[MS-OXPROTO]: Exchange Server Protocols Overview

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights.** This protocol 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 protocols, and may distribute portions of it in your implementations of the protocols 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 protocol documentation.
- **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 protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@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.

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. This protocol documentation is 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. A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability.
Microsoft Corporation	April 25, 2008	0.2	Revised and updated property names and other technical content.
Microsoft Corporation	June 27, 2008	1.0	Initial Release.
Microsoft Corporation	August 6, 2008	1.01	Updated references to reflect date of initial release.
Microsoft Corporation	September 3, 2008	1.02	Changed title of document.
Microsoft	December	1.03	Revised and edited technical content.

Corporation	3, 2008		
Microsoft Corporation	February 4, 2009	1.04	Revised and edited technical content.
Microsoft Corporation	March 4, 2009	1.05	Revised and edited technical content.

Table of Contents

Introduction

The Office Exchange Protocols (OEP) specify how mail clients and servers can communicate and store information related to e-mail, calendars, contacts, voice mail, task tracking and other user collaboration functionality.

This document is a companion to the protocol, data structure and file formats that are included in the Office Exchange protocol documents.

The sections below list definitions, protocol stacks, client configurations, and functional areas referred to throughout the rest of this overview.

1.1 Definitions

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

Address Book object
Attachment object
folder
folder associated information (FAI)
Folder object
IPM
message
Message object
named property
normal message
PIM
property
public folder
search folder
special folder

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

1.2 References

1.2.1 Normative References

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

[MS-RPCH] Microsoft Corporation, "Remote Procedure Call Over HTTP Protocol Specification", July 2006, http://go.microsoft.com/fwlink/?LinkId=121108.

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

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

[WSIBASIC] "Web Services Interoperability (WS-I) Organization Basic Profile Version 1.0"., April 2004, http://www.ws-i.org/Profiles/BasicProfile-1.0.html.

1.2.2 Informative References

None.

2 System Architecture

The Office Exchange Protocols<1> are designed around a common conceptual model for messaging and directory information that is described in this section.

2.1 Data Model

2.1.1 Common Concepts

While protocols dealing with directory versus messaging information are significantly different because of their disparate purposes, there are important similarities between their exposed data models, which are explained below.

2.1.1.1 Object Orientation

The data models for both directory and messaging protocols are object-oriented.

- The data model exposed for directory information consists of container objects, address list objects, a number of different types of mail-enabled **Address Book objects** (mailbox, public folder, distribution list, contact, conference room, and resource) and user-interface templates that are used by the client to present the Address Book objects.
- The data model for messaging consists of the **message** store, **folder**, message, and **Attachment objects**.

2.1.1.2 Property Model

The directory and messaging data models structure the data on objects as properties. An object's **property** consists of a 32-bit unsigned property tag and a typed property value. Properties can be either single or multivalued (accessed as a list of values) and support a large variety of types, such as string, integer, Boolean and so on.

The Office Exchange Protocol documentation set defines standard property tags to represent both messaging and directory data. From a documentation and coding standpoint, these property tags are usually referred to using a literal name such as **PidTagSubject** or **PidTagEmailAddress**.

For example, the subject of a **message** would be accessed as the **PidTagSubject** property (ID 0x0037) of the **Message object**, and the e-mail address of an address book entry would be accessed as the **PidTagEmailAddress1** property (ID 0x8093). Both properties are single-valued string properties, type 0x001F, or Unicode (UTF-16LE) string.

The Master Property List described in [MS-OXPROPS] contains the comprehensive set of properties, types and related structures and errors.

There are two additional property model features which the server SHOULD support for messaging information, but need not support for directory information.

- **Named Properties.** A mechanism adding properties that are not part of the standard set of Office Exchange Protocol properties, that the client can use to support custom forms and data types.
- **Streaming of long values.** A method for retrieving and updating long text and binary properties such as the body of a message or the data of an attachment.

The Property and Stream Object Protocol described in [MS-OXCPRPT] provides these features and includes getting, setting and deleting properties.

2.1.1.3 Interobject Relationships

There are similarities and differences in Office Exchange Protocol's modeling of interobject relationships.

