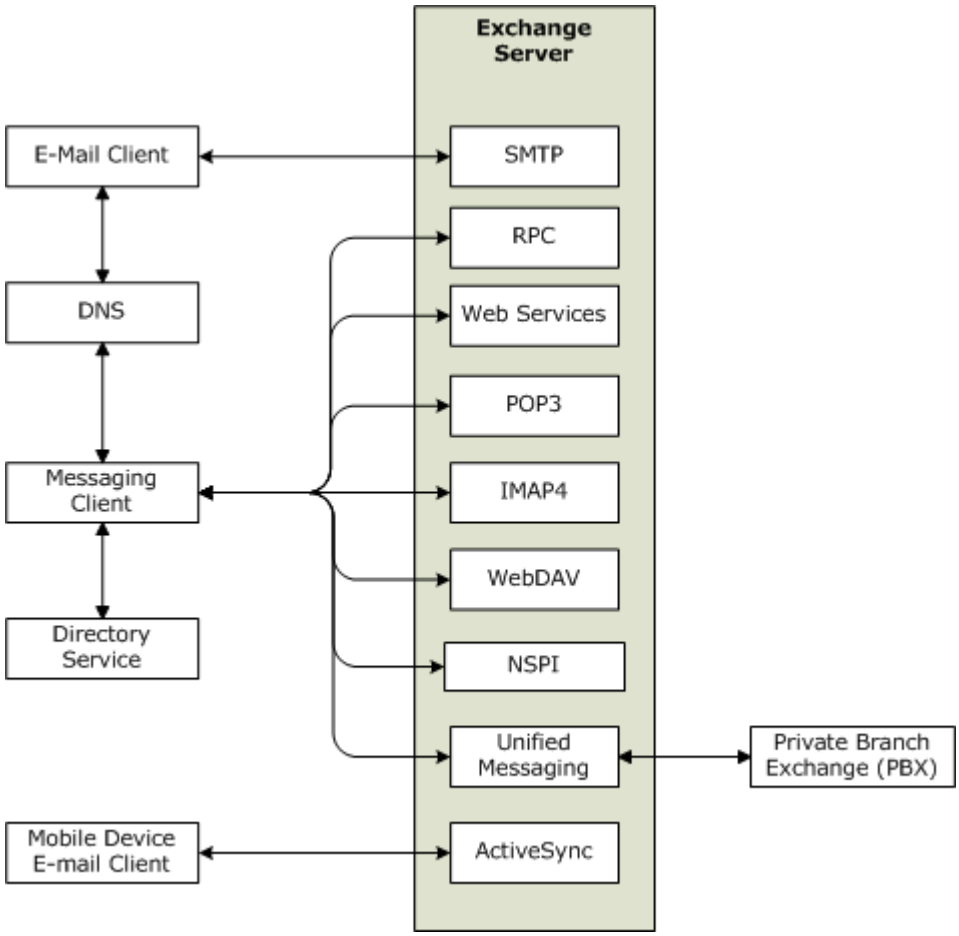

Figure 1:

2.1 System Overview

This section provides an overview of the system environment surrounding the Exchange Server system. Figure 1 above illustrates the Exchange Server system from a protocols perspective, where the server provides protocols for clients. The clients that interoperate with the server perform messaging tasks, and ancillary entities provide essential supporting services.

Exchange servers provide a rich set of interfaces with which clients can interoperate. Each protocol exposes a set of functionality that pertains to specific classes of operations. For example, SMTP, POP3, and IMAP4 constitute a set of Internet Standard protocols that simple e-mail clients use to send, retrieve, and manage e-mail; Web Services offers a standardized interface for middle-tier applications to build value-added services; WebDAV provides a set of interfaces that caters to

distributed authoring; and the RPC interface provides 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 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 a vast set of functionality that spans from basic storage to accessing, updating, and synchronizing address books, appointments, and shared folders. These specifications have been produced to capture the aggregate functionality of Exchange Server. This document provides an introductory framework that helps the reader to gain a broad understanding of the Exchange Server Protocols system and provides an organized roadmap for the reader to quickly locate further reading.

The contents of the Exchange Server Protocols documentation set have been organized in this document into a number of different functional groups, which are presented and described in section 2.2. However, it is important to understand two key functional components of Exchange Server, as they are not illustrated in Figure 1. An Exchange server can be regarded as having two functional components: an information store and a message processing system. The following subsections explain these functions in more detail.

2.1.1 Information Store

The information store component provides storage functionality for Exchange servers, as specified in [MS-OXCSTOR]. From a functional point of view, the information store is a hierarchical storage system consisting of **folders** and **messages**. The information 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.

2.2 Protocol Summary

The Exchange Server Protocol document set can be broken down into functional groups as illustrated in the following diagram.

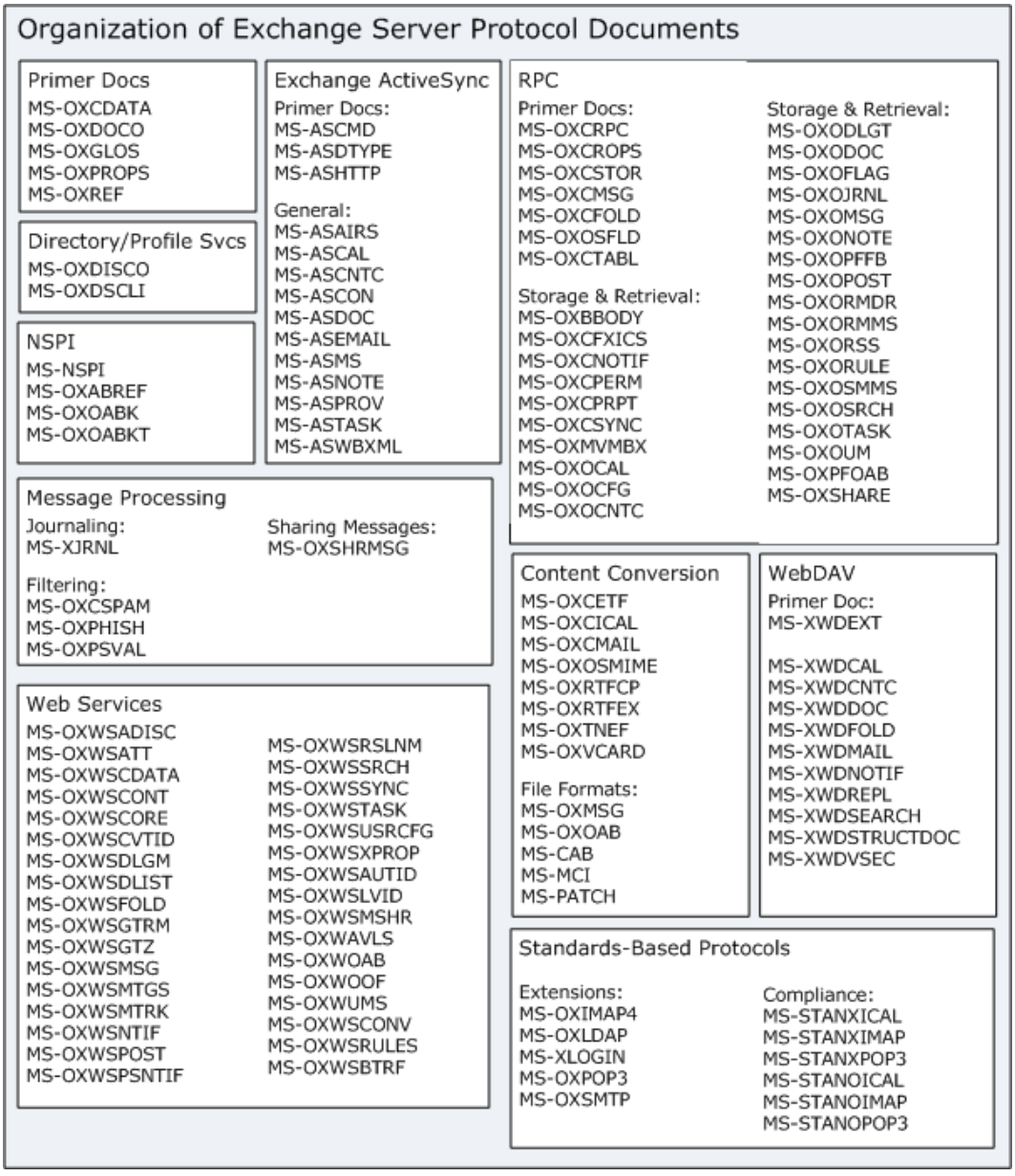

Figure 2:

2.2.1 Exchange Server Primer Protocols

The primer protocols contain data structures, **properties**, references, and document format information that is broadly useful.

Protocol Name	Description	Document Short Name
Data Structures	Specifies structures that are used by multiple protocol areas.	[MS-OXCDATA]

Protocol Name	Description	Document Short Name
Exchange Server Protocols Document Roadmap	Provides an overview of the specifications that are included in the Exchange Server Protocols documentation set.	[MS-OXDOCO]
Exchange Server Protocols Master Property List Specification	Contains the definition of each property that is used in the objects that are described by MS-OXO-prefixed documents.	[MS-OXPROPS]
Exchange Server Protocols Master Reference	Provides the list of references used in the Exchange Server Protocol documentation set.	[MS-OXREF]

2.2.2 RPC Primer/Storage and Retrieval Protocols

The RPC primer specifications describe how messaging information is organized and formatted for packaging and transmitting data between clients and servers. The storage and retrieval protocols describe the tasks and objects used to store and retrieve messages related to calendars, tasks, and personal contacts. The following figure illustrates the hierarchical relationships between the RPC storage and retrieval protocol specifications.

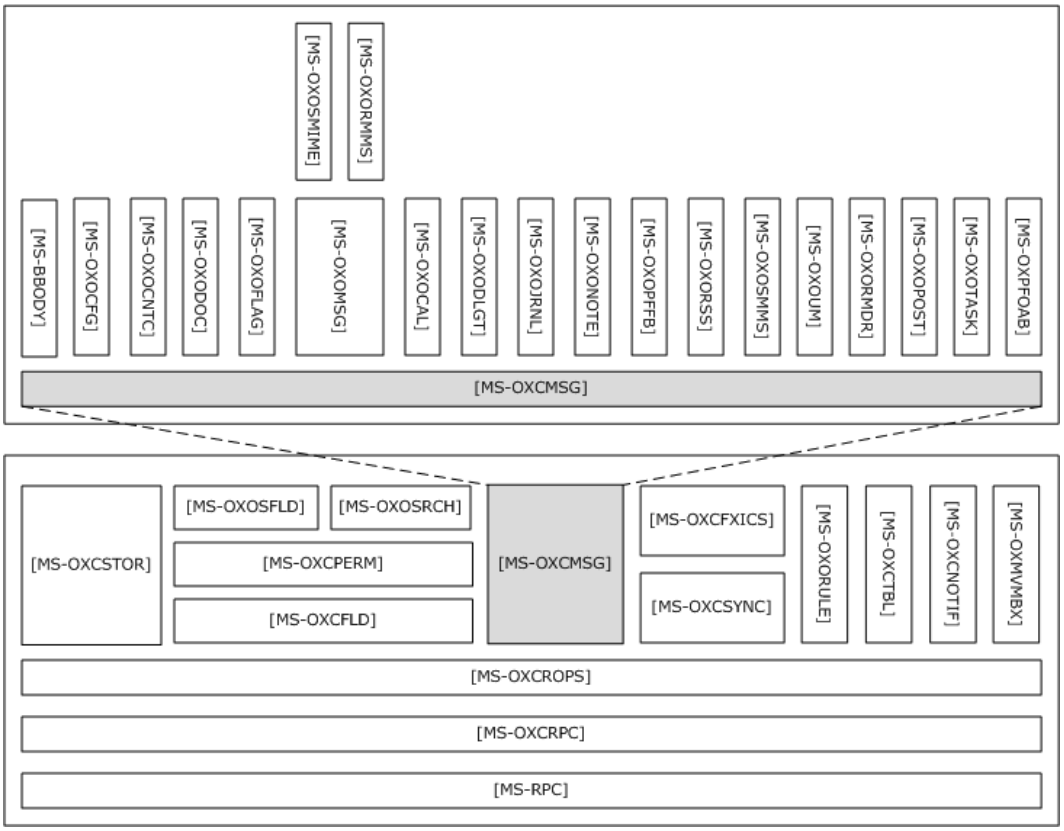

Figure 3:

2.2.2.1 RPC Primer Protocol Documents

Protocol Name	Description	Document Short Name
Wire Format	A stateful protocol that specifies how a client makes calls to servers containing messaging data. This protocol is a foundation upon which other protocols listed in the property and stream object protocol section are dependent.	[MS-OXCRPC]
Remote Operations (ROP) List and Encoding	Specifies the Remote Operations (ROP) List and Encoding Protocol, which is used by clients to access and modify mailbox information on servers.	[MS-OXCROPS]
Store Object	Specifies the properties and permissible operations for the core message store objects.	[MS-OXCSTOR]
Message and Attachment Object	Specifies the properties and permissible operations for the core **Message** and **Attachment objects**. Core Message objects form the basis for many higher-level objects such as appointments (see [MS-OXOCAL]) and contacts (see [MS-OXOCNTC]).	[MS-OXCMSG]
Folder Object	Specifies the properties and permissible operations for **Folder objects**.	[MS-OXCFOLD]
Special Folders	Specifies the properties and operations used to create and locate **special folders** in a mailbox using the Remote Operations (ROP) List and Encoding Protocol, as specified in [MS-OXCROPS].	[MS-OXOSFLD]
Table Object	Specifies the properties and permissible operations for the core table objects.	[MS-OXCTABL]

2.2.2.2 RPC Storage & Retrieval

Protocol Name	Description	Document Short Name
Best Body Retrieval	Specifies the mechanism for determining the best format of storing and retrieving message bodies.	[MS-OXBBODY]
Bulk Data Transfer	Specifies the order and data flow for transferring data between clients and servers.	[MS-OXCFXICS]
Core Notifications	Specifies the subscription and delivery mechanisms for push and polled notifications on changes.	[MS-OXCNOTIF]
Exchange Access and Operation Permissions	Specifies how folder permission lists stored on the server are retrieved.	[MS-OXCPERM]
Property and Stream Object	Specifies the properties and permissible operations for the core property and stream objects.	[MS-OXCPRPT]
Mailbox Synchronization	Specifies how Messaging object data is synchronized between servers and clients.	[MS-OXCSYNC]
Mailbox Migration	Describes the interaction between clients and servers when moving a mailbox to a new server.	[MS-OXMVMBX]

Protocol Name	Description	Document Short Name
Appointment and Meeting Object	Specifies properties and operations for appointment and meeting requests and responses (such as accept, decline, and counter-propose) using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG], and the E-Mail Object Protocol specified in [MS-OXOMSG].	[MS-OXOCAL]
Configuration Information	Specifies the location and properties of client and server configuration data, such as shared category lists and working hours, using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOCFG]
Contact Object	Specifies the properties and operations permissible on contacts and personal distribution lists using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOCNTC]
Delegate Access Configuration	Specifies connections to, and configuration of, mailboxes when acting on behalf of another user or resource. This specification also describes interactions with messages when acting on behalf of another user, as specified in [MS-OXOMSG], [MS-OXOTASK], and [MS-OXOCAL].	[MS-OXODLGT]
Document Object	Specifies the properties and operations permissible on documents using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXODOC]
Informational Flagging	Specifies the properties and operations related to flagging messages for follow-up using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOFLAG]
Journal Object	Specifies the properties and operations permissible on Journal objects using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOJRNL]
E-mail Object	Specifies the properties and operations permissible on e-mail messages using the Remote Operations (ROP) List and Encoding Protocol, as specified in [MS-OXCROPS].	[MS-OXOMSG]
Note Object	Specifies the properties and operations permissible on Note objects using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXONOTE]
Public Folder Availability	Specifies the format of free/busy data that describes the availability of a user or resource. This information is persisted in **public folders**. This data can be used to efficiently schedule meetings and/or provide presence information.	[MS-OXOPFFB]
Post Object	Specifies the properties and operations permissible on posts using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOPOST]
Reminder Settings	Specifies the properties and interaction model for reminders set on messages, as specified in [MS-OXCMSG].	[MS-OXORMDR]
Rights-Managed E-mail Object	Specifies the properties of rights-managed encoded messages using the E-Mail Object Protocol as specified in [MS-OXOMSG] as well as related use of the Rights Management Services (RMS) Protocol, as specified in [MS-RMPR].	[MS-OXORMMS]

Protocol Name	Description	Document Short Name
RSS Object	Specifies the properties and operations for representing RSS items using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXORSS]
E-mail Rules	Specifies the mechanism and properties for manipulating server-side rules via the Remote Operations (ROP) List and Encoding Protocol, as specified in [MS-OXCROPS], in order to act on incoming and outgoing e-mail messages.	[MS-OXORULE]
SMS and MMS Object	Specifies the properties and operations used to represent Short Message Service (SMS) and Multimedia Messaging Service (MMS) messages using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOSMMS]
Search Folder List Configuration	Specifies the properties and operations used to manipulate **search folder** list configuration using the Remote Operations (ROP) List and Encoding Protocol, as specified in [MS-OXCROPS].	[MS-OXOSRCH]
Task-Related Objects	Specifies the properties and operations for contacts and personal distribution lists using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOTASK]
Voice Mail and Fax Objects	Specifies the properties and operations for representing voice mail and fax messages using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].	[MS-OXOUM]
Offline Address Book (OAB) Public Folder Retrieval	Specifies the method of delivering OAB data to clients.	[MS-OXPFOAB]
Sharing Message Object	Specifies how to share mailbox folders between clients.	[MS-OXSHARE]

2.2.3 Content Conversion protocols

When clients and servers send or receive data in standards-based formats, they are expected to use the following protocols to convert from the standard-based formats into one or more of the Exchange Server Protocols supported formats.

