

[MS-ASCMD]: ActiveSync Command Reference Protocol Specification

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	December 3, 2008	1.0	Initial Release.
Microsoft Corporation	January 15, 2009	1.01	Revised and edited technical content.

Table of Contents

1	Introduction.....	6
1.1	Glossary	6
1.2	References	7
1.2.1	Normative References	7
1.2.2	Informative References	9
1.3	Protocol Overview	9
1.4	Relationship to Other Protocols.....	9
1.5	Prerequisites/Preconditions.....	10
1.6	Applicability Statement.....	10
1.7	Versioning and Capability Negotiation.....	10
1.8	Vendor-Extensible Fields	10
1.9	Standards Assignments	10
2	Messages.....	10
2.1	Transport.....	10
2.2	Message Syntax.....	10
2.2.1	Commands	10
2.2.1.1	Autodiscover	10
2.2.1.1.1	Request	11
2.2.1.1.2	Response.....	12
2.2.1.2	FolderCreate.....	17
2.2.1.2.1	Request.....	17
2.2.1.2.2	Response.....	20
2.2.1.3	FolderDelete.....	23
2.2.1.3.1	Request	23
2.2.1.3.2	Response.....	25
2.2.1.4	FolderSync	27
2.2.1.4.1	Request	27
2.2.1.4.2	Response.....	29
2.2.1.5	FolderUpdate.....	36
2.2.1.5.1	Request	36
2.2.1.5.2	Response.....	39
2.2.1.6	GetAttachment	41
2.2.1.6.1	Request	41
2.2.1.6.2	Response.....	41
2.2.1.7	GetItemEstimate	41
2.2.1.7.1	Request	42
2.2.1.7.2	Response.....	45
2.2.1.8	ItemOperations.....	49
2.2.1.8.1	Delivery of Content Requested by Fetch	49
2.2.1.8.2	Request	51
2.2.1.8.3	Response.....	62

2.2.1.9	MeetingResponse	71
2.2.1.9.1	Request	71
2.2.1.9.2	Response	74
2.2.1.10	MoveItems	77
2.2.1.10.1	Request	77
2.2.1.10.2	Response	79
2.2.1.11	Ping	81
2.2.1.11.1	Request	82
2.2.1.11.2	Response	85
2.2.1.12	Provision	88
2.2.1.13	ResolveRecipients	89
2.2.1.13.1	Request	89
2.2.1.13.2	Response	92
2.2.1.14	Search	98
2.2.1.14.1	Request	100
2.2.1.14.2	Response	118
2.2.1.15	SendMail	124
2.2.1.15.1	Request	124
2.2.1.15.2	Response	124
2.2.1.16	Settings	124
2.2.1.16.1	Request	125
2.2.1.16.2	Response	143
2.2.1.17	SmartForward	156
2.2.1.17.1	Request	156
2.2.1.17.2	Response	157
2.2.1.18	SmartReply	157
2.2.1.18.1	Request	157
2.2.1.18.2	Response	158
2.2.1.19	Sync	158
2.2.1.19.1	Request	160
2.2.1.19.2	Response	194
2.2.1.19.3	Content Class Specific XSDs	224
2.2.1.20	ValidateCert	245
2.2.1.20.1	Request	245
2.2.1.20.2	Response	248
2.2.2	Status Codes	250
2.2.2.1	FolderCreate Status Codes	250
2.2.2.2	FolderDelete Status Codes	252
2.2.2.3	FolderSync Status Codes	253
2.2.2.4	FolderUpdate Status Codes	255
2.2.2.5	GetItemEstimate Status Codes	256
2.2.2.6	MeetingResponse Status Codes	257
2.2.2.7	MoveItems Status Codes	258
2.2.2.8	Ping Status Codes	260