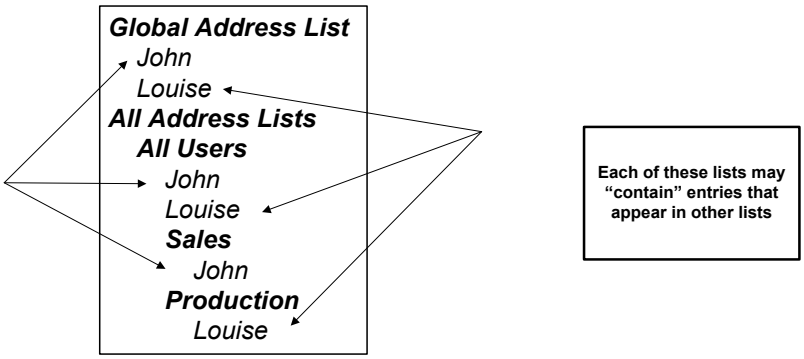

Figure 1: Hierarchy and containment in Address Book NSPI

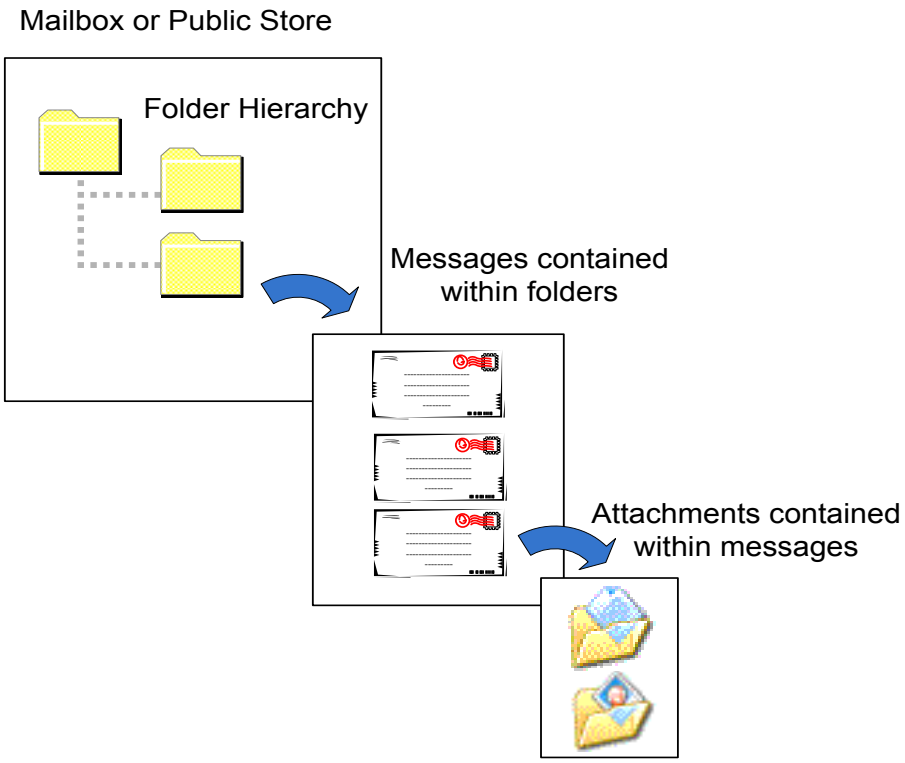

Figure 2: Hierarchy and Containment in a PIM and object store

2.1.1.3.1 Hierarchy

The directory and messaging data models both expose the following hierarchy.

- The directory data model exposes containers in a forest of tree-structured hierarchies. When it connects to the Address Book, the client opens the root container of the tree, and navigates

down the hierarchy one level at a time to find the Global Address List container and Address List container objects.
- The messaging data model stores the **Folder objects** belonging to a single mailbox or public folders in a single tree-structured hierarchy. **Message** and **Attachment objects** are not part of the hierarchy.

2.1.1.3.2 Containment

The directory and messaging data models also expose the following containment.

- In the directory data model, a Global Address List or address list object is said to "contain" the list of mail-enabled objects that the client displays when the user selects it for viewing. Similarly, a distribution list object is said to contain its members. In neither case is a parent-child relationship implied.
- In the messaging data model, **folders** can "contain" **Message objects**, and **messages** can "contain" attachments. A message can only be in one folder; an attachment can only be in one message.

> ***Note:*** There is an exception to this rule: a user can create a search folder in their mailbox and that folder will appear to "contain" messages which match the specified search criteria but are actually located in other folders in the user's store.

2.1.1.4 Table Model

The Office Exchange Protocols specify a common model for browsing and searching the following row-based data, as specified in [MS-OXCTABL].

Hierarchy Tables

- **Folders** in the hierarchy beneath a given folder
- Object hierarchy below a directory object

Contents Tables

- **Messages** contained within a given folder
- Contents of the global address list or address list object
- Members of a distribution list

Other tables

- Attachments on a message

The client receives the data from the server in tabular form, where each row is an individual item (folder, message, recipient, etc) and the columns are **property** values.

The client can control the list of columns to return, the list of columns to sort by, and, if desired, specify criteria restricting which rows are returned. The types of sorts and restriction criteria supported are limited for many of the table types, but this is not the case for folder contents tables which allow virtually any type of sorting (including categorized sorts) and restriction.

The Table Object Protocol [MS-OXCTABL] makes it possible for the client to display scrollable, tabular views in such a way that the client is only required to download the information needed to display the current screen view. Table objects can also support client implementations of efficient type-ahead search and scrollbar positions.

2.1.2 Directory Data Model Specifics

The Address Book NSPI data model is a replicated forest of directory objects, although only the Global Address List, address list, and mail-enabled **Address Book object** are of interest to the Office Exchange Protocols.

Each object is named by an LDAP-style X.500 distinguished name (DN), although at the protocol level there are also other ID's that are used to specify objects.

The Office Exchange Protocol directory model provides the following functionality for the client.

- Provides pick lists for e-mail recipients and scheduling, including the Global Address List and additional address list subsets which the client user can browse. These address lists contain mail-enabled entries of the following types:
    - Mailbox entry*:* When an e-mail **message** is sent to a mailbox entry, the server will route that message to the appropriate server instance and deliver it to the delivery **folder** (typically the Inbox) of the mailbox store corresponding to this entry.
    - Public folder entry*:* The server can allow certain public folders to be mail-enabled, which means that you can include the public folder as a recipient on the e-mail address. In this case the server will deliver the message to that public folder.
    - Distribution list entry*:* This type of entry consists of a list of mail-enabled entries. When an e-mail message is sent to a distribution list, the server will recursively expand the distribution list and route a copy of the message to each entry on the list.
    - Contact entry*:* These entries represent e-mail addresses on other systems to which the server can route mail. The server will transfer a copy of messages sent to such addresses to the specified foreign or disconnected e-mail system.

- Resource entry*:* A special type of mailbox to represent resources that can be scheduled.
- Room entry*:* A special type of mailbox to represent rooms that can be scheduled.
- Mail agent entry*:* Any other mail-enabled entry that is not one of the above types.

- Provides what's necessary for the client to display details on an Address Book object: When the user selects an object from an address list to see the properties of that object, the Address Book not only provides the values of the different properties, it also returns the display template (arrangement of fields on the dialog) that the client SHOULD use.

Performs ambiguous name resolution (ANR): When a user types recipients into the To, Cc, or Bcc fields of a message they are composing (as opposed to choosing the recipient from an address list), the directory provides the efficient searching support to find the entries that best match the given recipient names.

2.1.3 Messaging Data Model Specifics

The top-level data object of the messaging data model is a store. There are two kinds of stores– mailbox store and public folders store. Each can contain **folders**, which can in turn contain **messages**, which can in turn contain attachments.

2.1.3.1 Mailbox Store

A mailbox store consists of three types of data.

- A limited, non-extensible set of properties that contain user settings.
- A hierarchical set of **folders** belonging to the mailbox.
- Optionally, read/unread information for public folder messages the user has viewed.

The mailbox folder hierarchy includes a number of standard folders, that are created and maintained by the server, as well as user-created folders.