Protocol Name	Description	Document Short Name
Enriched Text Format (ETF) Message Body Conversion	Specifies server behavior when dealing with a text/enriched message body.	[MS-OXCETF]
iCalendar to Appointment Object Conversion	Specifies how to convert between [RFC2445], [RFC2446], [RFC2447] and Appointment and Meeting objects, as specified in [MS-OXOCAL], in conjunction with [MS-OXCMAIL].	[MS-OXCICAL]
RFC2822 and MIME to E-mail Object Conversion	Specifies conversion from Internet standard e-mail conventions to Message objects as specified in [MS-OXOMSG].	[MS-OXCMAIL]

Protocol Name	Description	Document Short Name
S/MIME E-mail Object	Specifies the properties of secure MIME signed and encrypted messages using the E-mail Object Protocol, as specified in [MS-OXOMSG].	[MS-OXOSMIME]
Rich Text Format (RTF) Compression	Specifies how to encode and decode a compressed stream in RTF message bodies.	[MS-OXRTFCP]
Rich Text Format (RTF) Extensions	Specifies how to encapsulate additional content formats (such as HTML) within the RTF body property of messages and attachments.	[MS-OXRTFEX]
Transport Neutral Encapsulation Format (TNEF)	Specifies how to encode and decode Message and Attachment objects to an efficient stream representation.	[MS-OXTNEF]
vCard to Contact Object Conversion	Specifies how the vCard format can be used by a Contact object to communicate with other contact systems.	[MS-OXVCARD]

2.2.3.1 Content Conversion File Formats

Protocol Name	Description	Document Short Name
.MSG File Format	Describes the file format used to serialize core Message objects and derived object types.	[MS-OXMSG]
Offline Address Book (OAB) Format and Schema	Specifies the offline address book (OAB) file formats. OABs are local client files that store address list information so that the client can access this information when it does not have a connection to the server or is working offline.	[MS-OXOAB]
Cabinet File Format	Specifies the Microsoft cabinet file format. Cabinet files are compressed packages containing a number of related files.	[MS-CAB]
MCI Compression and Decompression	Specifies the format of MSZIP compressed data as used in the MSZIP compression mode of cabinet files.	[MS-MCI]
LZX DELTA Compression and Decompression	Specifies the format of DELTA compression, a technique in which one set of data can be compressed within the context of a reference set of data that is supplied both to the compressor and decompressor.	[MS-PATCH]

2.2.4 Exchange ActiveSync Protocols

The Exchange ActiveSync protocols define how to share and synchronize data 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.

Protocol Name	Description	Document Short Name
ActiveSync AirSyncBase	Specifies the elements and complex types in the AirSyncBase namespace, which are used by the AirSync commands to identify	[MS-ASAIRS]

Protocol Name	Description	Document Short Name
Namespace	the size, type and content of the data returned to the client in response messages. The AirSyncBase namespace contains elements and complex types used in both request and response command messages.	
ActiveSync Calendar Class	Specifies the protocol format for the interchange of calendar data between a server and a client device.	[MS-ASCAL]
ActiveSync Command Reference	Specifies how to synchronize e-mail, attachments, contact information, calendar data, and document objects.	[MS-ASCMD]
ActiveSync Contact Class	Specifies the protocol format for the interchange of contact data between a server and a client device.	[MS-ASCNTC]
ActiveSync Conversations	Specifies the XML-based protocol used to improve the ways in which e-mails are displayed in the conversation view.	[MS-ASCON]
ActiveSync Document Class	Specifies how documents stored in Windows SharePoint Services and on file shares using Universal Naming Convention (UNC) paths are communicated from the server to the client.	[MS-ASDOC]
Exchange ActiveSync Data Types	Specifies the format of each data type used by the Exchange ActiveSync XML schema definitions (XSDs), which is transmitted in Wide Area Protocol (WAP) Binary XML (WBXML) format.	[MS-ASDTYPE]
ActiveSync E-Mail Class	Specifies the XML representation of e-mail data sent or received on mobile devices that communicate by using the Exchange ActiveSync protocols.	[MS-ASEMAIL]
ActiveSync HTTP	Specifies the format of communication between a client and a server via an HTTP POST using UTF-8 encoding.	[MS-ASHTTP]
ActiveSync Short Message Service (SMS)	Specifies an XML-based format to send and receive SMS messages.	[MS-ASMS]
ActiveSync Notes Class	Specifies the format and mechanisms that allow a mobile client to synchronize user notes.	[MS-ASNOTE]
ActiveSync Provisioning	Specifies an XML-based format to communicate security policy settings to client devices.	[MS-ASPROV]
ActiveSync Tasks Class	Specifies the format for the interchange of task data between a client and a server.	[MS-ASTASK]
ActiveSync WAP Binary XML (WBXML)	Specifies how WBXML functionality is utilized and specifies the tokens and code pages used to perform WBXML encoding.	[MS-ASWBXML]

2.2.5 Directory/Profile Services Protocols

The Directory/Profile Services protocols allow clients to find and use mail server configuration information.

Protocol Name	Description	Document Short Name
AutoDiscover HTTP Service	Specifies a request and response schema that allows clients to obtain configuration information about RPC- and Web service-based endpoints on a server. Server deployment and client discovery of the AutoDiscover HTTP service is specified in [MS-OXDSCLI].	[MS-OXDISCO]
Autodiscover Publishing and Lookup	Specifies how Autodiscover servers publish their locations. Autodiscover enables clients to retrieve URLs that are needed to gain access to Web services offered by Autodiscover servers.	[MS-OXDSCLI]

2.2.6 Name Service Provider Interface (NSPI) Protocols

The Name Service Provider Interface (NSPI) protocols specify a request and response format and related operations that allow access to address book, user, group, and resource information via a directory service.

Protocol Name	Description	Document Short Name
Name Service Provider Interface (NSPI)	Specifies how a client requests access to manipulate remote **Address Book objects** in a directory store.	[MS-NSPI]
Address Book Name Service Provider Interface (NSPI) Referral	Specifies how a client initially connects to a server and finds a new NSPI server instance.	[MS-OXABREF]
Address Book Object	Specifies the properties and operations permissible for lists of users, contacts, groups, and resources.	[MS-OXOABK]
Address Book User Interface Templates	Specifies the properties and operations permissible for address book templates.	[MS-OXOABKT]

2.2.7 Standards-Based Protocols

Exchange servers support a number of different standard protocols for e-mail (POP3, SMTP, IMAP4, and WebDAV) and directory information (LDAP). The Exchange Server Protocols documentation set specifies a number of extensions to these standards primarily for authentication and authorization.

Protocol Name	Description	Document Short Name
Internet Message Access Protocol Version 4 (IMAP4) Extensions	Includes an extension to the IMAP4rev1 Protocol, as specified in [RFC3501], to support NT LAN Manager (NTLM) authentication and access to mailboxes other than the one associated with the authenticated user.	[MS-OXIMAP4]
Lightweight Directory Access Protocol (LDAP) Profile	Specifies the Outlook client implementation of LDAPv3. The Lightweight Directory Access Protocol (LDAP) Profile Protocol is a binary TCP protocol that enables directory access.	[MS-OXLDAP]
POP3 extensions for NTLM authentication	Specifies an extension to the POP3 Mailbox Protocol, as specified in [RFC1939], to support NTLM authentication.	[MS-OXPOP3]

Protocol Name	Description	Document Short Name
SMTP Message Submission	Specifies an extension to the SMTP Message Submission Protocol, as specified in [RFC4409], to support NTLM authentication.	[MS-OXSMTP]
SMTP Protocol: AUTH LOGIN Extension	Specifies the extension to the Simple Mail Transfer Protocol (SMTP) to support a simple **base64-encoded** authentication mechanism.	[MS-XLOGIN]

2.2.7.1 Compliance-Related Standards-based Protocols

Protocol Name	Description	Document Short Name
Outlook iCalendar Standards Compliance	Specifies the level of support provided by the Outlook iCalendar component 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 Compliance	Specifies the level of support provided by the Internet Message Access Protocol (IMAP).	[MS-STANOIMAP]
Outlook Post Office Protocol Version 3 (POP3) Standards Compliance	Specifies the level of support provided by the Post Office Protocol version 3 (POP3) service.	[MS-STANOPOP3]
Exchange iCalendar Standards Compliance	Specifies the level of support provided by the Exchange iCalendar component 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 Compliance	Specifies the level of support provided by the Internet Message Access Protocol (IMAP).	[MS-STANXIMAP]
Exchange Post Office Protocol Version 3 (POP3) Standards Compliance	Specifies the level of support provided by the Post Office Protocol version 3 (POP3) service.	[MS-STANXPOP3]

2.2.8 Message Processing Protocols

Message processing protocols specify methods and extensions to existing protocols that allow clients and servers to interpret metadata for journaling and spam, phishing, and postmark validation and filtering.

2.2.8.1 Journal Message Processing

Protocol Name	Description	Document Short Name
Journal Record Message Format	Specifies a set of extensions based on [RFC2822], [RFC2045], and [RFC2046] to represent metadata required to journal or archive e-mail messages.	[MS-XJRNL]

2.2.8.2 Filter Message Processing

Protocol Name	Description	Document Short Name
Spam Confidence Level, Allow and Block Lists	Enables sharing of preferences for handling of unsolicited e-mail message filtering between the client and the server.	[MS-OXCSPAM]
Phishing Warning	Identifies and tags e-mail messages that are designed to trick recipients into divulging sensitive information to a non-trustworthy source.	[MS-OXPHISH]
E-Mail Postmark Validation	Specifies how to implement and validate a postmark property to help determine whether a message is spam.	[MS-OXPSVAL]

2.2.8.3 Sharing Message Processing

Protocol Name	Description	Document Short Name
Sharing Message Attachment Schema	Specifies the Sharing Message Attachment schema, which defines the schema for an XML document used to establish a sharing relationship between two servers on behalf of client applications.	[MS-OXSHRMSG]

2.2.9 Web Distributed Authoring and Versioning (WebDAV)

Web Distributed Authoring and Versioning (WebDAV) is a set of methods, headers, and content types that extends the Hypertext Transport Protocol – HTTP/1.1, as specified in [RFC2616]. WebDAV allows for the reading and writing of data to servers, as specified in [RFC4918].

The following protocols extend [RFC4918] to provide additional functionality.

Protocol Name	Description	Document Short Name
WebDAV Extensions for Calendar Support	Specifies property extensions to [RFC4918], [MS-WDVME], and [MS-WDVSE] to allow creation and manipulation of Calendar objects. These extensions specify properties that allow clients to locate a user's default calendar, get and set events on a calendar, and locate and gain access to a user's default free/busy information.	[MS-XWDCAL]
WebDAV Extensions for Contacts Support	Specifies property extensions to the WebDAV protocol that allow creation and manipulation of Contact objects.	[MS-XWDCNTC]

Protocol Name	Description	Document Short Name
WebDAV Extensions for Documents Support	Specifies property extensions to [RFC4918], [MS-WDVME], and [MS-WDVSE] that allow the creation and manipulation of Document objects.	[MS-XWDDOC]
WebDAV Core Extensions	Specifies extensions to WebDAV [RFC2518] that provide additional functionality necessary for implementing WebDAV-based protocols.	[MS-XWDEXT]
WebDAV Extensions for Folder Support	Specifies a set of methods, headers, and properties that extends the HTTP and WebDAV protocols to support folders.	[MS-XWDFOLD]
WebDAV Extensions for E-Mail Support	Used to send, receive, and manipulate e-mail through HTTP.	[MS-XWDMAIL]
WebDAV Extensions for Notifications	Extends the WebDAV protocol, as specified in [RFC4918], to provide event notifications.	[MS-XWDNOTIF]
WebDAV Extensions for Replication	Specifies the client-server replication of Web resources.	[MS-XWDREPL]
WebDAV Extensions for Search	Specifies extensions to the WebDAV protocol [RFC4918] to allow clients to request content searches. This protocol also allows clients to create persistent search folders on the server	[MS-XWDSEARCH]
WebDAV Extensions for Structured Documents	Specifies extensions to the WebDAV protocol [RFC4918] to allow creation and manipulation of structured documents. This allows clients to retrieve, insert, change, and remove individual pieces of structured documents on the server.	[MS-XWDSTRUCTDOC]
WebDAV Extensions for Security	This extension specifies how to request and set the Exchange security descriptor by using the WebDAV methods PROPFIND and PROPPATCH.	[MS-XWDVSEC]

2.2.10 Web Services and HTTP-based Protocols

Web Services and HTTP/1.1 provide a standards-based layer upon which to build specific client-server protocols. The protocols specified below 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.

Protocol Name	Description	Document Short Name
Availability Web Service	Specifies methods used to retrieve calendar-specific data from a server, such as whether a user is free or busy.	[MS-OXWAVLS]
Service Configuration	Specifies the request-response messages for retrieving organization policy configuration from a server.	[MS-OXWCONFIG]
Mailtips Extensions to	Specifies the format for requests and responses that	[MS-OXWMT]

Protocol Name	Description	Document Short Name
Web Service	provide information to the author composing a message to show what the sent text will look like.	
Offline Address Book (OAB) Retrieval	Specifies how clients download full and incremental versions of the Offline Address Book (OAB). When this protocol is not available, the OAB Public Folder Retrieval Protocol [MS-OXPFOAB] is used.	[MS-OXWOAB]
Voice Mail Settings Web Service	Specifies the schema and methods used to configure server-based voice mail, including requesting that the server initiate playback of voicemail messages on a telephone.	[MS-OXWUMS]
Out of Office (OOF) Web Service	Specifies how to configure automated response behavior when a user is unavailable.	[MS-OXWOOF]
AutoDiscover Publishing and Lookup SOAP-based Web Service	Allows clients to learn e-mail configuration settings for specific e-mail addresses. Web Services protocols implement a single method named **GetUserSettings** to request configuration settings by passing in the e-mail address.	[MS-OXWSADISC]
Attachment Handling Web Service	Provides clients with the basis for attaching objects to messages.	[MS-OXWSATT]
Authentication Identification Extension	Specifies an extension which is an extended HTTP header that defines the authentication schemes that are supported by a client.	[MS-OXWSAUTID]
Common Exchange Server Web Service Data Types	Specifies common global elements and types used by other Web Services protocols.	[MS-OXWSCDATA]
Contacts Web Service	Provides the messages needed to create, get, update, delete, move, and copy contact items on the server.	[MS-OXWSCONT]
Core Items Web Service	Specifies the Core Items Web Service Protocol, which is responsible for creating, updating, and deleting items on the server.	[MS-OXWSCORE]
Convert Item Identifier Web Service	Enables a client 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	Provides the capability for delegate access management to mailbox information stored on a server.	[MS-OXWSDLGM]
Distribution List Creation and Usage Web Service	Provides clients with the ability to query the server for distribution lists and to expand a distribution list into the constituent e-mail addresses. In addition, it provides the capability to create, delete, get, move, update, and copy distribution lists.	[MS-OXWSDLIST]
Folders and Folder Permissions Web Service	Provides clients with the folder operations for retrieving folder permission lists that are stored on the server. The protocol specifies the elements and operations for creating	[MS-OXWSFOLD]

Protocol Name	Description	Document Short Name
	and locating the special folders in a mailbox.	
Get Rooms List Web Service	Provides a client with a list of locations of meeting rooms within the server organization. This protocol also provides the client with the list of meeting rooms within a selected location room list.	[MS-OXWSGTRM]
Get Server Time Zone Web Service	Provides the capability for returning time zone information that is used by the server.	[MS-OXWSGTZ]
Federated Internet Authentication Web Service	Specifies the Federated Internet Authentication Web Service Protocol, which defines the interaction between the server and standard Internet authentication protocols.	[MS-OXWSLVID]
E-mail Message Types Web Service	Provides clients with the ability to create, update, and delete e-mail items on the server	[MS-OXWSMSG]
Folder Sharing Web Service	Specifies the Folder Sharing Web Service Protocol, which is responsible for managing Calendar folders that are shared between users in separate organizations.	[MS-OXWSMSHR]
Calendaring Web Service	Specifies the messages for creating, getting, deleting, updating, moving, and copying meeting items on the server.	[MS-OXWSMTGS]
Message Tracking Web Service	Provides the capability for finding and returning information about message delivery by the server.	[MS-OXWSMTRK]
Notifications Web Service	Specifies how clients receive pull notifications from the server.	[MS-OXWSNTIF]
Post Items Web Service	Provides clients with the ability to send items from the server.	[MS-OXWSPOST]
Push Notifications Web Service	Allows a client to receive subscribed events sent by the server.	[MS-OXWSPSNTIF]
Resolve Recipient Names Web Service	Enables a client with incomplete recipient identifying information to retrieve a list of matching and similar recipients that are known to the server.	[MS-OXWSRSLNM]
Mailbox Search Web Service	Specifies a protocol to search the contents of a mailbox and return the specified folders or items.	[MS-OXWSSRCH]
Mailbox Contents Synchronization Web Service	Provides the capability to keep a local mailbox store synchronized with the server mailbox store.	[MS-OXWSSYNC]
Tasks Web Service	Provides the capability to create, update, and delete task items on the server.	[MS-OXWSTASK]
User Configuration Web Service	Specifies the SOAP request-response messages used to create, get, update, and delete user configuration objects.	[MS-OXWSUSRCFG]
Extended Exchange Server Web Service	Specifies the properties of an extended property object attached to an exchange item.	[MS-OXWSXPROP]

2.3 Environment

The following sections identify the context in which the Exchange Server 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 components of the system communicate.

2.3.1 Dependencies on this System

The Exchange Server Protocols system supports a diverse set of protocols, ranging from standards-compliant protocols such as SMTP, to rich, proprietary protocols such as the Exchange RPC APIs. 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 Exchange Server Protocols.

2.3.1.1 RPC-enabled Clients

These messaging clients 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, appointments, and shared storage.

2.3.1.2 Non-RPC-enabled Clients

These messaging clients communicate with Exchange servers through standards-based protocols such as SMTP, POP3, and IMAP4. Most of these clients only offer standard e-mail services such as sending and retrieving e-mail messages. More advanced features such as contacts and appointments are usually implemented on the client rather than on the server.

2.3.1.3 Mobile Device Clients

Exchange ActiveSync provides 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 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 includes the capability to integrate e-mail and related messaging systems, and 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 Exchange Server 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 are dependent 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

Domain Name Service (DNS) 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.

2.3.3 Communications within the System

This section describes the internal communications within the Exchange Server system. For a helpful reference to this section, see Figure 1: System Overview in section 2.1.

2.3.3.1 Between an E-mail Client and Exchange Servers

This represents the inbound e-mail transmission between an e-mail client and Exchange servers. SMTP or SMTP plus Exchange-specific extensions to SMTP [MS-OXSMTP] are implemented for this communication.

2.3.3.2 Between a Messaging Client and Exchange Servers

In this context, a messaging client refers to any generic client that makes use of the Exchange messaging system. A messaging client does not necessarily have to be an e-mail client. As illustrated in Figure 1: System Overview (see section 2.1), messaging clients have a variety of protocols options to communicate with Exchange servers, including RPC, POP3, IMAP4, WebDAV, Web Services, NSPI and Unified Messaging.

2.3.3.3 Between a Mobile Client Device and Exchange Servers

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

2.3.3.4 Between a Messaging Client and Directory Service

Exchange Server Autodiscover requires messaging clients to communicate with the directory service to locate Autodiscover sites. The Lightweight Directory Access Protocol (LDAP) is used for this communication.

2.3.3.5 Between a Messaging Client and Domain Name Service (DNS)

Exchange Server Autodiscover requires messaging clients to communicate with DNS servers to locate alternate Autodiscover sites.

2.3.4 Assumptions and Preconditions

This section briefly documents the assumptions and preconditions required by the system. The scope of this discussion is intended to be implementation-independent and is limited to the system-level.

- A domain controller is required to service the server domain, authentication requests, and 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 all intended clients can resolve the domain name.
- The Exchange server functional components are started collectively and the Exchange server accepts client requests.
- The Directory Service is accessible to the Exchange server. Any intermediate firewalls, routers, or connection points between components of the system have all required ports and gateways open for communication between them.