2.2.2.9	Provision Status Codes	262
2.2.2.10	ResolveRecipients Status Codes	263
2.2.2.11	Search Status Codes	265
2.2.2.12	Sync Status Codes	267
2.2.2.13	ValidateCert Status Codes	270
3	<i>Protocol Details</i>	273
3.1	Common Details.....	273
3.1.1	Abstract Data Model	273
3.1.2	Timers	273
3.1.3	Initialization	273
3.1.3.1	Initial Synchronization.....	273
3.1.4	Higher-Layer Triggered Events	273
3.1.5	Message Processing Events and Sequencing Rules.....	273
3.1.5.1	Downloading Policy Settings.....	274
3.1.5.2	Synchronizing Inbox, Calendar, Contacts, and Tasks Folders	275
3.1.5.3	Synchronizing a Folder Hierarchy	278
3.1.5.4	Receiving and Accepting Meeting Requests	280
3.1.5.5	Handling Status Errors.....	282
3.1.6	Timer Events.....	283
3.1.7	Other Local Events	283
4	<i>Protocol Examples</i>	283
4.1	Downloading the Current Server Security Policy by Using the Provision Command	283
4.2	Discovering Account Settings by Using the AutoDiscover Command	283
4.2.1	Request XML Body Outline	284
4.2.2	Response XML Body Outline - Case Error	285
4.2.3	Response XML Body Outline - Case Redirect.....	285
4.2.4	Response XML Body Outline - Case Server Settings.....	286
4.2.5	Response XML Body Outline - Case Framework Error	287
4.2.6	Response XML – Case Framework Default	288
4.3	Synchronizing Data by Using the Sync Command	289
4.3.1	Synchronizing Folders.....	289
4.3.2	Fetching an E-Mail by Using the ServerID.....	290
4.3.3	Uploading New ApplicationData to the Server	290
4.3.4	Updating ApplicationData on the Server	291
4.3.5	Downloading Current Information from the Server	292
4.3.6	Identifying Acceptance of Partial Collections.....	293
4.3.7	Identifying Acceptance of MIME Content	293
4.3.7.1	Sync Request With Support for MIME Content.....	293
4.3.7.2	Sync Response With MIME Content.....	294
4.3.7.3	Sync Request With BodyPreferences and MIME Support	295
4.3.7.4	Sync Response with MIME Support.....	295
4.3.8	Identifying That More Content is Ready for Download.....	297

4.3.9	Synchronizing the Calendar Folder	297
4.4	Pinging the Server for Updates by Using the Ping Command.....	298
4.4.1	Ping Command Request.....	298
4.4.2	Ping Command Response	298
4.4.2.1	Typical Response	298
4.4.2.2	Changes Found.....	299
4.4.2.3	HeartbeatInterval Error	299
4.4.2.4	Folder Error.....	299
4.5	Fetching E-Mail and Attachments by Using the ItemOperations Command	300
4.5.1	Fetching an E-Mail Item	300
4.5.2	Fetching a MIME E-Mail Item	302
4.5.2.1	Request.....	302
4.5.2.2	Response	303
4.5.3	Fetching an E-Mail Item with a LongId	305
4.5.4	Fetching an Attachment	310
4.5.4.1	Sync Command Request	310
4.5.4.2	Sync Command Response.....	311
4.5.4.3	ItemOperation Command Request.....	313
4.5.4.4	ItemOperation Command Response	313
4.6	Retrieving and Changing OOF Settings by Using the Settings Command.....	315
4.6.1	Retrieving OOF Settings	315
4.6.2	Turning On the OOF Message.....	316
4.6.3	Turning Off the OOF Message	318
4.7	Retrieveing User Information by Using the Settings Command	319
4.8	Setting Device Information by Using the Settings Command.....	320
4.9	Setting a Device Password by Using the Settings Command.....	321
4.10	Accessing Documents on File Shares and URIs by Using the Search and ItemOperations Commands	321
4.10.1	Issuing a Search for Item Metadata	322
4.10.2	Fetching an Item Based on Metadata	325
4.11	Searching for an Item in the Mailbox by Using the Search Command	327
4.11.1	Keyword Search	327
4.11.2	Forward a Search Result	330
4.12	Resolving Recipients by Using the ResolveRecipient Command	331
4.12.1	RecipientCount Response for a GAL Entry	331
4.12.2	Response for a Contact Entry	332
5	Security.....	332
5.1	Security Considerations for Implementers.....	332
5.2	Index of Security Parameters.....	332
6	Appendix A: Office/Exchange Behavior.....	332
	Index.....	336