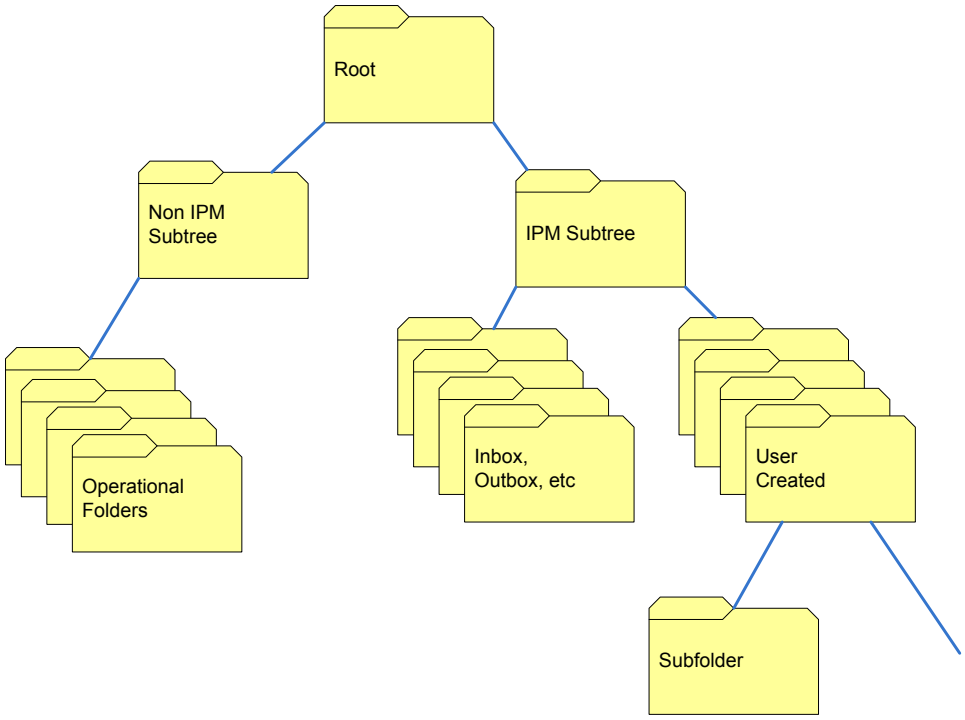

Figure 3: Mailbox Store Hierarchy

Below the root of the store are two subfolders–the **IPM** Subtree and the Non-IPM Subtree. The folders below the IPM Subtree would be expected to be those displayed by the client, while the folders below the Non-IPM Subtree are operational folders.

2.1.3.2 The Public Folders Store

In order for a client to present the public folders store to the user as a single tree of **folders**, it connects to a specific server instance to retrieve the hierarchy information. Every server instance that hosts the public folders store has knowledge of the complete public folder hierarchy (including their names and the parent-child relationships).

When the client needs to display the contents of a particular public folder, it will attempt to get the folder's contents from the hierarchy server. If that server instance does not store the folder's contents, then it can get a list of server instances which do contain a copy of the folder. The client connects to each server in the list until it is able to locate one that contains the contents of the folder.

The public folder hierarchy has a structure similar to what's found in mailbox stores. Under the root directory, there are, again, two subfolders–**IPM** Subtree and Non-IPM Subtree. In this case,

all of the folders under the IPM Subtree are created by users. There are two operational folder hierarchies under the Non-IPM Subtree which are especially important to this overview because they are where a client can download offline address book updates and query for free/busy information.

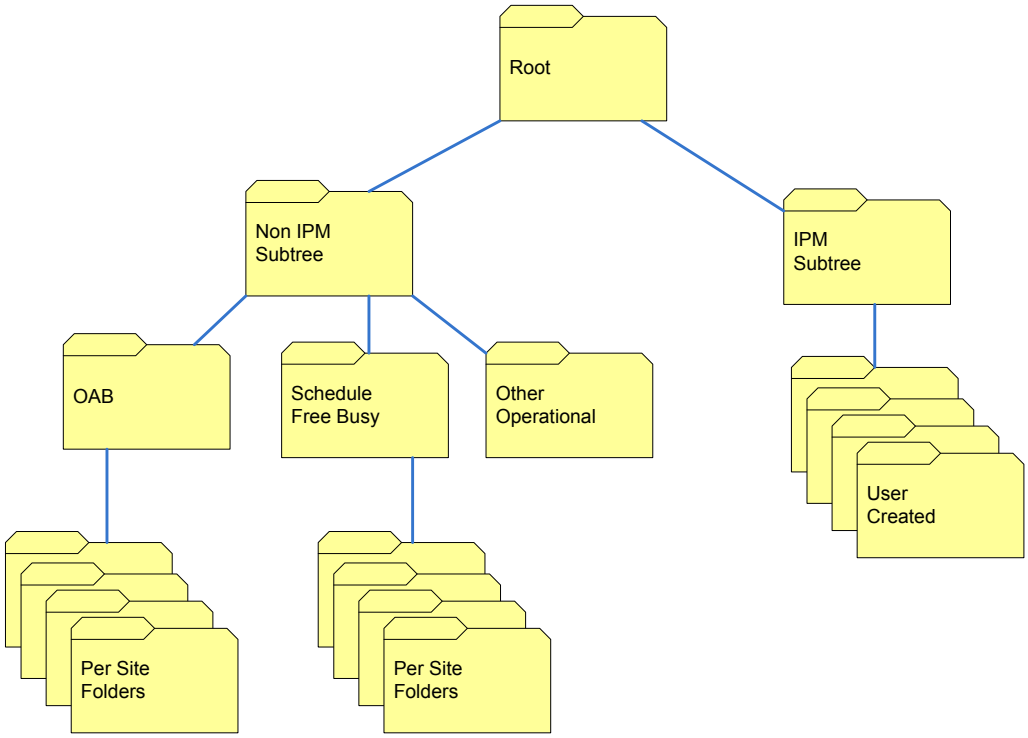

Figure 4: Public Folder Store Hierarchy

2.1.3.3 Folders

A **folder** contains **messages** (contents) and subfolders (hierarchy).

In addition, the following things are associated with a folder object.

- Well-known properties (for example, **PidTagDisplayName**)
- Certain computed properties (for example, **PidTagContentCount**, **PidTagContentUnreadCount**)
- Security descriptor for controlling access to the folder
- Rules that apply to messages which are created in or delivered to the folder

For public folder-based folder objects, the computed properties **PidTagContentCount** and **PidTagContentUnreadCount** cannot be obtained from the folder hierarchy, they need to be obtained from the folder. To compute **PidTagContentUnreadCount** for a public folder, the

client might need to access associated read/unread information stored in the mailbox store and send it to the server hosting the folder for the computation.