2.4 Use Cases

This section presents a number of use cases that illustrate the key functionality of Exchange servers. Due to the complexity of the Exchange Server 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 Exchange Server 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. 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.

2.4.1 Server Information Discovery

2.4.1.1 Synopsis

This 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.

2.4.1.2 Builds on Use Case(s)

None.

2.4.1.3 References

- LDAP and LDAP Directories [RFC1823]
- LDAP Data Interchange Format LDIF [RFC2849]
- DNS and DNS SRV Records [RFC1034]
- HTTP [RFC2616]

- [MS-OXDISCO]

2.4.1.4 Requirements

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

2.4.1.5 Protocol-specific Details

- Using RPC:

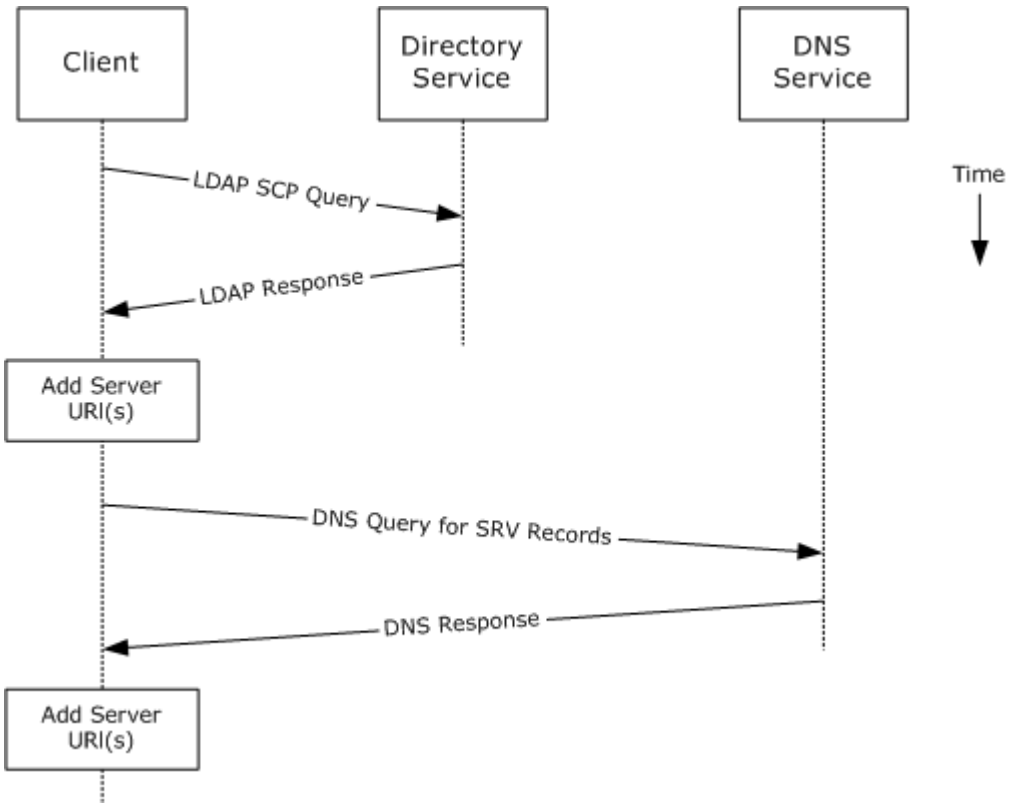

Figure 4:

1. Client contacts LDAP server (Directory Service) for Service Connection Point (SCP) objects via LDAP.
2. LDAP server returns SCP object(s), which reference an Autodiscover server URI(s) or another LDAP server. If the SCP returns another LDAP server, then repeat step i) with the new LDAP server until URI(s) are returned for Autodiscover server(s).
3. Client parses the URI and adds the appropriate Autodiscover server URIs to the list of possible Autodiscover server URIs.
4. 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, then the corresponding Autodiscover server URI records are added to the list of possible Autodiscover server URIs on the client.

   Note It is not an error if the DNS server does not return any DNS SRV records in response to the DNS search.

6. Client uses the Autodiscover server URI to contact the Autodiscover server via HTTP to query server information.

2.4.2 Log on to a Mailbox

2.4.2.1 Synopsis

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

Note This use case only applies to RPC clients.

2.4.2.2 Builds on Use Case(s)

2.4.1 Server Information Discovery

2.4.2.3 References

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

2.4.2.4 Requirements

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

2.4.2.5 Protocol-specific Details

- Using RPC

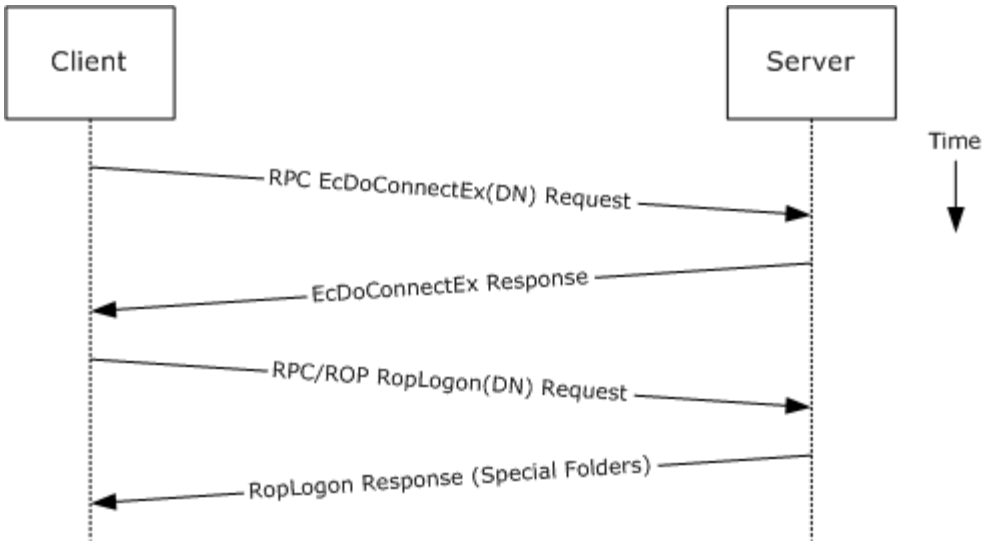

Figure 5:

1. Client uses the discovery process from the previous use case to identify the appropriate server.
2. The client connects to the Exchange server via RPC and issues an EcDoConnectEx (or EcDoConnect) request, 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 RopLogon Remote Operation (ROP) request to attempt to log on to the mailbox DN.
5. Upon successful logon, the Exchange server returns a list of special Folder IDs [MS-OXOSFLD] depending on the logon action requested by the client.

2.4.3 Create a Message

2.4.3.1 Synopsis

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

2.4.3.2 Builds on Use Case(s)

2.4.2 Log on to a Mailbox

2.4.3.3 References

- [MS-OXCROPS]
- [MS-OXCMSG]
- [MS-OXCFOLD]
- [MS-OXWSMSG]

- [MS-XWDEXT]
- [MS-XWDMAIL]
- [MS-ASDTYPE]
- [MS-ASHTTP]
- [MS-ASCMD]
- [MS-ASEMAIL]
- [RFC2616]

2.4.3.4 Requirements

The client knows the folder ID in which to create the message. The folder ID can be one of the special folders returned from the logon operation.

2.4.3.5 Protocol-specific Details

Note Several Remote Operations (ROPs) can be batched into a single request [MS-OXCROPS].

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.

1. Using RPC:

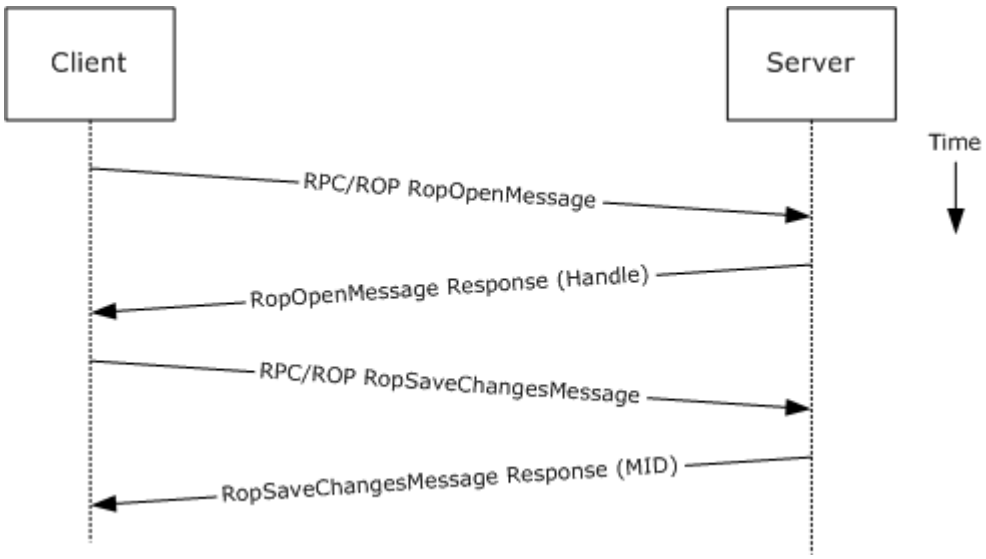

Figure 6:

1. The client logs on to a mailbox as per the previous use case.
2. The client issues a RopOpenMessage request to the Exchange server referencing the folder.
3. The Exchange server responds with a handle to the message.

4. Using the Message ID, the client issues a RopSaveChangesMessage request to the server to persist the message to storage.

5. The server returns the Message ID (MID) for the new message.

2. Using WebDAV:

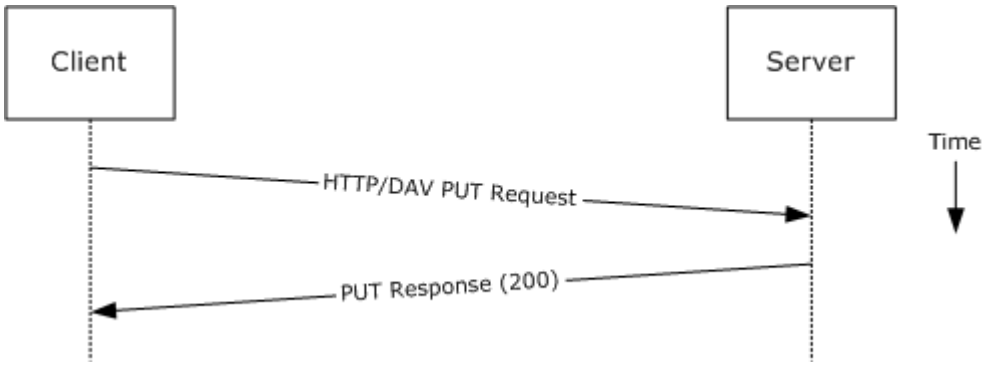

Figure 7:

1. Client uses the HTTP PUT request [RFC2616] to create a new message document resource, with Exchange-specific extensions specified in [MS-XWDEXT]. The e-mail document contains the required properties, as specified in [MS-XWDMAIL].

2. The Exchange server responds as specified in [RFC2616] plus Exchange-specific extensions.

3. Using Exchange ActiveSync:

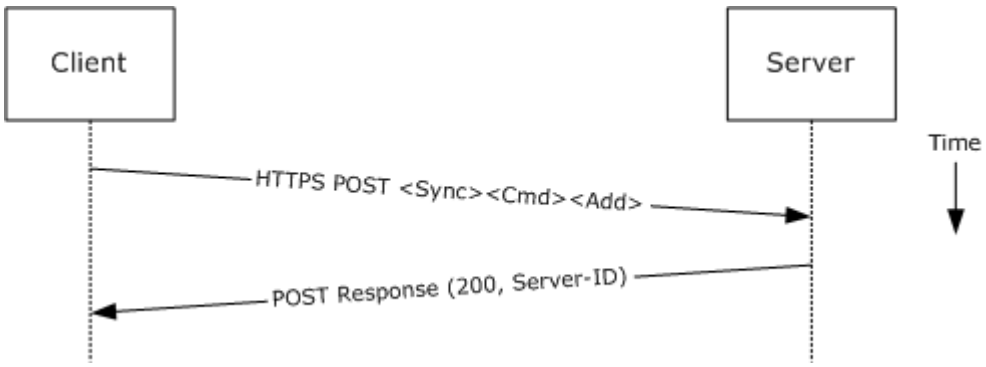

Figure 8:

1. Client uses the Sync Add Command request [MS-ASCMD] to upload/create new ApplicationData on the server. ApplicationData of class "E-mail" needs to contain the required XML schema elements, as specified in [MS-ASEMAIL].

2. Exchange server responds with a Server ID that identifies the uploaded ApplicationData.

4. Using Exchange Web Services:

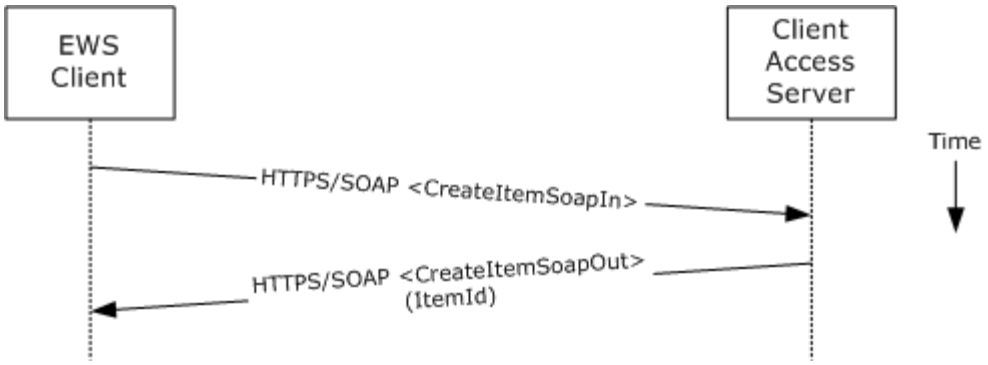

Figure 9:

1. The client uses the HTTPS/SOAP **CreateItemSoapIn** request ([MS-OXWSMSG] section 3.1.4.1) to create a new message item in the specified folder.

2. The Exchange Client Access server responds with a **CreateItemSoapOut** response ([MS-OXWSMSG] section 3.1.4.1), which includes the <ResponseCode> element indicating the status of the operation and the <ItemId> element, whose value uniquely identifies the new message.

2.4.4 Create a Strongly-Typed Message

2.4.4.1 Synopsis

This 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.

2.4.4.2 Builds on Use Case(s)

2.4.3 Create a Message

2.4.4.3 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]
- [MS-XWDMAIL]
- [MS-XWDCNTC]
- [MS-XWDDOC]
- [MS-XWDSTRUCTDOC]
- HTTP Extensions for WebDAV [RFC4918]

2.4.4.4 Requirements

The kind of strongly-typed message to be created needs to be pre-determined.

2.4.4.5 Protocol-specific Details

1. Using RPC:

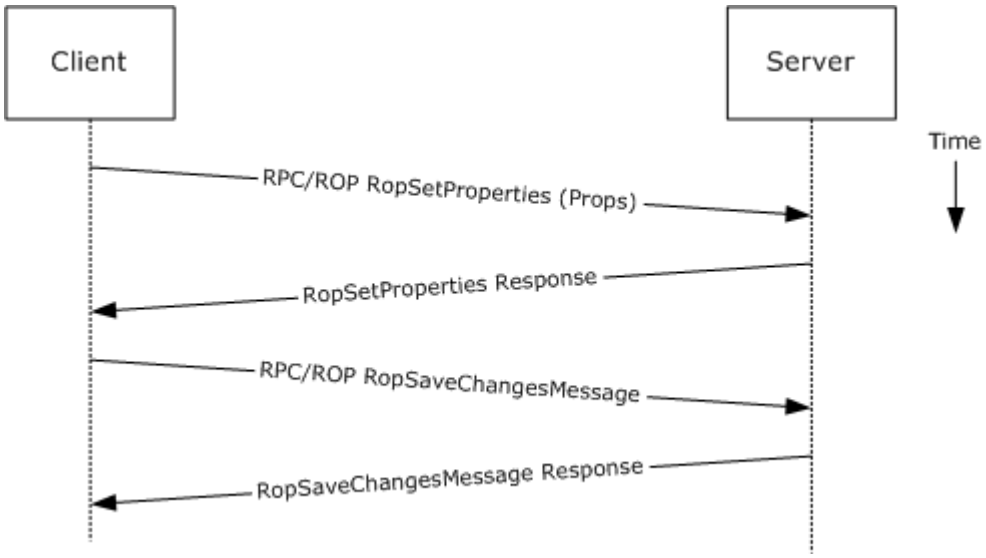

Figure 10:

1. The client creates a message as per the previous use case.
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 request 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 request 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 the any of the required properties associated with the selected message class are not present.

2. Using WebDAV:

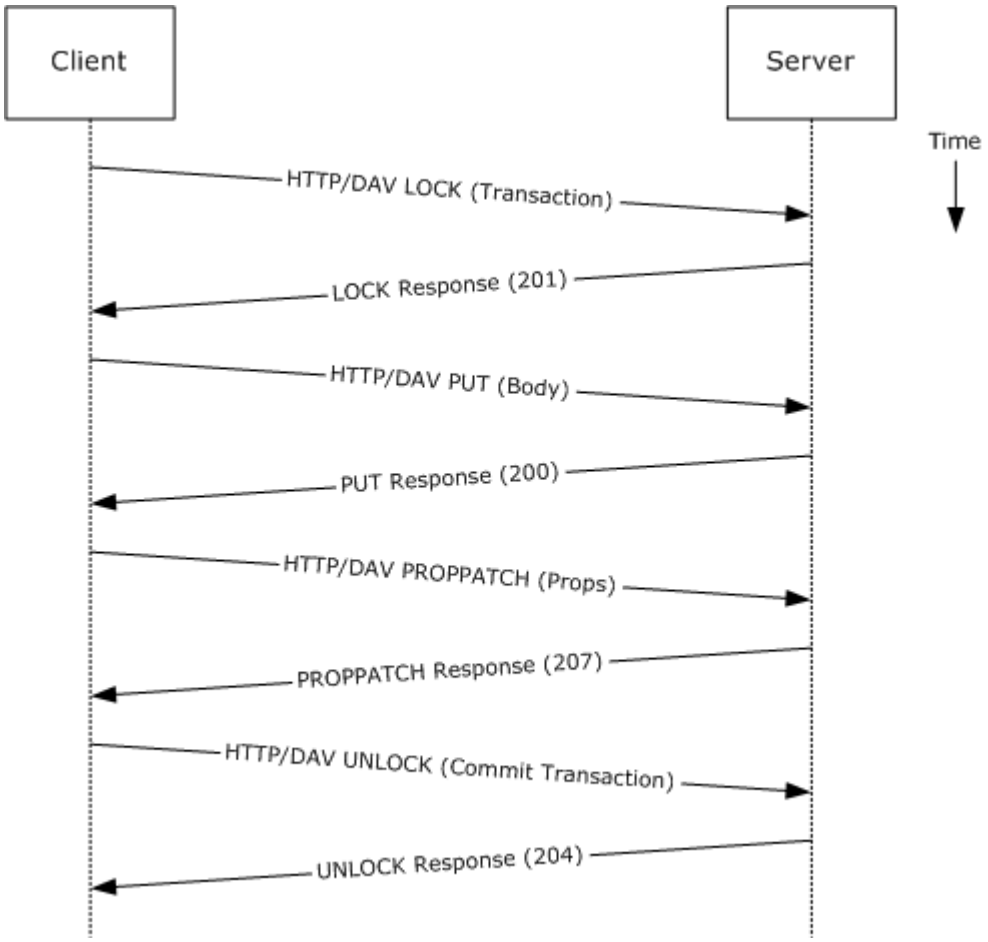

Figure 11:

1. The client creates a transaction [MS-XWDEXT].
2. The client prepares the body for the message [MS-XWDMAIL].
3. The client uses PUT to set the body of the message.
4. The client prepares a list of property-value pairs to be set on the message. The property-value pairs include the message class plus the required properties associated with the specific message class value.
5. The client uses the PROPPATCH request to add the required properties to the message.
6. The client uses UNLOCK to commit the transaction.
7. The server returns the success or failure of the operation.