1 Introduction

This document specifies the ActiveSync protocol commands which are used by a client, typically a mobile device, to synchronize and exchange objects with a server. These objects include e-mail messages, folders, and attachments, contact information, meetings, calendar data, and documents.

1.1 Glossary

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

- address list**
- address book**
- ambiguous name resolution (ANR)**
- appointment**
- ASCII**
- attachment**
- binary large object (BLOB)**
- character set**
- class**
- collection**
- contact**
- distribution list**
- domain**
- Domain Name System (DNS)**
- GUID**
- header field**
- Hypertext Transfer Protocol (HTTP)**
- Inbox folder**
- journal**
- locale**
- mailbox**
- meeting**
- MIME**
- named property**
- organizer**
- Out of Office (OOF)**
- Outbox folder**
- Personal Information Manager (PIM)**
- plain text**
- read receipt**
- recipient**
- reminder**
- search folder**

S/MIME
Secure Socket Layers (SSL)
Sent Mail folder
Simple Mail Transfer Protocol (SMTP)
special folder
Uniform Resource Identifier (URI)
WAP binary XML (WBXML)
Wireless Access Protocol (WAP)
XML

The following terms are specific to this document:

Global Address List (GAL): The **address list** that conceptually represents the default address list for an **address book**.

Universal Naming Convention (UNC): A standard naming format for specifying the location of network resources such as shared files or devices on a network. The format is "\\<servername>\<share>\<filename>", where <servername> is a NetBIOS name, FQDN **domain** name, or IPv4 address; <share> is a logical share point for accessing <servername>; and <filename> is the name of the file or device.

XML schema definition (XSD): A language proposed by the W3C **XML** Schema Working Group for use in defining schemas. Schemas are useful for enforcing structure and/or constraining the types of data that can be used validly within other **XML** documents. **XML** schema definition refers to the fully specified and currently recommended standard for use in authoring **XML** schemas.

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

[DNS-SRV] Microsoft Corporation, "A new feature is available that enables Outlook 2007 to use DNS Service Location (SRV) records to locate the Exchange Autodiscover service", August 2007, <http://go.microsoft.com/fwlink/?linkid=3052&kbid=940881>.

[MS-ASAIRS] Microsoft Corporation, "ActiveSync AirSyncBase Namespace Protocol Specification", December 2008.

[MS-ASCAL] Microsoft Corporation, "ActiveSync Calendar Class Protocol Specification", December 2008.

[MS-ASCNTC] Microsoft Corporation, "ActiveSync Contact Class Protocol Specification", December 2008.

[MS-ASDOC] Microsoft Corporation, "ActiveSync Document Class Protocol Specification", December 2008.

[MS-ASDTYPE] Microsoft Corporation, "ActiveSync Data Type Protocol Specification", December 2008.

[MS-ASEMAIL] Microsoft Corporation, "ActiveSync E-Mail Class Protocol Specification", December 2008.

[MS-ASHTTP] Microsoft Corporation, "ActiveSync HTTP Protocol Specification", December 2008.

[MS-ASPROV] Microsoft Corporation, "ActiveSync Provisioning Protocol Specification", December 2008.

[MS-ASTASK] Microsoft Corporation, "ActiveSync Tasks Class Protocol Specification", December 2008.

[MS-ASWBXML] Microsoft Corporation, "ActiveSync WAP Binary XML(WBXML) Protocol Specification", December 2008.