2.1.3.4 Messages

The term **message** is a broad one. In addition to representing e-mail messages, by containing a specific set of **property** values, core **Message objects** can also represent appointments on a schedule, tasks, contacts, notes, journal items, and even forms and views.

Messages are divided into two groups depending on whether the client displays the message or uses it to store operational information (for example a form). A message that the client displays is called a **normal message**, while a message used for operational purposes is called a **folder associated information (FAI)** message. When requesting the contents table for a folder, the client specifies whether they want a contents table of normal messages or a contents table containing the FAI messages.

The most important message property is the message class (**PidTagMessageClass**). Message class is a required string type property which indicates the intended use for the message. The value of the message class determines the following:

- The form the client uses to display the message
- The properties the client stores on the message

The following table gives examples of message class values and what kind of message they represent.

Message class	Purpose
IPM.Note	Standard e-mail message
IPM.Post	Standard message posted to a public folder
IPM.Appointment	Appointment on a schedule
IPM.Task	Task
IPM.Contact	Contact
IPM.StickyNote	Sticky Note
IPM.Activity	Journal Item
REPORT.IPM.Note.DR	Delivery report
REPORT.IPM.Note.NDR	Non delivery report
REPORT.IPM.Note.IPNRN	Read notification
REPORT.IPM.Note.IPNNRN	Non-read notification

This is not a full list of the common message classes in use and, in any case, the list of message class values is not fixed. When a data object type is created, a new message class value is generally created to distinguish messages that are associated with that data object type, and potentially with a type-specific form. The set of objects that are transported via the Office Exchange Protocol are listed in the Object protocols section.

It is beyond the scope of this overview document to discuss all message properties, but for illustrative purposes, some e-mail message properties are described below.

- **From***:* The sender of the message does not need to be set by the client when sending a new message except in delegation scenarios. To send as or on behalf of another mail-enabled directory object, special properties are set on the message. When the message is received, the sender and sent on behalf information is available as properties of the message.
- **To, CC,** and **BCC***:* The recipients of a message are not really objects, rather each recipient is described by a set of properties such as address type, e-mail address, etc. The properties are set using a special data structure, and can be retrieved using a table interface.
- **Subject***:* The subject is stored as a **property** of a message. In order to prevent successive replies accumulating a multiplicity of "RE:" or "FW:" prefixes, there are two properties– **PidTagSubject** and **PidTagNormalizedSubject**, where the former is the full subject line, and the latter is without a reply or forwarding prefix.
- **Body***:* The contents of the message are contained within one of three possible properties, which MAY be streamed when any of them are set or retrieved.
- **Attachments***:* The attachments on a message are GET and SET as separate objects with a related set of properties.

2.1.3.5 Attachments

Attachments can be placed on **messages**. Like stores, messages, and **folders**, the **Attachment object** supports a **property** interface, although the number of properties used in the Office Exchange protocol is very limited.

The attachments can be files or messages. By attaching a message which itself has attachments, nesting of attachments can occur.

2.2 Data Stores

The following table lists the main data repositories accessed by a client relating to its operation against a server.

Store Name	Purpose	Required?
Mailbox	Stores "**message**" items (e-mail, appointments, contacts, tasks, etc.) in a set of hierarchical **folders** that belong to a mailbox object. The mailbox object is associated with a corresponding Address Book mailbox object. Additional features include access control, asynchronous e-mail delivery, content type conversion, rules, and persistent search folders.	Yes
Public Folders	Virtual, replicated store of message items in hierarchical folders. Servers in the organization that have a Public Store maintain a replicated copy of the folder hierarchy. Not all servers store the contents of every folder, and as a result, a client might have to connect to multiple servers to satisfy a user request to view information from a folder.	Depends on client configuration
Global Address List	An organization-wide list of all mail-enabled entries that the user can choose as recipients of an e-mail message.	Yes
Offline Address Book (OAB)	A partial copy of Address Book information from a directory service stored locally on the client machine. OAB data is generated by the server and retrieved by the client via HTTP or public folders. See [MS-OXWOAB] and [MS-OXPFOAB]. The format of the OAB file is described in [MS-OXOAB].	Depends on client configuration
.MSG File Format	The .MSG file format is a container for a static representation of the properties and streams that comprise the Message and Attachment Object Protocol described in [MS-OXCMSG].	Depends on client configuration

2.3 Protocol Relationships

Each of the protocols within the Office Exchange Protocols documentation set belongs to one of three protocol stacks–Remote Procedure Call (RPC)-based protocols, HTTP-based protocols, and standards-based protocols. Each of these will be discussed in this section.

The following diagram serves to illustrate the relationships between the protocols.