Note Several documents can be created/modified/deleted within a transaction.

3. Using Exchange ActiveSync:

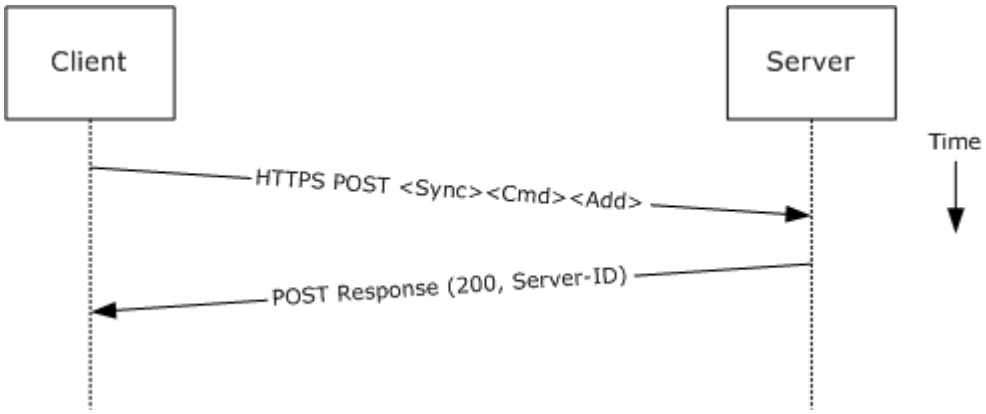

Figure 12:

1. The client creates a new message (ApplicationData) as per the previous use case.
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 to add the required schema items to the message (ApplicationData).
4. The server returns the success or failure of the operation.

4. Using Exchange Web Services:

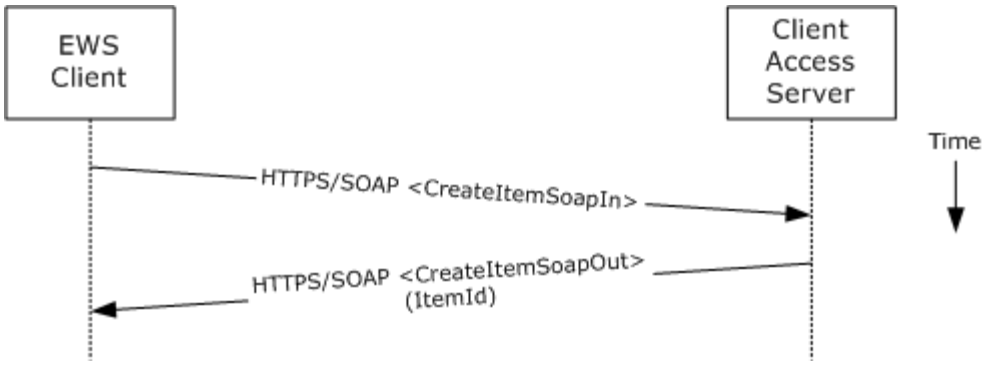

Figure 13:

1. The client uses the HTTPS/SOAP **CreateItemSoapIn** request ([MS-OXWSMSG] section 3.1.4.1) 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 Exchange Client Access Server responds with a **CreateItemSoapOut** response ([MS-OXWSMSG] section 3.1.4.1), which includes a <ResponseCode> element specifying the status of the operation and an <ItemId> element whose value uniquely identifies the new message.

2.4.5 Add an Attachment

2.4.5.1 Synopsis

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

2.4.5.2 Builds on Use Case(s)

2.4.3 Create a Message

2.4.5.3 References

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

2.4.5.4 Requirements

The attachment available to the client.

2.4.5.5 Protocol-specific Details

1. Using RPC:

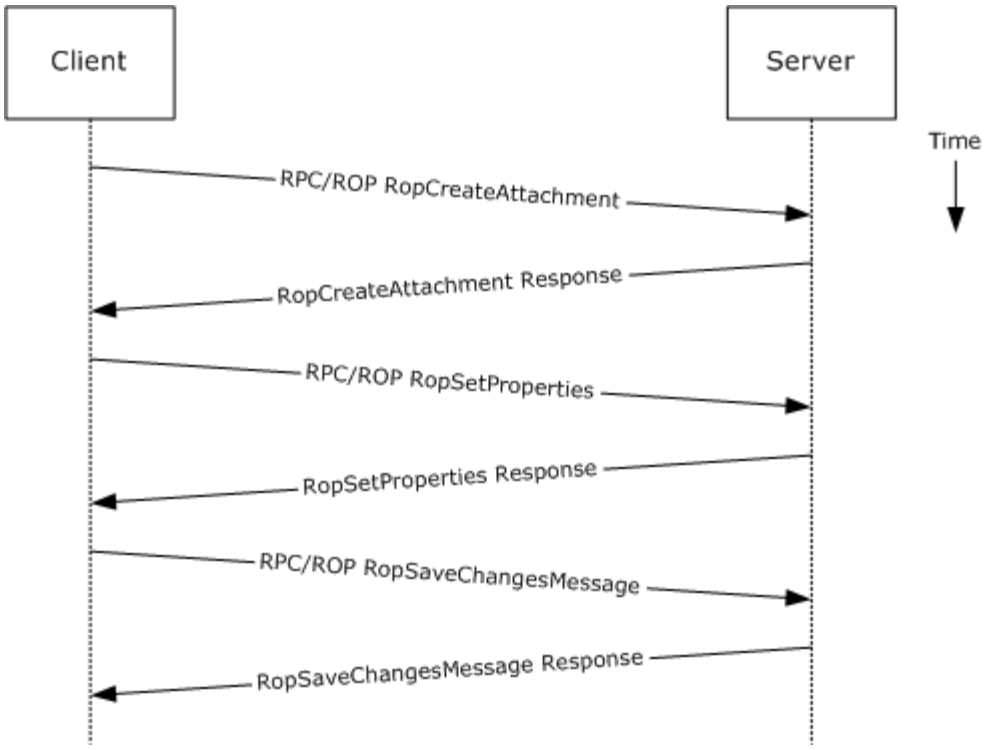

Figure 14:

1. The client creates a message as per the previous use case.
2. The client issues a RopCreateAttachment request 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 request 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 attachment request to the Exchange server to persist the new attachment.
8. The Exchange server returns the success or failure of the save operation. This operation can fail when the any of the required properties for an attachment are not present.

2. Using WebDAV:

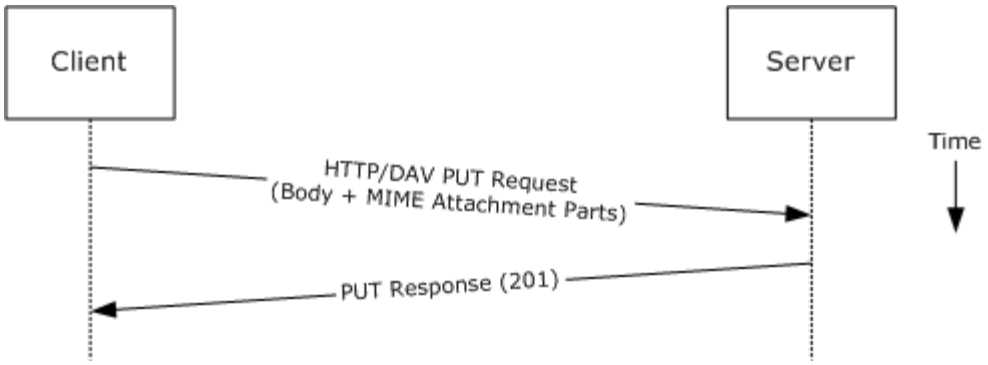

Figure 15:

1. The client uses the HTTP PUT request [RFC2616] to replace the existing message document resource, specifying a new message body that contains the attachment data as embedded MIME body parts.

2. The Exchange server responds per [RFC2616] and Exchange-specific extensions.

3. Using Exchange ActiveSync:

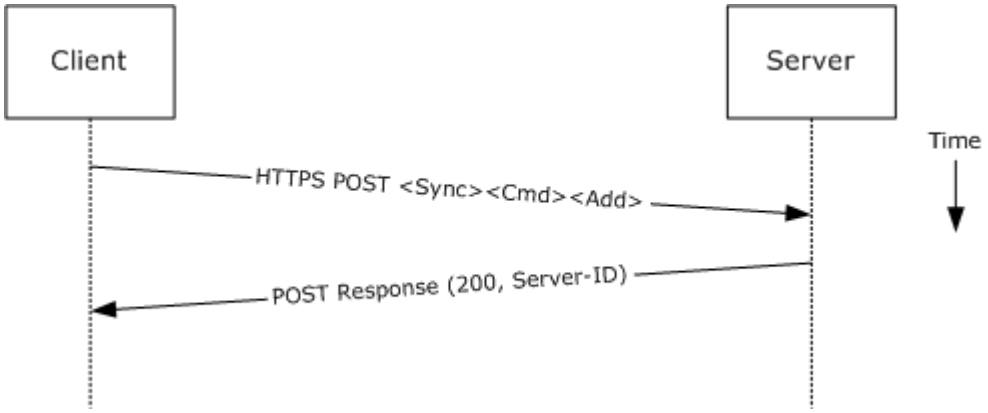

Figure 16:

1. The client creates a new message (ApplicationData) as per the previous use case.

2. The client uses the Sync Add Command request [MS-ASCMD] to upload/create new ApplicationData of class "Attachment" on the server. The attachment is added to a collection inside the message.

3. Exchange server responds with a server ID that identifies the uploaded ApplicationData (attachment).

4. Using Exchange Web Services:

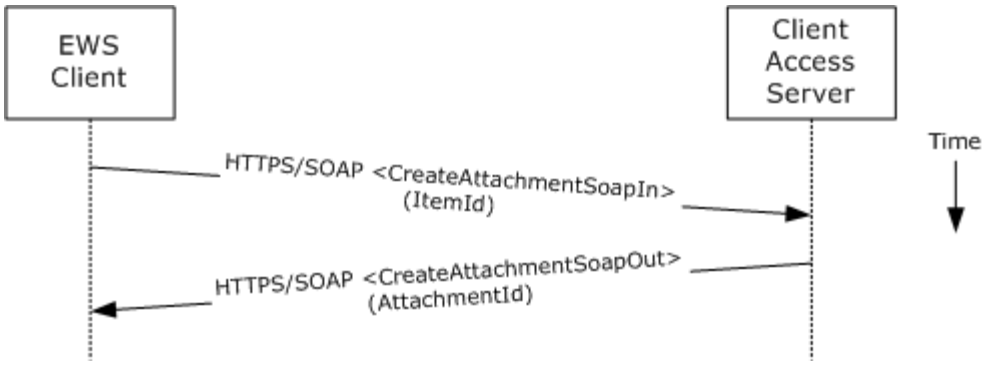

Figure 17:

1. The client creates a new message item as specified in section 2.4.3 and notes the value of the <ItemId> element for the new message.
2. The client uses the HTTPS/SOAP **CreateAttachmentSoapIn** request ([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 Exchange Client Access server responds with a **CreateAttachmentSoapOut** response ([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.

2.4.6 Resolve a Recipient from an Address Book

2.4.6.1 Synopsis

This 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.

2.4.6.2 Builds on Use Case(s)

None.

2.4.6.3 References

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

2.4.6.4 Requirements

The client has a specific recipient name to resolve.

2.4.6.5 Protocol-specific Details

1. Using RPC:

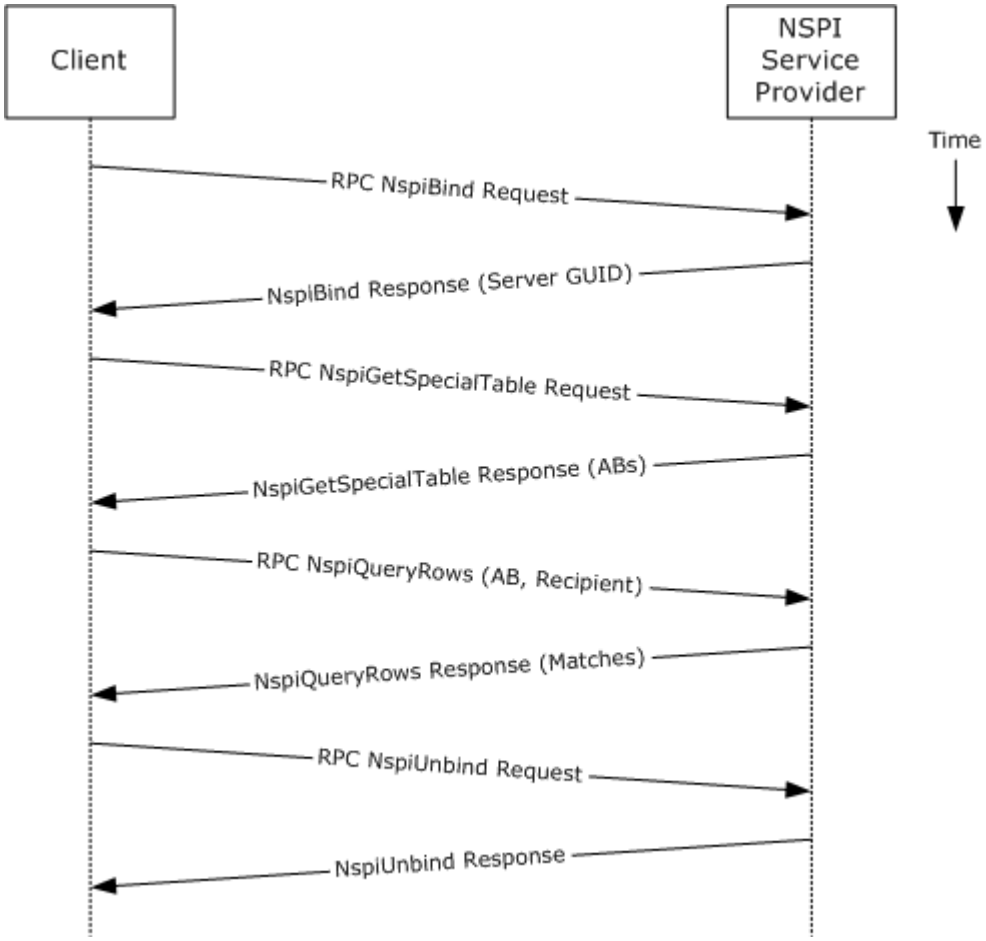

Figure 18:

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

Note If no rows are returned, then a match was not found for the recipient. If a single row is returned, then an exact match was found. If more than one row is returned, then 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.

2. Using Exchange ActiveSync:

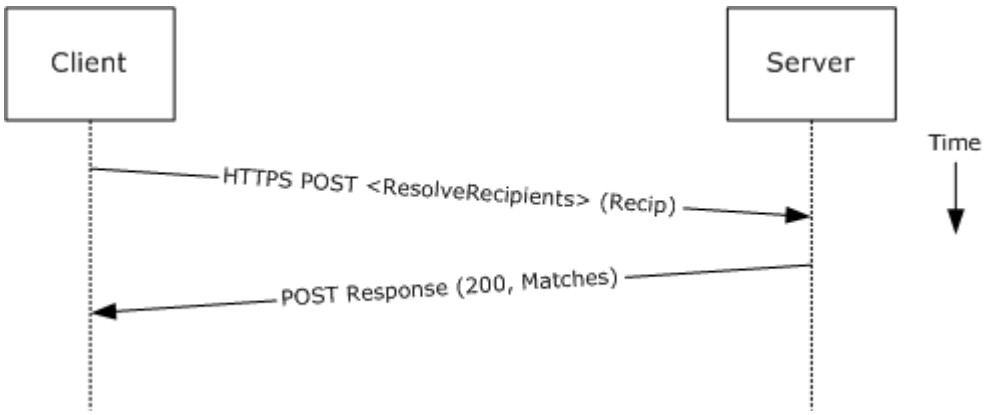

Figure 19:

1. The client issues a ResolveRecipients command request [MS-ASCMD] for the specific recipient to the server.

2. The Exchange server responds to the request by returning 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.

3. Using Exchange Web Services:

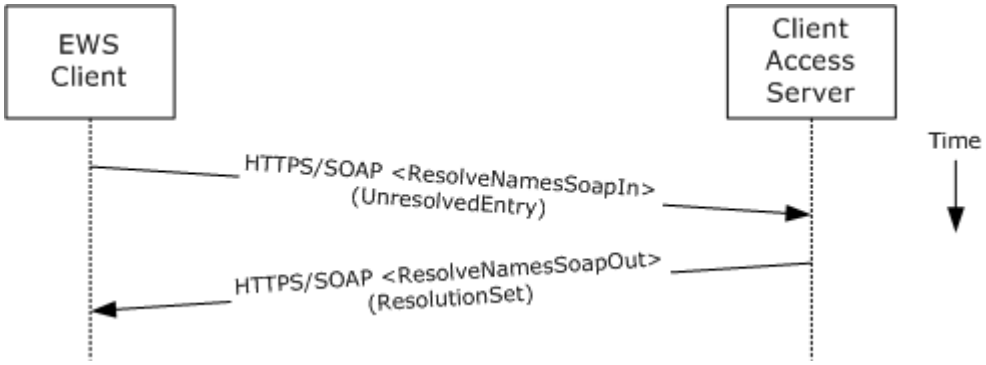

Figure 20:

1. The client uses the HTTPS/SOAP **ResolveNamesSoapIn** request ([MS-OXWSRSLNM] section 3.1.4.1.4.1) to resolve an unresolved entry.