[MS-OXDISCO] Microsoft Corporation, "Autodiscover HTTP Service Protocol Specification", June 2008.

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

[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>.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", RFC 821, August 1982, <http://www.ietf.org/rfc/rfc821.txt>.

[RFC822] Crocker, D.H., "Standard for ARPA Internet Text Messages", RFC 822, August 1982, <http://www.ietf.org/rfc/rfc0822.txt>.

[W3C-XML] World Wide Web Consortium, "XML Schema (Second Edition)", October 2004, <http://www.w3.org/XML/Schema>.

1.2.2 Informative References

[AUTODISCOVER] Microsoft Corporation, "White Paper: Exchange 2007 Autodiscover Service", November 2007, <http://technet.microsoft.com/en-us/library/bb332063.aspx>.

1.3 Protocol Overview

This protocol consists of a set of **XML**-based commands that are used by a client device to synchronize and exchange its e-mail, files, and data with a server.

The client first uses the **Provision** command to get and then acknowledge security policy settings from the server.

The client can then call the **Autodiscover** command to get a user's account configuration. The client can then view and modify server data related to that account, including e-mail messages and **attachments**, folders, **contacts**, and calendar requests.

Typically, the next command sent by the client is **Sync**, to receive a synchronization key from the server, and synchronize the messages within in a folder, and optionally **Ping** to set the synchronization refresh interval.

The client processes outgoing e-mail using the **SendMail**, **SmartReply**, and **SmartForward** commands. For incoming messages, the client can call the **ItemOperations** command to fetch the message, then use the **GetAttachment**, **GetItemEstimate**, and **MoveItems** commands. **S/MIME** messages are processed with the **ResolveRecipients** and **ValidateCert** commands.

The client calls the **FolderSync**, **FolderCreate**, **FolderUpdate**, and **FolderDelete** commands to update, create, and delete **mailbox** folders on the server.

For **meeting** requests, the client calls the **MeetingResponse** command.

The client can set and request server parameters with the **Settings** command.

The client uses the **Search** command to find particular items on the server.

1.4 Relationship to Other Protocols

The commands in this specification are used over a **Hypertext Transfer Protocol (HTTP)** connection, as specified in [RFC2616].

The header information for command requests and responses is specified in [MS-ASHTTP].

There are several related **class** protocols that use the commands defined in this specification to synchronize data. For details about the class protocols, see [MS-ASEMAIL], [MS-ASCNTC], [MS-ASDOC], [MS-ASCAL], and [MS-ASTASK].

All simple data types in this document conform to the data type definitions specified in [MS-ASDTYPE].

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is applicable in scenarios where a client has to synchronize its messages and files with a server.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol consists of a series of **XML** elements contained in request or response messages between a client and server. The XML block containing the command and parameter elements is transmitted in either the request body of a request, or in the response body of a response. The request body and request response are always preceded by the **HTTP** header, as specified in [MS-ASHTTP].

All command messages use **WAP binary XML (WBXML)**, except for the **Autodiscover** command, which uses plain XML. For more details about **WAP** binary XML, see [MS-ASWBXML].

2.2 Message Syntax

2.2.1 Commands

2.2.1.1 Autodiscover

The **Autodiscover** command facilitates the discovery of core account configuration information by using the user's **Simple Mail Transfer Protocol (SMTP)** address as the primary input. <1> For more details about the Autodiscover **HTTP** Service, see [MS-OXDISCO].

The **Autodiscover** command request and response messages are sent in **XML** format, not **WBXML** format.

The client SHOULD use the **Autodiscover** command as an initial response to common HTTP errors. Common HTTP errors are specified in [MS-ASHTTP] section 2.2.2.1.1. **Autodiscover** has the ability to retrieve an updated URL when a **mailbox** has been moved, or a user is trying to connect to the wrong server.