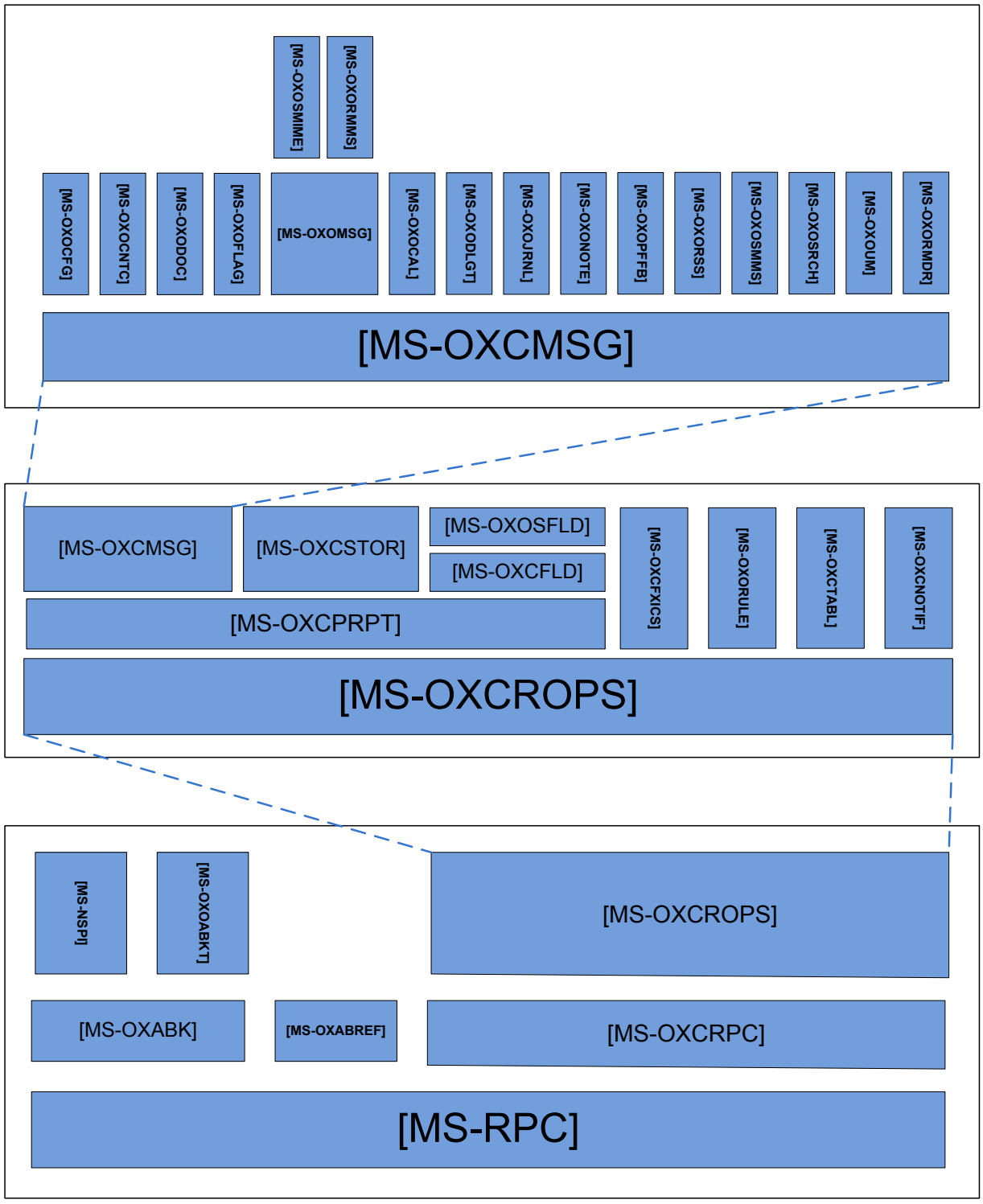

Figure 5: Office/Exchange protocol stack

2.3.1 Remote Procedure Call (RPC) based Protocols

The task of sending e-mail, publishing/retrieving information and coordinating across networks can be achieved multiple ways. Some of these are stateless, others procedural or object oriented. Each has their tradeoffs (for example, security, bandwidth, and extensibility). The majority of the Office Exchange protocols are procedural, and part of the RPC-based protocol stack.

Like other procedural systems, there exist groupings of functionality that can be thought of as technology areas. In this suite there are two main areas: those that deal with personal/shared mailbox storage (which includes calendar, tasks, personal contacts and the organization of the aforementioned) and Address Book (directory) functionality.

Protocol Name	Description	Document Short Name
Wire Format Protocol	The Wire Format Protocol is a stateful protocol which specifies how a client can make calls to one or more servers containing messaging data. This protocol is a foundation protocol upon which other protocols listed in the Property and Stream Object Protocol section are dependent.	[MS-OXCRPC]
Name Service Provider Interface (NSPI) Protocol	The NSPI Protocol which specifies how a client can make requests to a directory service in order to access and manipulate remote **Address Book objects** in a directory store.	[MS-NSPI]
Address Book Name Service Provider Interface (NSPI) Referral Protocol	The NSPI Referral Protocol specifies how a client initially connects to a server and can be used to find a new NSPI server instance.	[MS-OXABREF]

2.3.1.1 HTTP Tunneling

RPC-based protocols can be tunneled over HTTP when a client communicates over the Internet by using the “Remote Procedure Call Over HTTP Protocol” as specified in [MS-RPCH]. When operating in this mode, it is common that the RPC-based protocol endpoint for directory services

is not directly accessible. When HTTP tunneling is detected, the Address Book Name Service Provider Interface (NSPI) Referral protocol will correspondingly refer clients to a collocated NSPI server endpoint to ensure client connectivity.

2.3.1.2 Core and Object Protocols

Core and Object protocols exist at the application-layer and are built upon the protocols specified in [MS-OXCRPC], [MS-NSPI] and [MS-OXABREF].

2.3.1.2.1 Core Protocols

Protocols outlined in this section build on top of the Remote Operations (ROP) List and Encoding Protocol specified in [MS-OXCROPS]. [MS-OXCROPS] itself defines the set of operations and responses that are transmitted between client and server via the [MS-OXCRPC] protocol, including ordering and encoding of operation-level object data structures.

Protocol Name	Description	Document Short Name
Message and Attachment Object Protocol	The Message and Attachment Object Protocol 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 Protocol	The Folder Object Protocol specifies the properties and permissible operations for **Folder objects**.	[MS-OXCFOLD]
Data Structures Protocol	The Data Structures Protocol specifies structures which are used by multiple protocol areas.	[MS-OXCDATA]
Bulk Data Transfer Protocol	The Bulk Data Transfer Protocol specifies the ordering and data flow for transferring data between client and server implementations of the Office Exchange Protocol.	[MS-OXCFXICS]
Property and Stream Object Protocol	The Property and Stream Object Protocol specifies the properties and permissible operations for the core **property** and stream objects.	[MS-OXCPRPT]

Store Object Protocol	The Store Object Protocol specifies the properties and permissible operations for the core **message** store objects.	[MS-OXCSTOR]
Table Object Protocol	The Table Object Protocol specifies the properties and permissible operations for the core table objects.	[MS-OXCTABL]
Core Notifications Protocol Specification	The Core Notifications Protocol specifies the subscription and delivery mechanisms for push and polled notifications on changes.	[MS-OXCNOTIF]

2.3.1.2.2 Object Protocols

The Object protocols are the high-level objects, such as Appointments and Contacts, that are built upon or extend the Core objects listed in the previous section.