2. The Exchange Client Access server responds with a **ResolveNamesSoapOut** response ([MS-OXWSRSLNM] section 3.1.4.1.4.2), which includes the <ResponseCode> element and the <ResolutionSet> element ([MS-OXWSRSLNM] section 3.1.4.1.2.1) containing the list of matching names found in the resolution set.

2.4.7 Send a Message

2.4.7.1 Synopsis

This use case describes how a messaging client can send a message (submit a message for delivery) using the RPC and Exchange Web Services protocols. SMTP-only clients cannot send messages using this mechanism. For more details about SMTP-based message submission, see [MS-OXSMTP] and [RFC2821].

2.4.7.2 Builds on Use Case(s)

- 2.4.3 Create a Message
- 2.4.6 Resolve a Recipient from an Address Book

2.4.7.3 References

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

2.4.7.4 Protocol-specific Details

1. Using RPC:

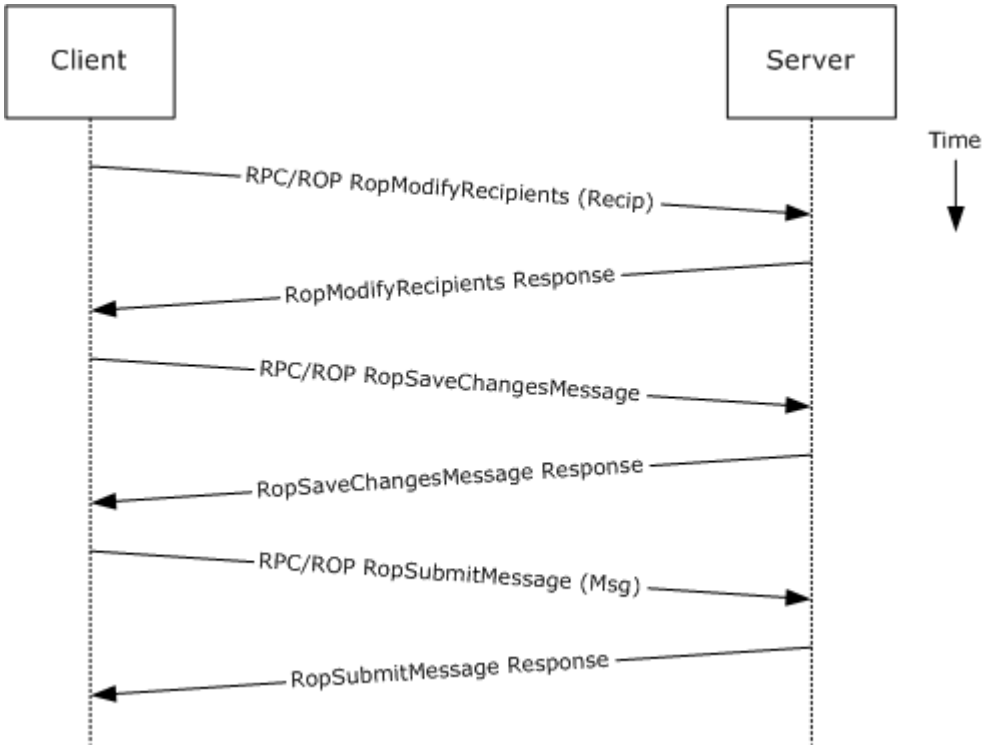

Figure 21:

1. The client creates a message as per the previous use case.
2. The client resolves a recipient as per the previous use case.
3. The client issues a RopModifyRecipients request 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 request to save the new recipient table.
6. The Exchange server returns the success or failure of the operation.
7. The client issues a RopSubmitMessage request to submit the message for delivery.
8. The Exchange server returns the success or failure of the operation.

2. Using Exchange Web Services:

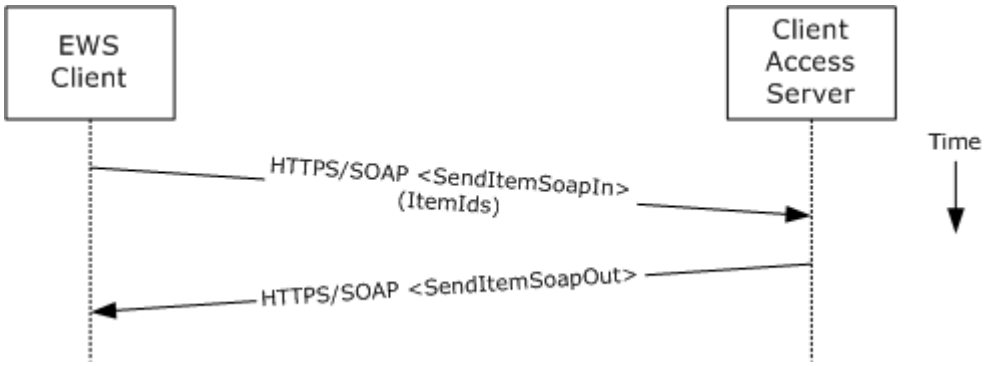

Figure 22:

1. The client uses the HTTPS/SOAP **SendItemSoapIn** request ([MS-OXWSMSG] section 3.1.4.7) 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 ([MS-OXWSMSG] section 3.1.4.7), which includes the <ResponseCode> element specifying the status of the operation.

2.4.8 Sending a Message to a Remote Recipient

2.4.8.1 Synopsis

This 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 utilized in message processing. The RPC protocol is used to illustrate this use case.

2.4.8.2 Builds on Use Case(s)

2.4.7 Send a Message

2.4.8.3 References

- [MS-SMTP]
- [MS-OXCETF]
- [MS-OXCICAL]
- [MS-OXCMAIL]
- [MS-OXOSMIME]
- [MS-OXRTFCP]
- [MS-OXRTFEX]
- [MS-OXTNEF]
- [MS-OXVCARD]

2.4.8.4 Requirements

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

2.4.8.5 Protocol-specific Details

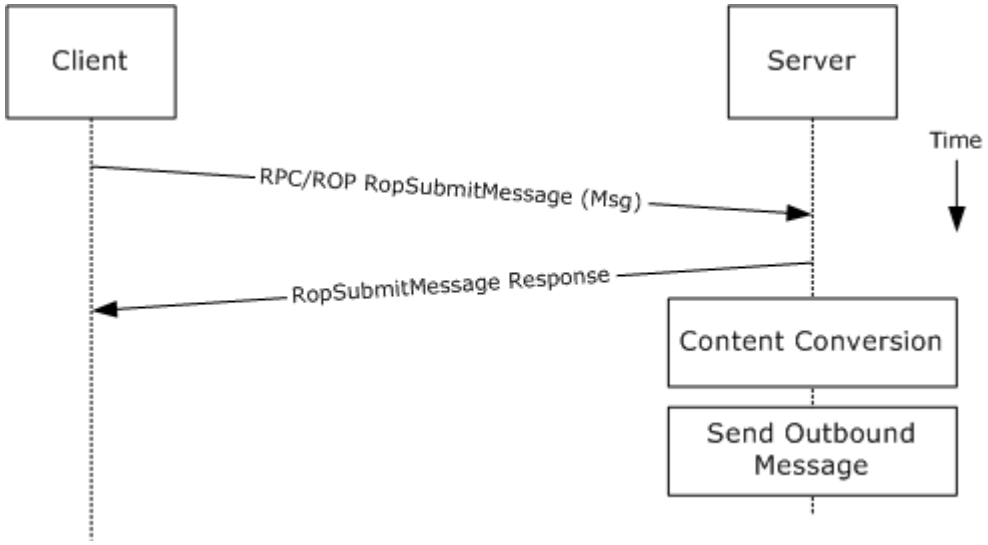

Figure 23:

1. The messaging client submits a message for delivery per use case 2.4.7 (Sending a Message), 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 details regarding SMTP mail submission and Exchange-specific SMTP extensions, see [RFC2821] and [MS-OXSMTP].

2.4.9 Open a Folder

2.4.9.1 Synopsis

This use case describes how a client opens a folder.

Note This use case only applies to clients using the RPC protocol.

2.4.9.2 Builds on Use Case(s)

2.4.2 Log on to a Mailbox

2.4.9.3 References

- [MS-OXCFOLD]
- [MS-OXPROPS]

2.4.9.4 Requirements

The client knows the folder ID (FID) or path for the folder that is to be opened.

2.4.9.5 Protocol-specific Details

- Using RPC:

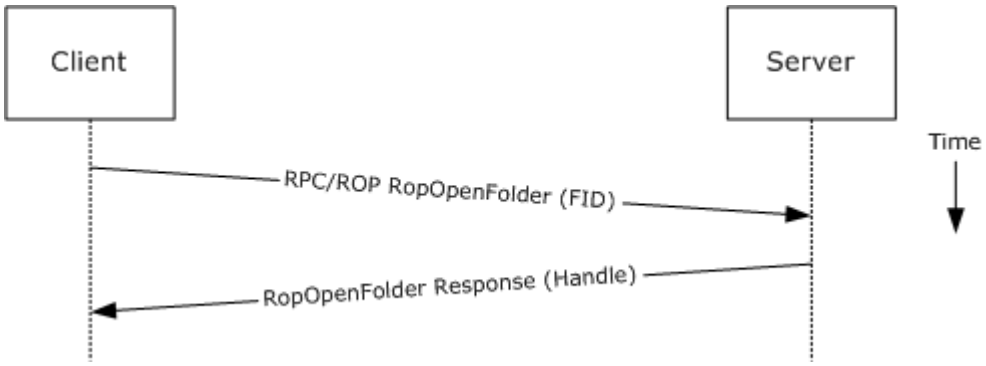

Figure 24:

1. The client logs on to a mailbox as per the previous use case.
2. The messaging client issues a RopOpenFolder request to Exchange server to open the specified folder.
3. The server responds with a handle to the folder.

2.4.10 Finding Items in a Folder that Match Search Criteria

2.4.10.1 Synopsis

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

2.4.10.2 Builds on Use Case(s)

2.4.9 Open a Folder

2.4.10.3 References

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

2.4.10.4 Requirements

The client knows the folder ID (FID) or path for the folder that is to be searched.

2.4.10.5 Protocol-specific Details

1. Using RPC:

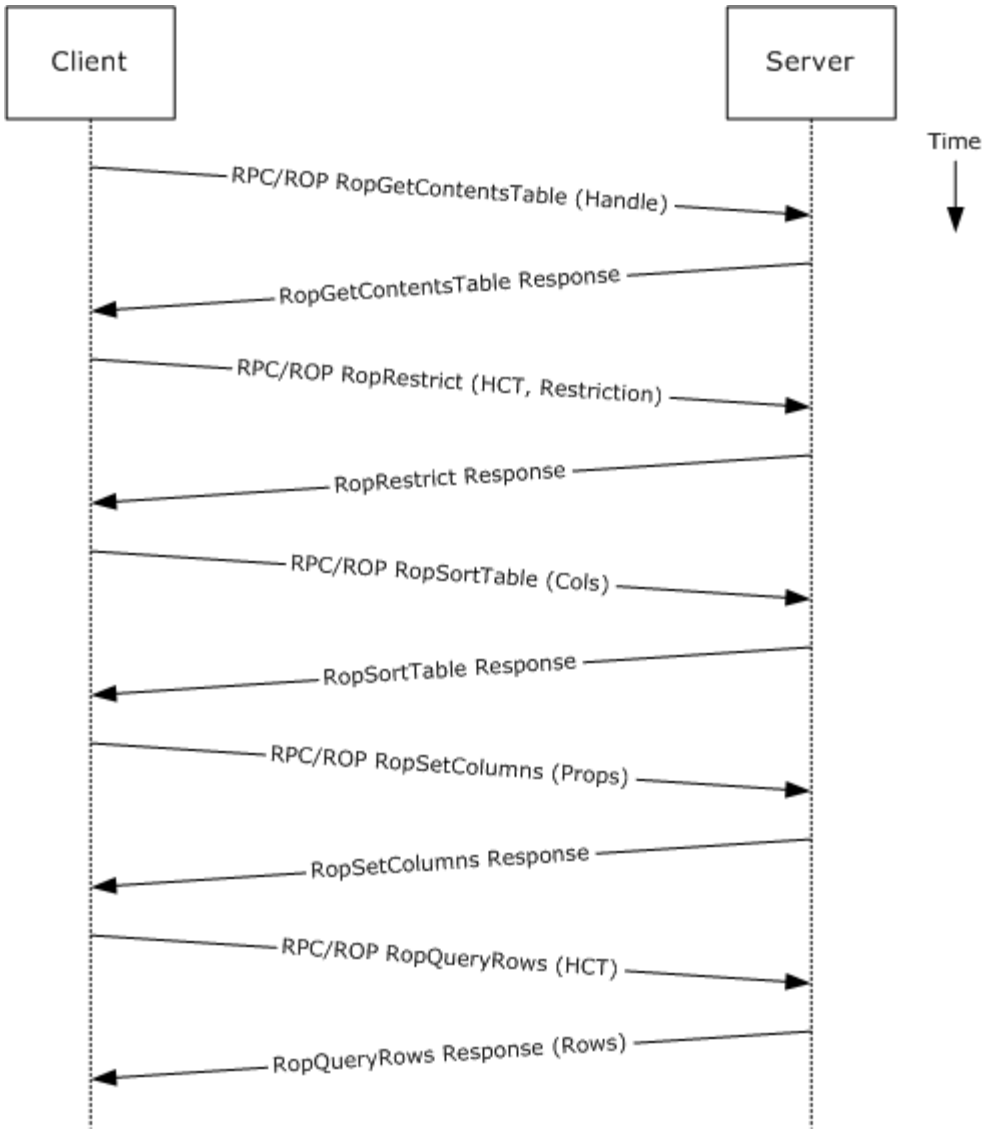

Figure 25:

1. The client opens the specified folder as per the previous use case.
2. The client issues a RopGetContentsTable request 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 [MS-OXCDATA].
5. The client issues a RopRestrict request with the constructed restriction to establish the search criteria.
6. The Exchange server responds to the RopRestrict request.

7. The client can optionally issue a RopSortTable request to specify a series of sort columns in the resulting table.

8. The client prepares a list of desired properties to retrieve, and issues a RopSetColumns request to indicate the desired property columns.

9. The Exchange server responds to the RopSetColumns request.

10. The client issues a RopQueryRows request to retrieve rows from the contents table.

11. 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 above, and each column correspond to the properties indicated in step 8 above.

2. Using WebDAV:

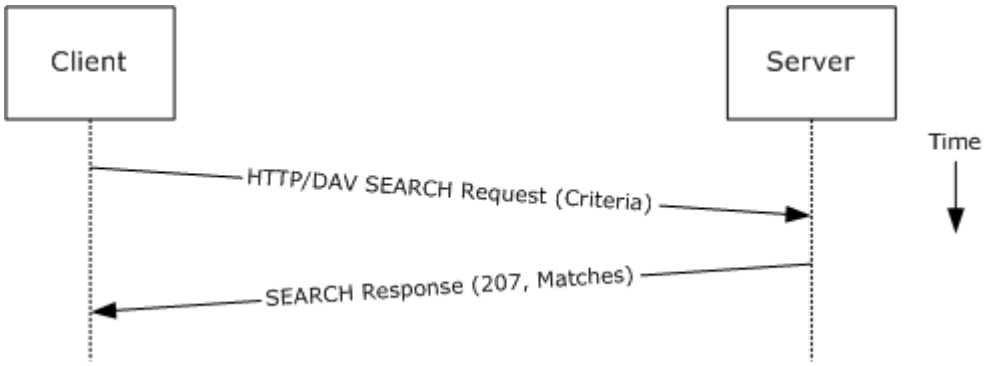

Figure 26:

1. The client issues a SEARCH request 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 a series of responses, where each corresponds to a matching entry in the folder.

3. Using Exchange ActiveSync:

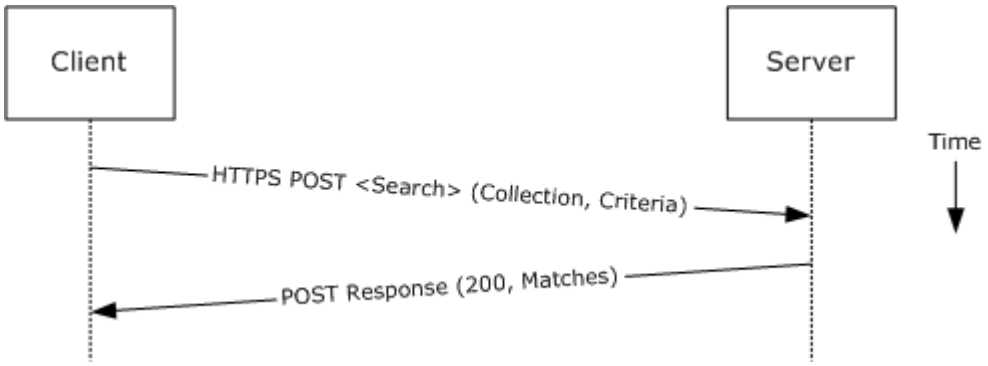

Figure 27:

1. The client issues a search command request 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 a collection of results, where each corresponds to an item in the folder that matches the search criteria.

4. Using Exchange Web Services:

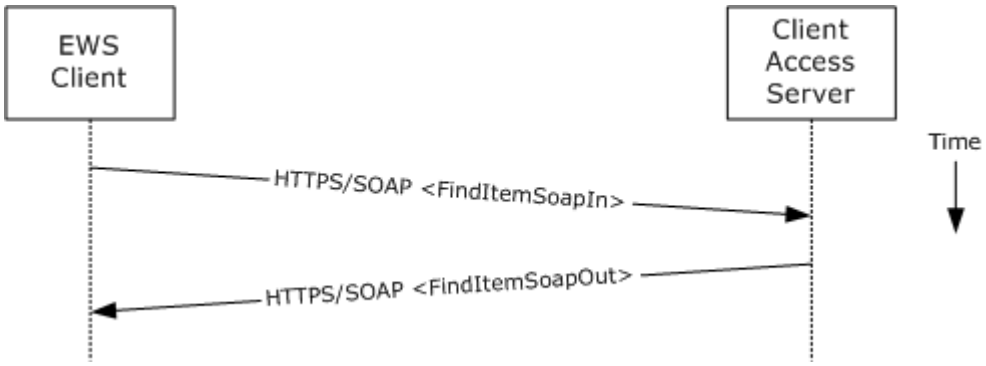

Figure 28:

1. The client uses the HTTPS/SOAP **FindItemSoapIn** request ([MS-OXWSSRCH] section 3.1.4.2.4.1) 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 Exchange Client Access server responds with a **FindItemSoapOut** response ([MS-OXWSSRCH] section 3.1.4.2.4.2), which includes the <ResponseCode> element specifying the status of the operation and the set of items that match the search criteria.

2.4.11 Delete Message(s)

2.4.11.1 Synopsis

This use case describes how a messaging client deletes messages.

2.4.11.2 Builds on Use Case(s)

2.4.9 Open a Folder

2.4.11.3 References

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

2.4.11.4 Requirements

- The client knows the folder ID (FID) or path for the folder in which the message(s) reside.
- The client knows the message ID(s) (MID) or name(s) for the message(s) that are to be deleted.