Protocol Name	Description	Document Short Name
Address Book Object	The Address Book Object Protocol specifies the properties and operations on lists of users, contacts, groups and resources using the Name Service Provider Interface Protocol described in [MS-NSPI].	[MS-OXOABK]
Address Book User Interface Templates	The Address Book User Interface Templates Protocol specifies the properties and operations permissible for address book templates, including location using the Address Book Protocol described in [MS-NSPI].	[MS-OXOABKT]

Appointment and Meeting Object	The Appointment and Meeting Object Protocol 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 specified in [MS-OXCMSG] and the E-mail Object Protocol specified in [MS-OXOMSG]	[MS-OXOCAL]
Configuration Information	The Configuration Information Protocol 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 specified in [MS-OXCMSG].	[MS-OXOCFG]
Contact Object	The Contact Object Protocol specifies the properties and operations permissible on contacts and personal distribution lists using the Message and Attachment Object Protocol specified in [MS-OXCMSG].	[MS-OXOCNTC]
Delegate Access Configuration	The Delegate Access Configuration Protocol specifies connections to and configuration of mailboxes when acting on behalf of another user or resource. It also specifies interactions with messages as described in [MS-OXOMSG], [MS-OXOTASK], and [MS-OXOCAL] when acting on behalf of another user.	[MS-OXODLGT]
Document Object	The Document Object Protocol specifies the properties and operations permissible on documents stored using the Message and Attachment Object Protocol as described in [MS-OXCMSG].	[MS-OXODOC]

Informational Flagging	The Informational Flagging Protocol specifies the properties and operations related to flagging messages for follow-up using the Message and Attachment Object Protocol described in [MS-OXCMSG].	[MS-OXOFLAG]
Journal Object	The Journal Object Protocol specifies the properties and operations permissible on journal objects using the Message and Attachment Object Protocol described in [MS-OXCMSG].	[MS-OXOJRNL]
E-mail Object	The E-mail Object Protocol specifies the properties and operations permissible, such as reply and forward, on e-mail messages, and includes dealing with recipients and attachments using the ROP List and Encoding Protocol described in [MS-OXCROPS].	[MS-OXOMSG]
Note Object	The Note Object Protocol specifies the properties and operations permissible on note objects using the Message and Attachment Object Protocol described in [MS-OXCMSG].	[MS-OXONOTE]
Public Folder Availability	The Public Folder Based Free/Busy Protocol specifies the format of Free/Busy data which describes the availability of a user or resource, which is persisted in **public folders**. This data can be used to efficiently schedule meetings and/or provide presence information.	[MS-OXOPFFB]
Post Object	The Post Object Protocol specifies the properties and operations permissible on posts using the Message and Attachment Object Protocol as specified in [MS-OXCMSG].	[MS-OXOPOST]

Reminder Settings	The Reminder Settings Protocol specifies the properties and interaction model for reminders set on messages as specified in [MS-OXCMSG].	[MS-OXORMDR]
Rights-Managed E-mail Object	The Rights-Managed E-mail Object Protocol 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): Client-Server Protocol as specified in [MS-RMPR].	[MS-OXORMMS]
RSS Object	The RSS Object Protocol 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	The E-mail Rules Protocol specifies the mechanism and properties for manipulating server-side rules via the ROP List and Encoding Protocol as specified in [MS-OXROPS] in order to act on incoming and outgoing e-mail messages.	[MS-OXORULE]
Special Folders	The Special Folders Protocol specifies the properties and operations used to create and locate the **special folders** in a mailbox using the ROP List and Encoding Protocol as specified in [MS-OXCROPS].	[MS-OXOSFLD]
S/MIME E-mail Object	The S/MIME E-mail Object Protocol specifies the properties of Secure MIME signed and encrypted messages using the E-mail Object Protocol as specified in [MS-OXOMSG].	[MS-OXOSMIME]

SMS and MMS Object	The SMS and MMS Object Protocol specifies the properties and operations necessary 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	The Search Folder List Configuration protocol specifies the properties and operations necessary to manipulate the **search folder** list configuration using the ROP List and Encoding Protocol as specified in [MS-OXCROPS].	[MS-OXOSRCH]
Task-Related Objects	The Task-Related Objects Protocol specifies the properties of and operations on contacts and personal distribution lists using the Message and Attachment Object Protocol as specified in [MS-OXCMSG].	[MS-OXOTASK]
Voice Mail and Fax Objects	The Voice Mail and Fax Objects Protocol specifies the properties and operations necessary to represent voice mail and fax messages using the Message and Attachment Object Protocol as specified in [MS-OXCMSG].	[MS-OXOUM]

2.3.2 Client RPC Operational Modes

There are two main ways to implement a client's use of RPC to operate with the server. Depending on the implementation, the client will use different components of the Office Exchange Protocol to communicate with the server.

Mode	Description
Online Mode	In this implementation, a client typically retrieves information directly from a server. User requests (sending an e-mail, scheduling an appointment, etc) are sent directly to the server to be performed immediately. When state changes occur on the server (for example, delivery of a new mail **message** or creation of

	a new public folder item by another user), the server is expected to notify the client of the change so the user's view can be updated.
Offline Mode	To make it possible for the user to keep working when the client is unable to connect to the server, the user can implement a client to run in an offline or primarily-offline mode. In such a mode, folders in the user's personal store could be replicated to a local offline cache and directory information could be replicated to a local offline address book (OAB). This could permit the client to retrieve information from the offline cache or offline address book. In this case, many user actions would result in operations against the client's information caches rather than making a request of the server. To keep the offline caches of information up to date, the client would be expected to periodically synchronize, using methods described in [MS-OXCSYNC], [MS-OXPFOAB] and [MS-OXWOAB].

Note that the Office Exchange Protocols allow for the client to offer gradation of functionality between these two modes. For instance the client can allow the user to work primarily in online mode, but maintain a cache of certain **folders** that can be used when the client is not connected to the server.

2.3.3 Client/Server Protocol Functional Areas

The client's operation against the server is divided into 10 functional areas. The client implementations described in the previous section use different subsets of these 10 areas. The table below outlines the areas used by each implementation.

Functional Area	Description	Document Short Name
AutoDiscover	The first time a client connects to a Server, the client needs to perform auto discovery to determine which server endpoints host the instances of the directory service and server protocols suitable for this session. This protocol is also called by the client after connectivity failures and when looking up mailbox information of other users.	[MS-OXDISCO]

Address Book Browsing	Used by a client to display a browseable list of address book entries stored in a directory service, to search a directory service, and to perform ambiguous name resolution.	[MS-OXOABK]
Address Book Details	When the user selects a particular address book entry, this is used to display detailed information about that entry.	[MS-OXOABKT]
Server Store Navigation and Modification	Used whenever a client works directly with objects in the server store (as opposed to from an offline cache).	[MS-OXCRPC],
Push Notification	Unlike where a client sends a request and a server or directory service responds synchronously, this is an area where a client receives notifications asynchronously. With Push Notification, a client simply registers for notifications for changes to the **folder** hierarchy and the contents of a list of specified folders. If the client does not make any requests of the server within a specified time, the server will use this protocol to notify the client of any recent server-side updates.	[MS-OXCNOTIF]
Offline Cache Synchronization	Used to synchronize the folder hierarchy and contents of specified folders from the server to the client's offline cache.	[MS-OXCFXICS], [MS-OXCSYNC]
Offline Address Book Synchronization	Used by a client to download a complete copy of, or incremental updates to, the offline address book.	[MS-OXPFOAB], [MS-OXWOAB]
Schedule Availability	When a user wishes to schedule a meeting with one or more users or resources, this protocol is used to determine free and busy times within a specified time interval.	[MS-OXWAVLS]

Out of Office Control	Used to mark the user “in” or “out” of the office, and to configure the behavior that occurs when the user receives an e-mail **message** and is “out of office.”	[MS-OXWOOF]

2.3.4 Web Services and HTTP-based Protocols

HTTP/1.1 and Web Services provide a standards-based layer upon which to build specific client-server protocols. The protocols listed below are built on client and server implementations of HTTP/1.1 [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
AutoDiscover HTTP Service	The AutoDiscover HTTP Service protocol specifies a request and response schema which allows clients to obtain configuration information related to the RPC- and Web Service-based endpoints exposed by the server. The method for server deployment and client discovery of the AutoDiscover HTTP service is specified in [MS-OXDSCLI].	[MS-OXDISCO]
Offline Address Book (OAB) Retrieval	The Offline Address Book (OAB) Retrieval protocol specifies how clients download new full and incremental versions of the Offline Address Book (OAB). When not available, the OAB Public Folder Retrieval protocol, as specified in [MS-OXPFOAB], is used.	[MS-OXWOAB]
Availability Web Service	The Availability Web Service protocol specifies methods that a client can use to retrieve calendar specific data, such as whether a user is free or busy.	[MS-OXWAVLS]

Out of Office (OOF) Web Service	The Out of Office (OOF) Web Service protocol specifies how to configure the automated response behavior of the e-mail system when a user is unavailable.	[MS-OXWOOF]
Voice Mail Settings Web Service	The Voice Mail Settings Web Service Protocol 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]

2.3.4.1 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, and is specified in [RFC4918].

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

Protocol Name	Description	Document Short Name
WebDAV Extensions for Calendar Support	The WebDAV Extensions for Calendar Support protocol specifies property extensions to [RFC4918], [MS-WDVME], and [MS-WDVSE] to allow for creation and manipulation of Calendar objects by using WebDAV. This protocol specifies properties that will allow clients to find the address for a user's default calendar, get and set events on a calendar, find the address to a user's default free/busy time, and get access to the user's free/busy time.	[MS-XWDCAL]
WebDAV Extensions for Contacts Support	The WebDAV Extensions for Contacts Support protocol extends the WebDAV protocol to allow the creation and manipulation of Contact objects by using WebDAV.	[MS-XWDCNTC]

WebDAV Extensions for Documents Support	The WebDAV Extensions for Documents Support protocol specifies property extensions to [RFC4918], [MS-WDVME], and [MS-WDVSE] to allow for creation and manipulation of Document objects by using WebDAV.	[MS-XWDDOC]
WebDAV Extensions for Folder Support	The WebDAV Extensions for Folder Support protocol 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	The WebDAV Extensions for E-Mail Support protocol extends the WebDAV protocol. For more details about the WebDAV protocol, see [RFC4918]. The WebDAV Extensions for E-Mail Support protocol is used by clients to send, receive, and manipulate e-mail through HTTP.	[MS-XWDMAIL]
WebDAV Extensions for Notifications	The WebDAV Extensions for Notifications protocol extends the WebDAV protocol, as specified in [RFC4918], to provide event notification for content that is contained on a WebDAV server.	[MS-XWDNOTIF]
WebDAV Extensions for Replication	The WebDAV Extensions for Replication protocol specifies the client-server replication of Web resources on a WebDAV server by means of the extension of the HTTP and WebDAV protocols.	[MS-XWDREPL]
WebDAV Extensions for Search	The WebDAV Extensions for Search protocol specifies extensions to the WebDAV protocol [RFC4918] to allow clients to request searches of content. This protocol also allows clients to create persistent search folders on the server.	[MS-XWDSEARCH]

WebDAV Extensions for Structured Documents	The WebDAV Extensions for Structured Documents protocol specifies extensions to the WebDAV protocol [RFC4918] to allow for creation and manipulation of structured documents. This protocol allows clients to retrieve, insert, change, and remove individual pieces of structured documents on the server.	[MS-XWDSTRUCTDOC]
XML-Data for Exchange DAV	The XML-Data for Exchange DAV protocol specifies extensions to the WebDAV protocol. It concerns a set of extensions to the property system that enables the specification of specific data types on properties that are used by the PROPFIND, PROPPATCH, BPROPFIND, BPROPPATCH, and SEARCH methods.	[MS-XWDTYPESYS]
WebDAV Extensions for Security	The WebDAV Extensions for Security specifies an extension to the WebDAV protocol. This extension specifies how to request and set the Exchange security descriptor by using the WebDAV methods PROPFIND and PROPPATCH.	[MS-XWDVSEC]

2.3.4.2 Exchange ActiveSync-Related Protocols

The Exchange ActiveSync-related protocols specify syntax and behavior that allow for data synchronization between a server and a (typically mobile) client. The Exchange ActiveSync-related protocols also provide for a mechanism that helps the client stay up-to-date when changes occur on the server; for example, when a new e-mail message arrives.