2.4.11.5 Protocol-specific Details

1. Using RPC:

Figure 29:

   1. The client opens the specified folder as per the previous use case.
   2. The client issues a RopDeleteMessages request with the list of message IDs to be deleted.
   3. The Exchange server returns the success or failure of the operation.

2. Using WebDAV:

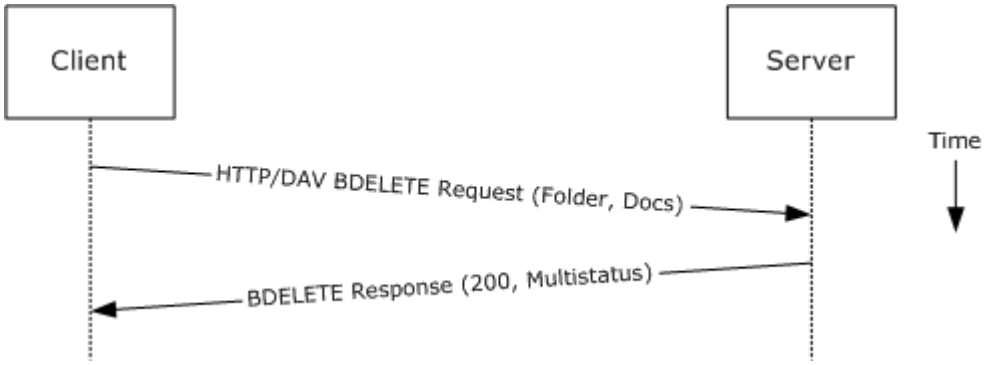

Figure 30:

   1. The client issues a BDELETE request to the Exchange server that references the desired folder path from which document(s) are to be deleted as the URI for the request. The list of documents to be deleted is indicated in the XML body of the request.
   2. The Exchange server responds with a multistatus response, with a separate item indicating the deletion status for each message deletion requested.

3. Using Exchange ActiveSync:

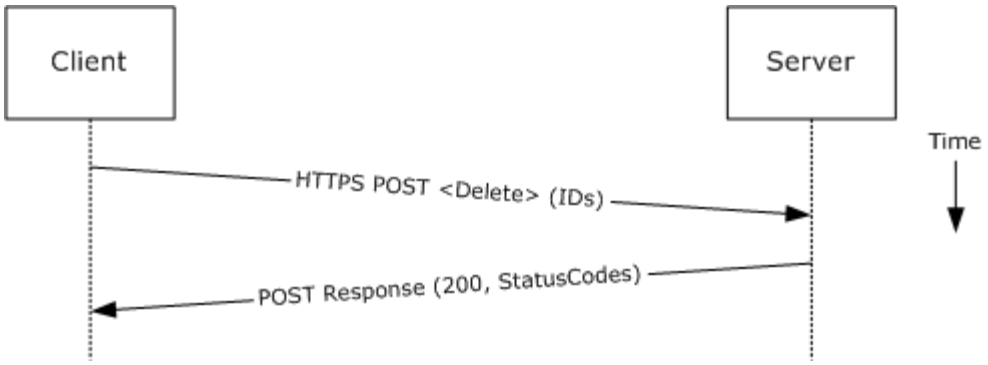

Figure 31:

1. The client issues a sync delete command request to the server, where each item to be deleted is listed in the schematized XML request body.
2. The Exchange server responds with a series of sync status codes, where each corresponds to the deletion status for a message in the deletion list.

4. Using Exchange Web Services:

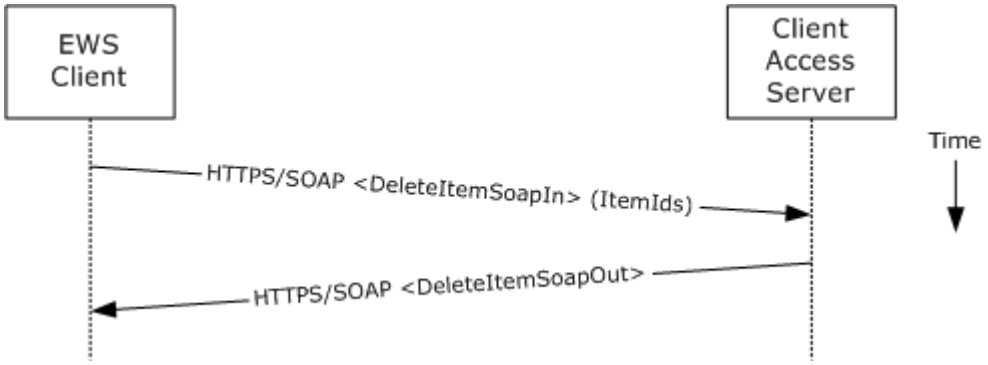

Figure 32:

1. The client uses the HTTPS/SOAP **DeleteItemSoapIn** request ([MS-OXWSMSG] section 3.1.4.4) to delete items specified in the <ItemIds> element.
2. The Exchange Client Access server responds with a **DeleteItemSoapOut** response ([MS-OXWSMSG] section 3.1.4.4), which includes a <ResponseCode> element for the deletion status of each item.

2.4.12 Synchronize Item(s)

2.4.12.1 Synopsis

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

2.4.12.2 Builds on Use Case(s)

2.4.9 Open a Folder

2.4.12.3 References

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

2.4.12.4 Requirements

The client knows the folder ID (FID) or path for the folder to synchronize.

2.4.12.5 Protocol-specific Details

1. Using RPC:

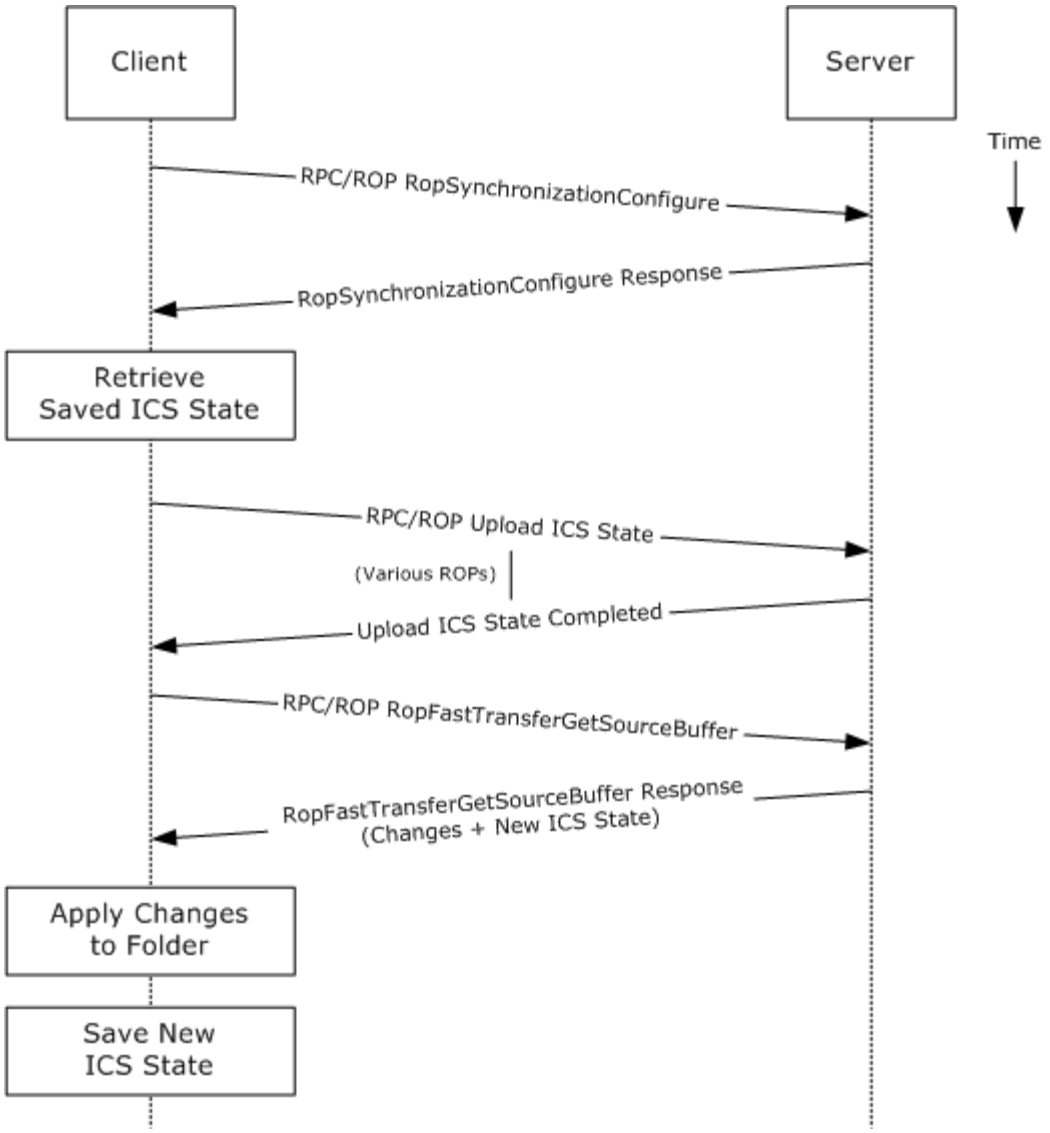

Figure 33:

1. The client opens the specified folder as per the previous use case.
2. The client issues a RopSynchronizationConfigure request 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, RopSynchronizationUploadStateStreamContinue, and RopSynchronizationUploadStateStreamEnd requests to upload the client's Incremental Change Synchronization (ICS) state [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 request 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 persists the new ICS state for subsequent synchronizations.

2. Using WebDAV:

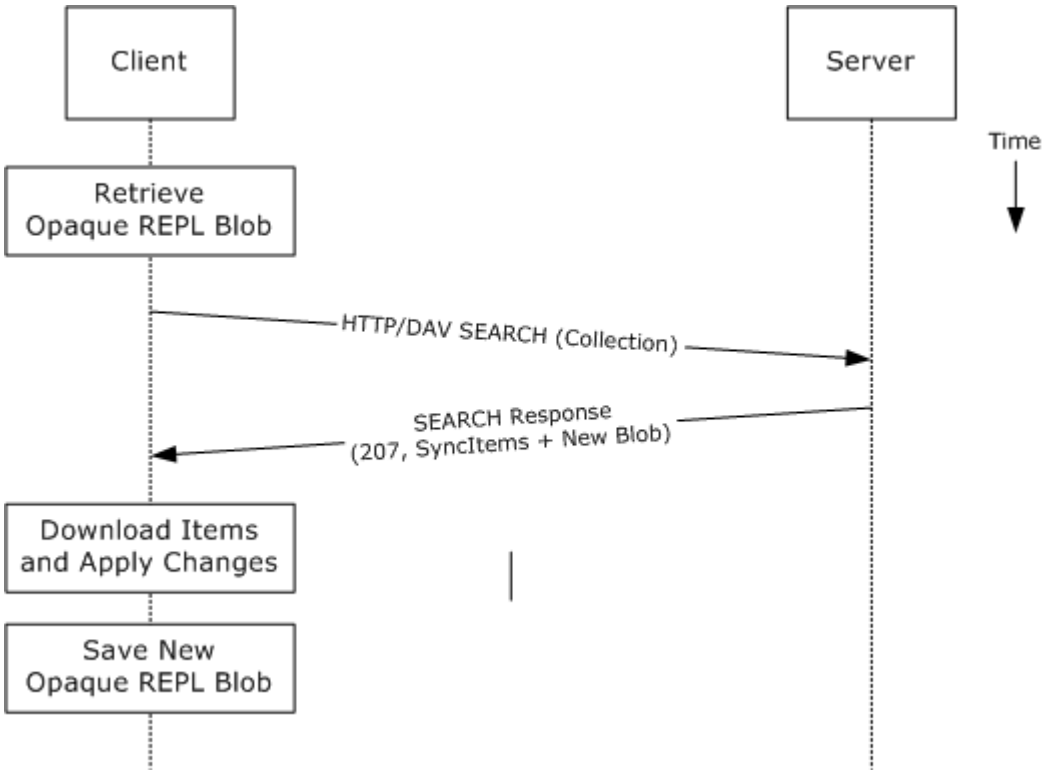

Figure 34:

1. The client issues a SEARCH request for a specific path (with an optional opaque binary large object (blob) that was saved from the last synchronization for this path or an empty blob if synchronizing for the first time) [MS-XWDREPL].

2. The Exchange server responds to the request with the list of changed items, as well as a new blob that corresponds to the current synchronization state.

3. The client downloads each changed item, applies the changes, and persists the new opaque blob for subsequent synchronizations.

3. Using Exchange ActiveSync:

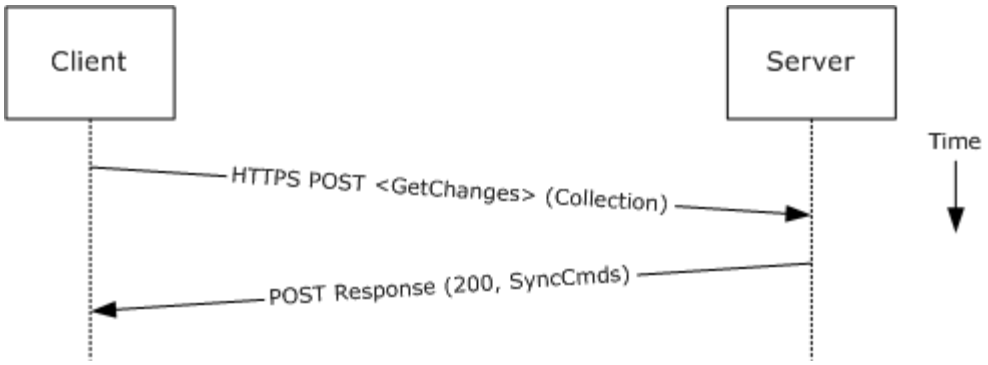

Figure 35:

1. The client issues a Sync GetChanges command request 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 commands that the client processes to synchronize with the Exchange server.

4. Using Exchange Web Services:

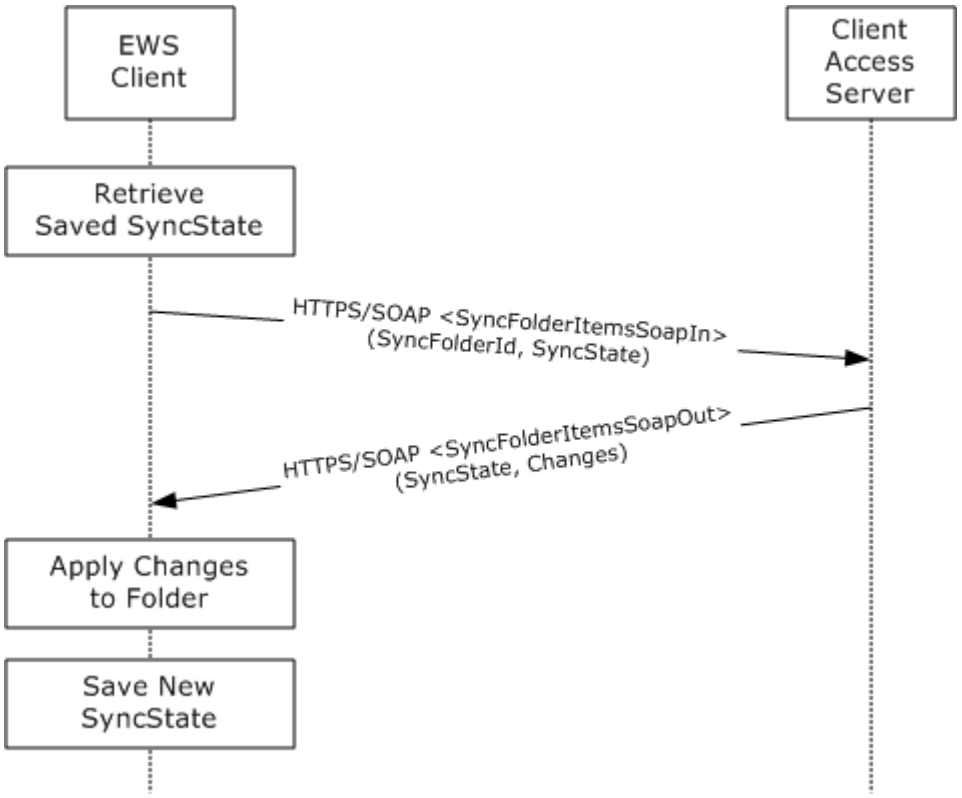

Figure 36:

1. The client uses the HTTPS/SOAP **SyncFolderItemsSoapIn** request ([MS-OXWSSYNC] section 3.1.4.2.4.1) 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 ([MS-OXWSSYNC] section 3.1.4.2.4.2), which includes a <ResponseCode> element, a new <SyncState> marker, and a list of changes.

2.4.13 Register For and Receive Notifications

2.4.13.1 Synopsis

This 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.

2.4.13.2 Builds on Use Case(s)

2.4.2 Log on to a Mailbox

2.4.13.3 References

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

2.4.13.4 Requirements

The client knows the folder ID (FID) or path for the folder that will be used to register server notifications.

2.4.13.5 Protocol-specific Details

1. Using RPC:

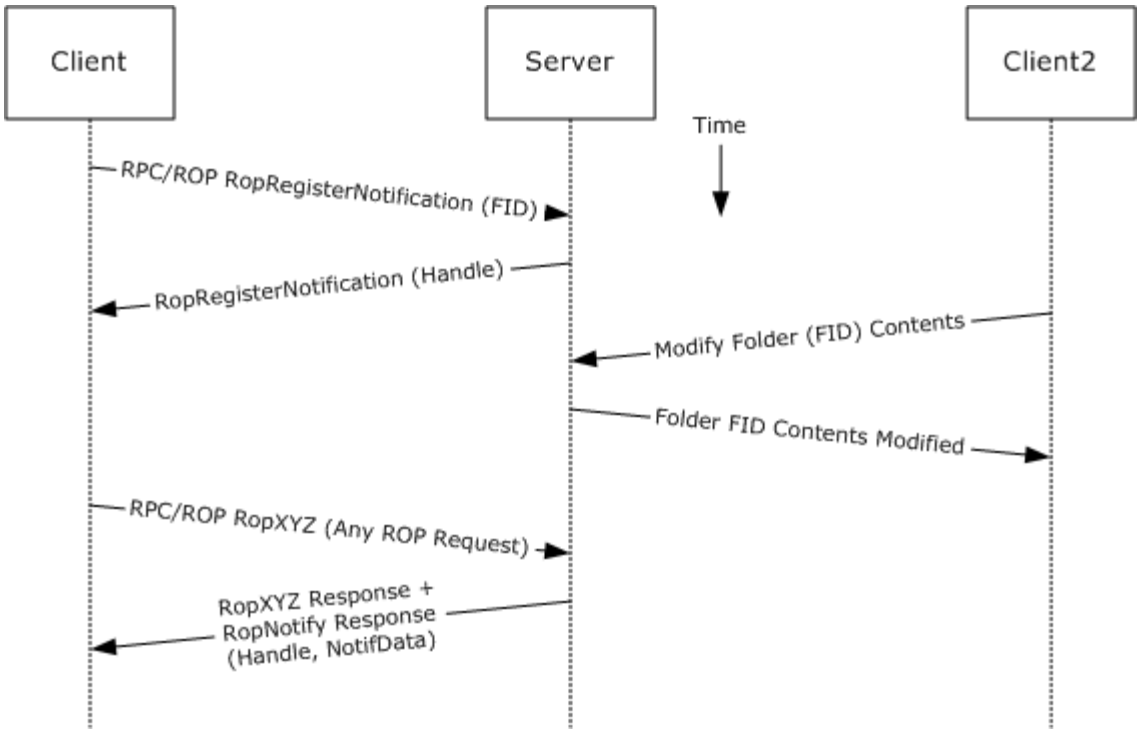

Figure 37:

1. The client logs on to the mailbox as per the previous use case.
2. The client issues a RopRegisterNotification request to register for notifications. 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 response and a result code of RopPending is returned for the overall ROP response.
5. The client examines the information in the RopNotify 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.