Protocol Name	Description	Document Short Name

ActiveSync AirSyncBase Namespace	The ActiveSync AirSyncBase Namespace protocol specifies the elements and complex types used by multiple Exchange ActiveSync commands to identify the size, type, and content of data returned to the client in the response message.	[MS-ASAIRS]
ActiveSync Calendar Class	The ActiveSync Calendar Class protocol specifies the ActiveSync protocol format for the interchange of calendar data.	[MS-ASCAL]
ActiveSync Command Reference	The ActiveSync Command reference specifies and enumerates the XML-based commands used by a client to synchronize e-mail, files, and data with a server.	[MS-ASCMD]
ActiveSync Contact Class	The ActiveSync Contact Class protocol specifies the ActiveSync protocol format for the interchange of contact data.	[MS-ASCNTC]
ActiveSync Document Class	The ActiveSync Document Class protocol specifies how documents stored in Windows SharePoint Services and on file shares using UNC paths is communicated from the server to the client.	[MS-ASDOC]
Exchange ActiveSync Data Types	The Exchange ActiveSync Data Types protocol specifies the required format of each data type used by the Exchange **ActiveSync XML schema definitions (XSDs)** to be sent in **Wide Area Protocol (WAP) Binary XML (WBXML)** format.	[MS-ASDTYPE]
ActiveSync E-Mail Class	The ActiveSync E-Mail Class protocol specifies the XML representation of e-mail data sent or received on (typically) mobile devices that communicate by using the Exchange ActiveSync protocols.	[MS-ASEMAIL]
ActiveSync HTTP	ActiveSync HTTP protocol specifies the format of communication between a client and a server via an **HTTP POST** using UTF-8 encoding.	[MS-ASHTTP]

ActiveSync Provisioning	The ActiveSync Provisioning protocol specifies an XML-based format that Exchange servers use to communicate security policy settings to client devices.	[MS-ASPROV]
ActiveSync Tasks Class	The ActiveSync Tasks Class protocol specifies the format for the interchange of task data between a client and a server.	[MS-ASTASK]
ActiveSync WAP Binary XML (WBXML)	The ActiveSync WAP Binary XML (WBXML) protocol specifies how **WBXML** functionality is utilized by the ActiveSync protocols. This document also specifies the **tokens** and **code pages** used to perform the WBXML encoding.	[MS-ASWBXML]

2.3.5 Standards-Based Protocols

The Office Exchange Protocols extend a number of standards-based protocols and data formats.

2.3.5.1 Standards-based client and server Protocols

A server implementing the Office Exchange Protocols is expected to support a number of different standard protocols for e-mail (POP3, SMTP, IMAP4, and WebDAV) and directory information (LDAP). The Office Exchange Protocol documentation set specifies a number of extensions to these standards, primarily for authentication and authorization, as described in the following documents.

Protocol Name	Description	Document Short Name
Lightweight Directory Access Protocol (LDAP) Profile	The Lightweight Directory Access Protocol (LDAP) Profile Protocol is a binary TCP protocol that enables directory access. This document specifies client implementation of LDAPv3.	[MS-OXLDAP]
IMAP4 extensions for NTLM authentication	This protocol includes an extension to the IMAP4rev1 protocol, as specified in IETF	[MS-OXIMAP4]

and alternate mailbox login	RFC3501, to support NTLM authentication and access to mailboxes other than the one associated with the authenticated user.	
POP3 extensions for NTLM authentication	This protocol includes an extension to the POP3 mailbox protocol, as specified in IETF RFC1939, to support NTLM authentication.	[MS-OXPOP3]
SMTP Message Submission	This protocol specifies an extension to the SMTP Message Submission protocol, as specified in IETF RFC4409, to support NTLM authentication.	[MS-OXSMTP]
SMTP Protocol: AUTH LOGIN Extension	The SMTP Protocol: AUTH LOGIN Extension protocol specifies the extension to the Simple Mail Transfer Protocol to support a simple, Base 64-encoded authentication mechanism.	[MS-XLOGIN]
Web Distributed Authoring and Versioning (WebDAV) Protocol: Security Descriptor Extensions	The Web Distributed Authoring and Versioning (WebDAV) Protocol: Security Descriptor Extensions specifies extensions to the WebDAV [IETF RFC2518] protocol used to configure security descriptors for access control to items stored on the server.	[MS-XWDVSEC]

2.3.5.2 Transforming Standards data

When clients and servers connect via the internet or via transports external to the managed environment such as a domain, and send or receive data in standards-based formats, they are expected to use the following Office Exchange Protocols to convert between standard-based formats and one or more of the formats that are part of the Office Exchange Protocols.

Protocol Name	Description	Document Short Name

RFC2822 and MIME to E-mail Object Conversion	The RFC2822 and MIME to E-mail Object Conversion Protocol specifies conversion from Internet standards e-mail conventions to **Message objects**, as specified in [MS-OXOMSG].	[MS-OXCMAIL]
iCalendar to Appointment Object Conversion	The iCalendar to Appointment Object Conversion Protocol specifies how to convert between IETF RFC2445, 2446 and 2447 and Appointment and Meeting objects as specified in [MS-OXOCAL], in conjunction with [MS-OXCMAIL].	[MS-OXCICAL]
Enriched Text Format (ETF) Message Body Conversion	The Enriched Text Format (ETF) Message Body Conversion Protocol specifies server behavior when dealing with a text/enriched **message** body.	[MS-OXCETF]
Journal Record Message Format	The Journal Record Message Format Protocol specifies a set of extensions based on IETF RFC2822, RFC2045 and RFC2046 to represent metadata required to journal or archive e-mail messages, including a method of capturing the original e-mail message sender and recipients, and a copy of the original message.	[MS-XJRNL]

3 Appendix

The following products use the protocols referenced in this overview:

Microsoft Office Outlook 2007 SP1

Microsoft Office Outlook 2003 SP2

Microsoft Exchange Server 2007 SP1

Microsoft Exchange Server 2003 SP2

4 Copyright statement

Copyright statements will be included per Microsoft publishing guidelines.

Index

<1> Protocol implementation differences between Exchange Server 2007 and Exchange Server 2003 are specified in the relevant protocol specifications.