2. Using WebDAV:

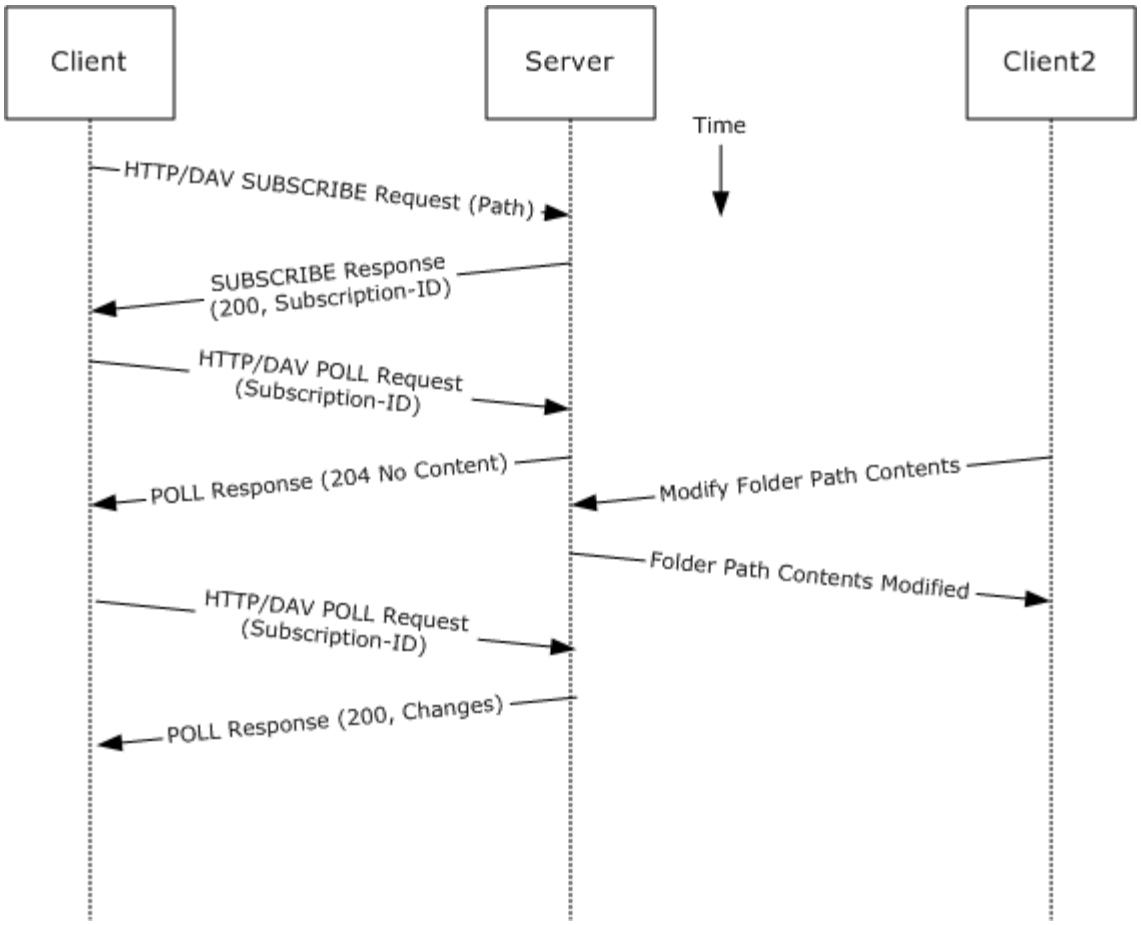

Figure 38:

1. The client issues a SUBSCRIBE command request to the Exchange server, where the URI specified in the HTTP request indicates the resource associated with the subscription.
2. The Exchange server responds with detailed information about the notification subscription, including a subscription ID.
3. The client periodically polls the server for changes using the POLL command request that includes the corresponding subscription ID and URI.
4. The Exchange server returns HTTP Status 204 (No Content) when no events have occurred since the last POLL, or returns a series of response items where each corresponds to an event that has occurred.

3. Using Exchange ActiveSync:

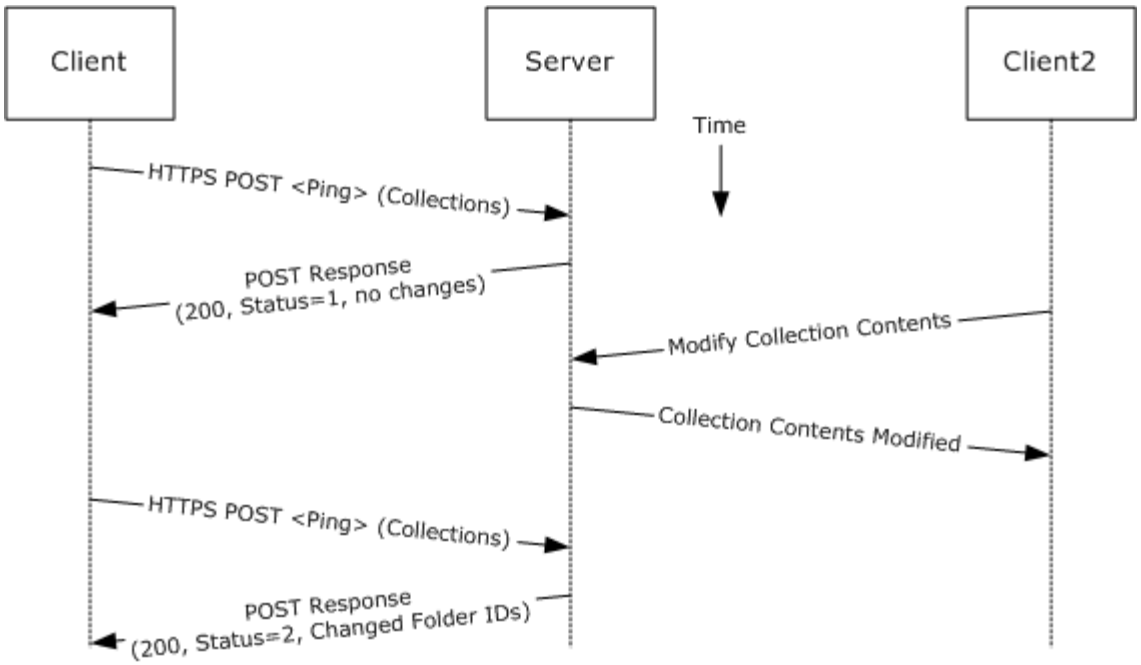

Figure 39:

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, then the Exchange server returns a status code of 1 [MS-ASCMD]. If events have occurred, then the server returns status code 2, followed by a list of folder IDs that need to be synchronized.

4. Using Exchange Web Services:

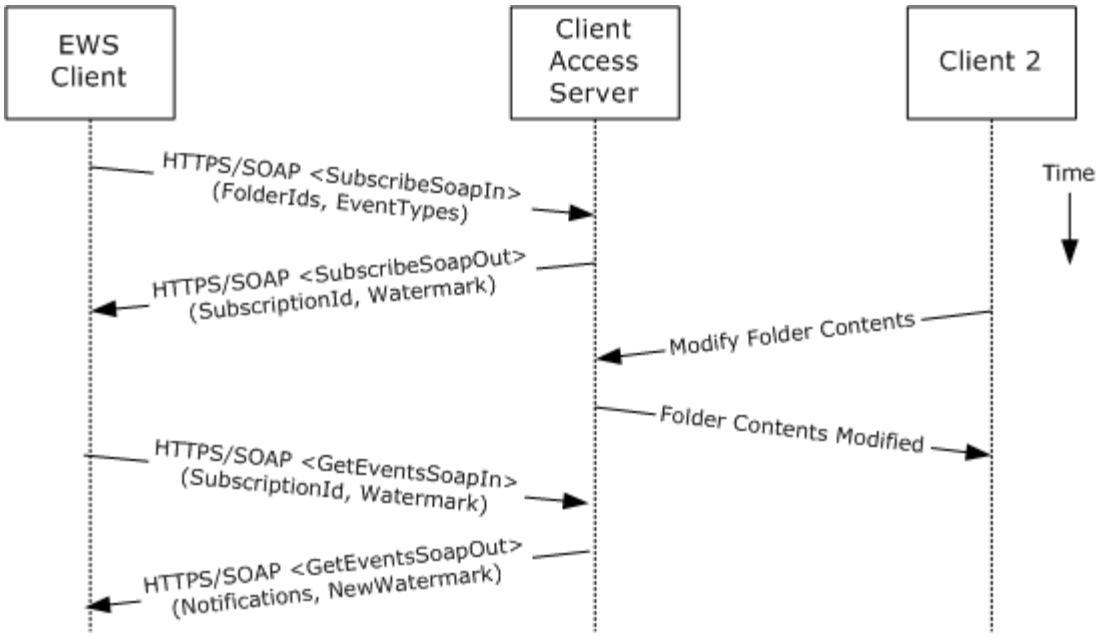

Figure 40:

1. The client uses the HTTPS/SOAP **SubscribeSoapIn** request ([MS-OXWSNTIF] section 3.1.4.3.4.1) 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 Exchange Client Access server responds with a <SubscribeSoapOut> response ([MS-OXWSNTIF] section 3.1.4.3.4.2), 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 ([MS-OXWSNTIF] section 3.1.4.1.3.1) 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 ([MS-OXWSNTIF] section 3.1.4.1.3.2), 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.

2.4.14 Provision a Mobile Client Device

2.4.14.1 Synopsis

This use case describes the multi-phase provisioning process for a mobile client device using Exchange ActiveSync.

Note This use case is specific to Exchange ActiveSync.

2.4.14.2 Build on Use Case(s)

None.

2.4.14.3 References

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

2.4.14.4 Requirements

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

2.4.14.5 Protocol-specific Details

Using Exchange ActiveSync:

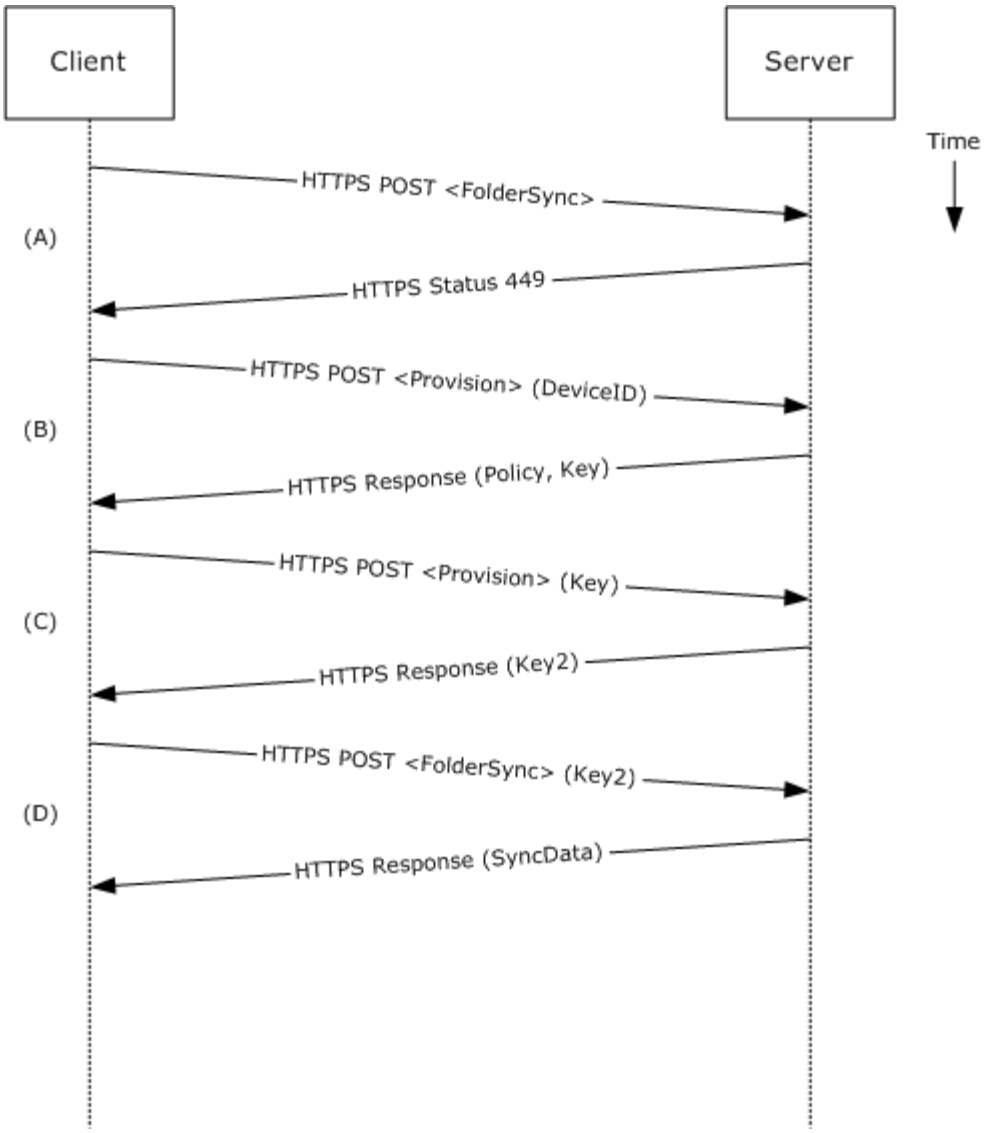

Figure 41:

1. Enforcement:

    1. A client without a current policy attempts to synchronize with an Exchange server (in this case, by issuing a FolderSync request).

    2. The Exchange server replies with HTTP status code 449 (Need Provisioning) indicating that the client's policy is not current.

2. Downloading the Policy:

    1. The client issues a provision request to download the latest policy from the Exchange server, referencing its own device ID for identification.

2. The Exchange server responds with an XML document representation of the current policy plus a policy key.

3. Client Acknowledges Receipt of Policy Settings:

    1. The client issues a FolderSync request to the Exchange server including the most recent policy key in the HTTP headers [MS-ASPROV].

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

2.5 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 chosen by the requesting client. The following sections outline several of the most commonly used mechanisms.

2.5.1 Version Negotiation using RPC

When a client attempts to connect with an Exchange Server via RPC [MS-OXCRPC], the client sends its own version as part of the connection request. Upon successful connect, the server responds with two version numbers: the server version, and the "best" version, where the former indicates the earliest version of Microsoft Exchange Server that the server is compatible with and the latter is the actual server version.

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.

2.5.2 Version Negotiation using WebDAV

A WebDAV client queries the capabilities of the server via the HTTP OPTIONS command [MS-XWDEXT] and by examining the Exchange server response to that command. The server uses a MS-WebStorage header when it supports the WebDAV protocol.

2.5.3 Version Negotiation using Web Services

Web Services clients contact the Exchange server using the industry-standard SOAP 1.1 [SOAP1.1] over HTTP [RFC2616] protocol. The server responds with a ServerVersionInfo XML structure with the server version details in the SOAP response header. The client can then determine the server's capabilities based on this information.

2.5.4 Version Negotiation using Exchange ActiveSync

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

2.6 Error Handling

The following sections provide an overview of the impact of failure for each component in Figure 1: System Overview in section 2.1. In addition, the following sections describe the impact of DNS or the directory service becoming unavailable.

2.6.1 SMTP

SMTP is the key gateway for receiving inbound mail from external servers and non-RPC clients. If inbound SMTP is unavailable, then 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, then 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.

2.6.2 RPC

Due to Exchange-specific implementation details, the RPC protocol is at the heart of the Exchange Information Store. Almost all other protocols are constructed around the RPC protocol and are dependent on it. If the RPC protocol becomes unavailable, then other protocols can experience service outages related to accessing storage items.

2.6.3 Web Services

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

2.6.4 POP3

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

2.6.5 IMAP4

If IMAP4 becomes unavailable, then e-mail clients using IMAP4 to access their mailbox will not being able to do so.

2.6.6 WebDAV

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

2.6.7 NSPI

If NSPI becomes unavailable, then operations related to address book lookup and recipient resolution will be affected. However, clients using Offline Address Books [MS-OXOAB] will still be able to perform recipient lookups and resolutions for the contacts listed in the local OAB.

2.6.8 Unified Messaging

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

2.6.9 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 (e.g. Web Access) to access their mailboxes until Exchange ActiveSync is available again.

2.6.10 DNS

If DNS becomes unavailable, then outbound e-mail routing will be impacted as the remote domain address cannot be resolved. Since Exchange Server Autodiscover [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.

2.6.11 Directory Service

The directory service is responsible for storing mail recipient information and is a primary mechanism for Autodiscover. If the directory service becomes unavailable, then NSPI features and Autodiscover will be unavailable.

2.7 Coherency Requirements

None.

2.8 Security

For a comprehensive and current discussion about security and protection for Microsoft Exchange Server, see Microsoft TechNet: Microsoft Exchange Server 2007 Security and Protection.

3 Examples

This section extends section 2.4 use cases by illustrating how one or more use cases can be combined to achieve specific end-user results. Unlike the use cases section, which focuses on abstract communications between general entities (e.g. "server" and "client"), this section delves into details that are specific to an Exchange server implementation (such as special folder IDs, and specific message classes and properties, etc.).

The examples are meant to showcase how the use cases presented can be applied to 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 easily applied to other similar real life scenarios.

In order to reduce the complexity of the examples, each example is demonstrated using the protocol (e.g. RPC or Exchange Active Sync) that is most commonly used for 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 to avoid duplication of information.

3.1 Display the Most Recent Message in the Inbox

3.1.1 Synopsis

This example illustrates how a series of use cases can be used to access an inbox and display the newest message in it.

Note This example uses the RPC protocol.

3.1.2 Use Case(s)

- 2.4.1 Server Information Discovery
- 2.4.2 Log on to a Mailbox
- 2.4.9 Open a Folder
- 2.4.10 Find Items in a Folder that Match Search Criteria

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

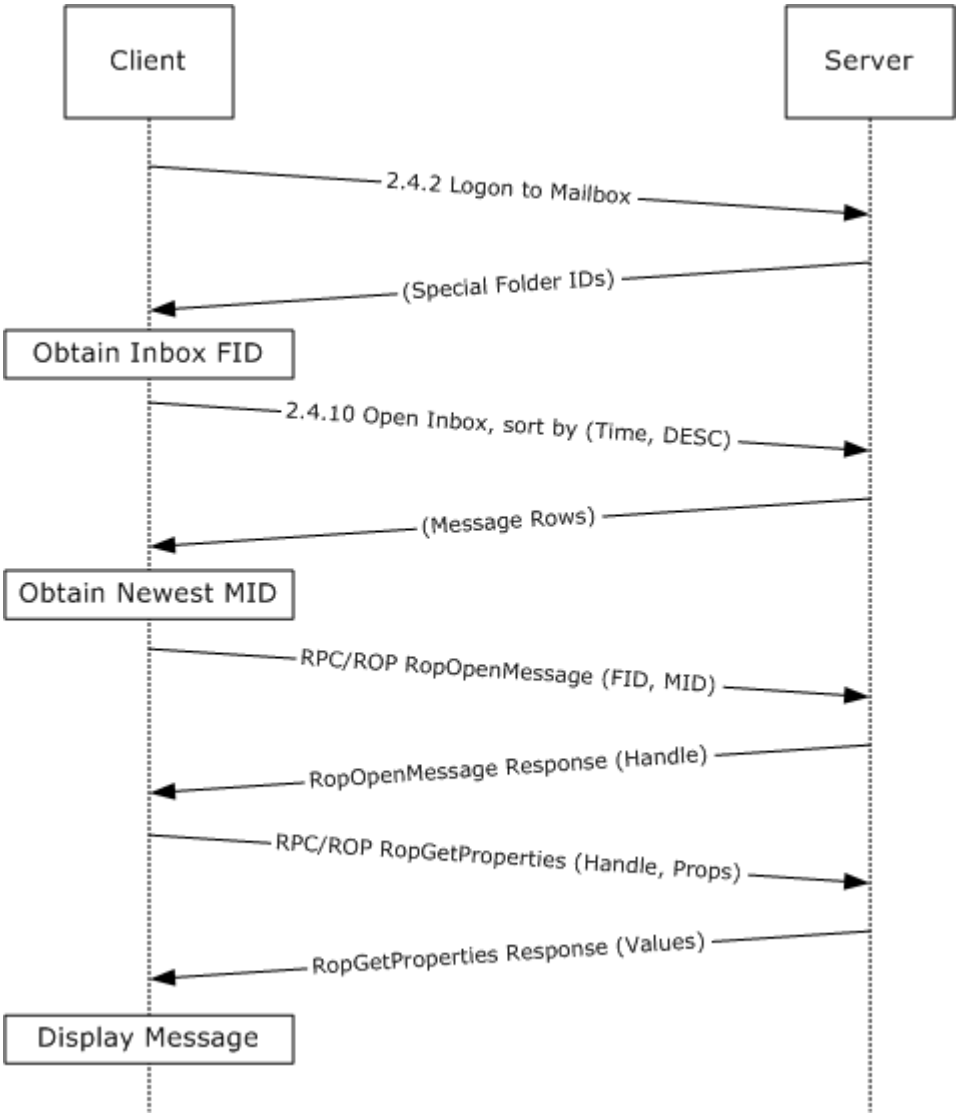

Figure 42: Display message steps

1. The client logs on to the mailbox per use case 2.4.2.
2. The client obtains the Folder ID (FID) for the inbox from the server as a result of successful logon.
3. Based on use case 2.4.10, the client queries the message rows in the inbox, and sorts the rows by message date (PidTagMessageDeliveryTime [MS-OXPROPS]) in descending order. The resulting table contains a column for the Message ID (PidTagMid [MS-OXPROPS]).

   Note A restriction can be used or not in this case.
4. The first row in the inbox contents table contains information about the newest message. Extract its Message ID (MID) from the PidTagMid column.

5. The client issues a RopOpenMessage [MS-OXPROPS] with the inbox FID and the MID returned. The Exchange server returns a handle to the opened message.

6. The client issues a RopGetProperties [MS-OXPROPS] with the message handle to retrieve the body of the message (PidTagBody [MS-OXPROPS]) 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 Compose and Send an E-mail Message with an Attachment

3.2.1 Synopsis

This example illustrates how a series of use cases can be used to enable an end-user to compose and send an e-mail.

Note This example uses the RPC protocol.

3.2.2 Use Case(s)

- 2.4.1 Server Information Discovery
- 2.4.2 Log on to a Mailbox
- 2.4.3 Create a Message
- 2.4.5 Add an Attachment
- 2.4.6 Resolve a Recipient from an Address Book
- 2.4.7 Send a Message

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 use case 2.4.2.

2. The client creates a new message in the drafts folder per use case 2.4.3.

3. An end user enters the name of the message recipient and chooses to "Resolve Recipient".

4. The client resolves the recipient per use case 2.4.6. 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, then 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) [MS-NSPI].

6. The end user chooses to add an attachment to the message.

7. The client loads the attachment data and creates a new attachment per use case 2.4.5.

8. The end user enters a body for the message and chooses to send the message.

9. The client submits the message per use case 2.4.7 to the Exchange server.

3.3 Set up and Displaying New Mail Notifications

3.3.1 Synopsis

This example illustrates how a client can build on use cases to register itself for new mail notifications.

Note This example uses the RPC protocol.

3.3.2 Use Case(s)

- 2.4.1 Server Information Discovery
- 2.4.2 Log onto a Mailbox
- 2.4.13 Register For and Receive Notifications

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

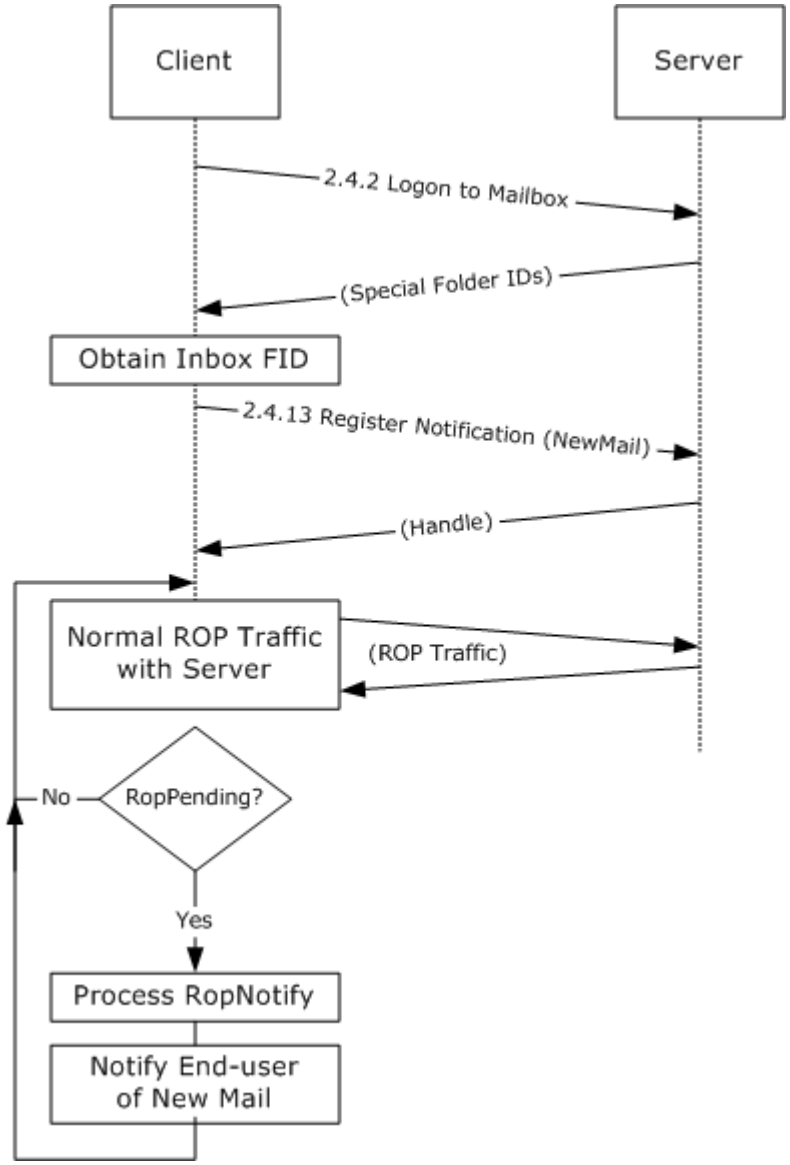

Figure 43:

1. The client logs onto the mailbox per use case 2.4.2.
2. The client obtains the Folder ID (FID) for the inbox from the Exchange server as a result of successful logon.
3. The client registers for notifications for new mail [MS-OXCNOTIF] by referencing the inbox's FID per use case 2.4.13.
4. The client continuously monitors for RopPending result codes from ROP responses as well as RopNotify responses in the ROP Response payload for notifications. The client will take appropriate action (e.g. play a sound) when a new mail notification is received.

3.4 Create an Appointment Request using Free-busy Data

3.4.1 Synopsis

This example illustrates how a series of use cases 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 Case(s)

- 2.4.1 Server Discovery Information
- 2.4.2 Log onto a mailbox
- 2.4.3 Create a Message
- 2.4.4 Create a Strongly-typed Message
- 2.4.6 Resolve a Recipient from an Address Book
- 2.4.7 Send a Message
- 2.4.9 Open a Folder
- 2.4.10 Finding Items in a Folder that Match Search Criteria

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 (see [MS-OXCSTOR]) and a drafts folder to store unsent messages.
- The invitee is listed in an address book.

3.4.5 Details

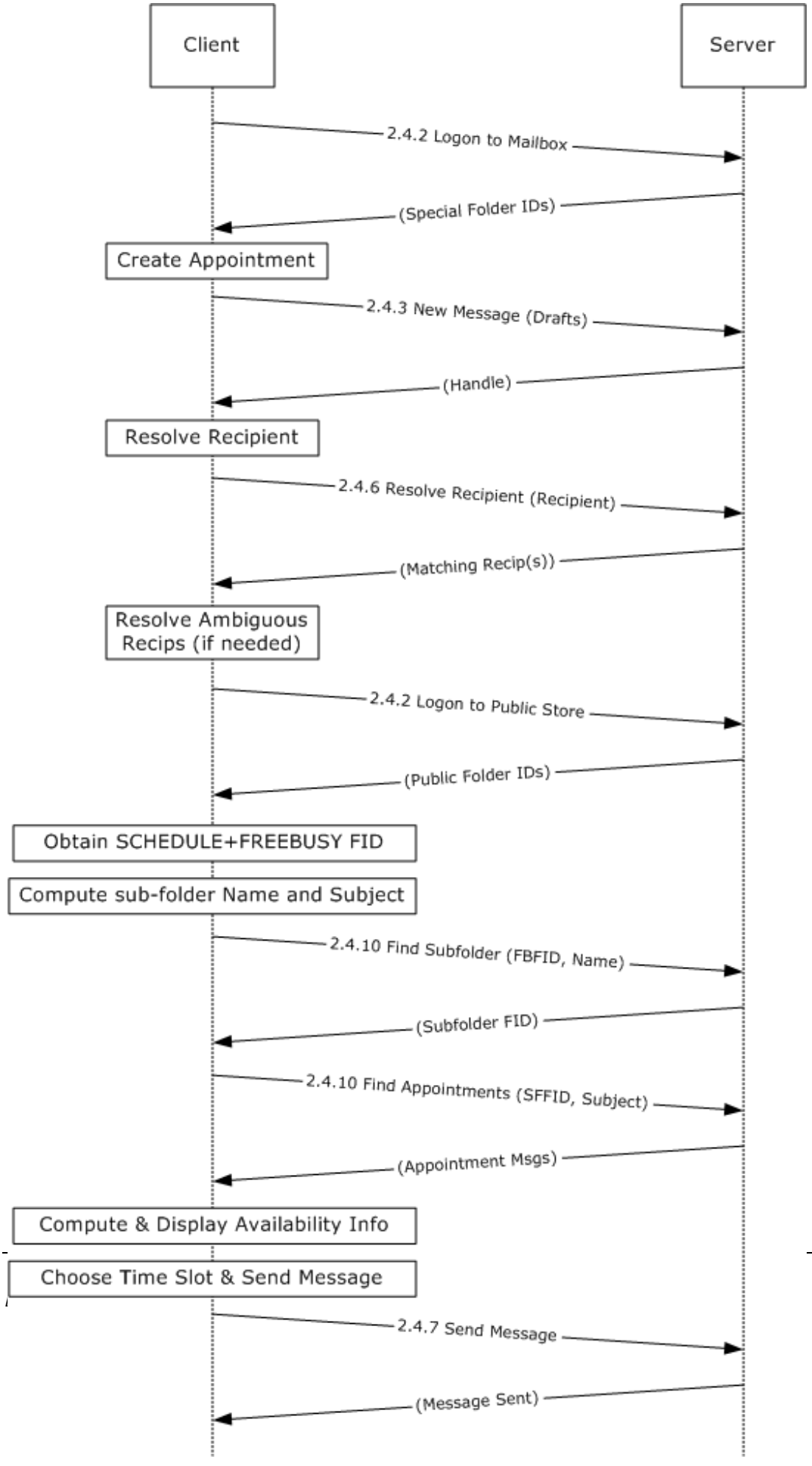
Client
Server
2.4.2 Logon to Mailbox
(Special Folder IDs)
Create Appointment
2.4.3 New Message (Drafts)
(Handle)
Resolve Recipient
2.4.6 Resolve Recipient (Recipient)
(Matching Recip(s))
Resolve Ambiguous Recips (if needed)
2.4.2 Logon to Public Store
(Public Folder IDs)
Obtain SCHEDULE+FREEBUSY FID
Compute sub-folder Name and Subject
2.4.10 Find Subfolder (FBFID, Name)
(Subfolder FID)
2.4.10 Find Appointments (SFFID, Subject)
(Appointment Msgs)
Compute & Display Availability Info
Choose Time Slot & Send Message
2.4.7 Send Message
(Message Sent)

Figure 44:

1. The client logs onto the mailbox per use case 2.4.2.
2. An end user chooses to create a new appointment.
3. The client creates a new message in the drafts folder per use case 2.4.3. The Exchange server returns a handle to the new message.
4. The end user enters the name of the invitee and chooses to "Resolve Recipient".
5. The client resolves the recipient per use case 2.4.6. The server returns the successful matches (or none if there are no matches).
6. If the Exchange server returns more than a single match, then 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) [MS-NSPI].
7. The client logs on to the public store [MS-OXCSTOR] (using steps similar to use case 2.4.2).
8. The Exchange server returns a list of special folder IDs upon successful logon. The client obtains the Folder ID (FID) of the "SCHEDULE+FREEBUSY" folder [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 [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 use case 2.4.9.
11. The client performs another folder search in the subfolder to find all appointment messages whose subject (PidTagSubject [MS-OXPROPS]) matches the computed subject in step 9 per use case 2.4.10.
12. The client scans the appointment messages to calculate the invitee's availability and presents it 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 chooses to send the invite.
15. The client submits the message per use case 2.4.7.

3.5 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

- 2.4.12 Synchronize Item(s)

- 2.4.14 Provision a Mobile Device

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

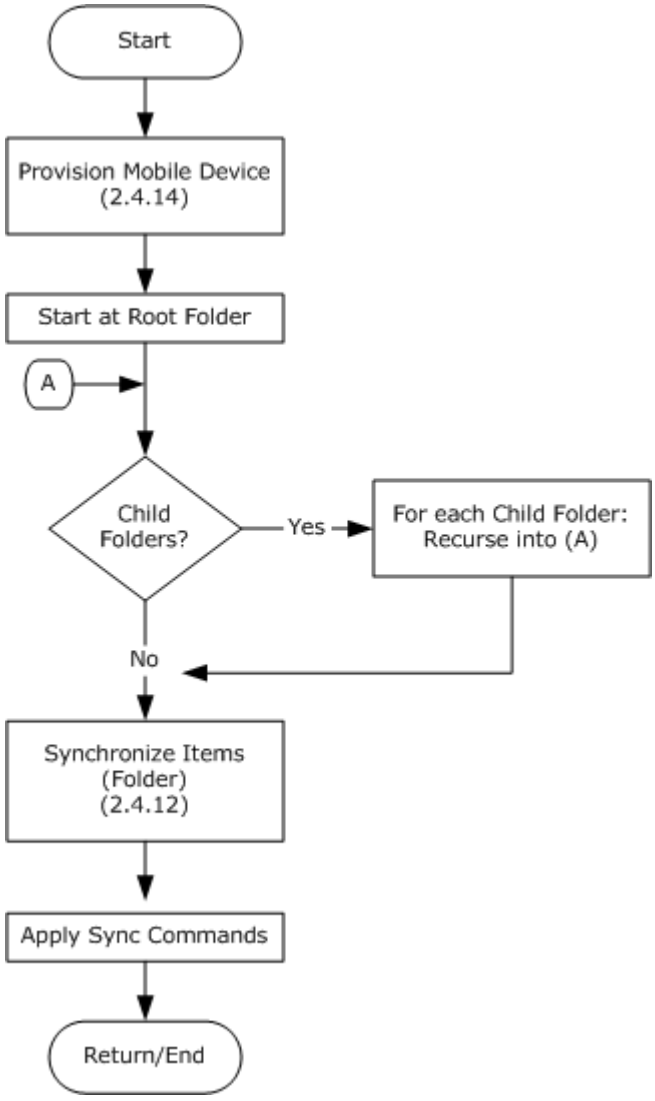

Figure 45:

1. The client undergoes the provisioning process per use case 2.4.14 to obtain the current policy and policy key.

   Note At this point, the FolderSync command 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 use case 2.4.12.

4 Microsoft Implementations

The information in this specification is applicable to the following versions of Office/Exchange:

- Microsoft® Office Outlook® 2003
- Microsoft® Exchange Server 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Exchange Server 2007
- Microsoft® Outlook® 2010
- Microsoft® Exchange Server 2010

5 Change Tracking

This section identifies changes that were made to the [MS-OXPROTO] protocol document between the May 2010 and August 2010 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

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

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

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.1 Glossary	56243 Added "base64 encoding" to the list of terms defined in [MS-OXGLOS].	N	New content added.
1.1 Glossary	56956 Removed "folder associated information (FAI)", "IPM", "named property", "normal message", and "PIM" from the list of terms defined in [MS-OXGLOS]. Specified definition number (1) for the term "property".	N	Content update.
1.2 References	55480 Removed unused reference [RFC2119].	N	Content removed.

6 Index