After a successful **Autodiscover** command response, the client SHOULD send an **Options** command to the new server. The **Options** command returns the newly supported protocol versions and commands if they changed due to the **Autodiscover** command.

2.2.1.1.1 Request

2.2.1.1.1.1 Request

The **Request** element contains the **Autodiscover** command request parameters.

Parent elements	Child elements	Data type	Number allowed
<Autodiscover>	<EmailAddress> <AcceptableResponseSchema>	Container	1...1 (required)

2.2.1.1.1.2 AcceptableResponseSchema

The **AcceptableResponseSchema** element indicates the schema in which the server MUST send the response.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only) <Response> (response only)	None	String	1...1 (required)

The schema MUST be
"http://schemas.microsoft.com/exchange/autodiscover/mobilesync/responseschema/2006".

2.2.1.1.1.3 EmailAddress

The **EmailAddress** element contains the **SMTP** e-mail address of the user and is used to identify the user's **mailbox** in the network.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only) <User> (response only)	None	String	1...1 (required)

If the user has multiple addresses, then the primary e-mail address **SHOULD** be returned in the **Autodiscover** command response. This address can be the same as the e-mail address that was sent in the request. The client device **SHOULD** record this address and **SHOULD** use this string for all additional communication.

2.2.1.1.2 *Response*

2.2.1.1.2.1 **Action**

The **Action** element encapsulates the server action type for this request, which can be one of the following: **Redirect**, **Settings**, or **Error**.

Parent elements	Child elements	Data type	Number allowed
None	<Redirect> (response only) <Settings> (response only) <Error> (response only)	Container	1...1 (required)

2.2.1.1.2.2 **Culture**

The **Culture** element specifies the client culture, which is used to localize error messages.

Parent elements	Child elements	Data type	Number allowed
<Response>	None	String	0...1 (optional)

The string **MUST** be of the form "en:en". <2>

2.2.1.1.2.3 **DebugData**

The **DebugData** element represents more information about the failure that can help the system administrator debug the source of the problem.

Parent elements	Child elements	Data type	Number allowed
<Error>	None	String	0...1 (optional)

Developers SHOULD NOT use this element for their own debugging.

2.2.1.1.2.4 DisplayName

The **DisplayName** element contains the user's display name in the directory service.

Parent elements	Child elements	Data type	Number allowed
<User>	None	String	0...1 (optional)

The client can choose to display or store this value on the device.

2.2.1.1.2.5 EmailAddress

The **EmailAddress** element contains the **SMTP** e-mail address of the user and is used to identify the user's **mailbox** in the network.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only) <User> (response only)	None	String	1...1 (required)

If the user has multiple addresses, then the primary e-mail address is returned in the **Autodiscover** command response. This address can be the same as the e-mail address that was sent in the request. The client device SHOULD record this address and SHOULD use this string for all additional communication.

2.2.1.1.2.6 Error

The **Error** element contains the error that was encountered while processing the request.

Parent elements	Child elements	Data type	Number allowed
<Action> (response only)	<Status> <Message>	Container	0...1 (optional)

	<DebugData>		
--	-------------	--	--

2.2.1.1.2.7 Message

The **Message** element contains the error string localized using the **Culture** specified in the **Response** element, enabling the client to display error status to the end-user.

Parent elements	Child elements	Data type	Number allowed
<Error>	None	String	0...1 (optional)

2.2.1.1.2.8 Name

The **Name** element specifies a URL if the **Type** element is set to **MobileSync**.

Parent elements	Child elements	Data type	Number allowed
<Server> (response only)	None	String	0...1 (optional)

If the **Type** element value is **MobileSync**, then the **Name** element specifies the URL that conveys the protocol. If the **Type** element value is **CertEnroll**, then the **Name** value is NULL.

2.2.1.1.2.9 Redirect

The **Redirect** element specifies the **SMTP** address of the requested user.

Parent elements	Child elements	Data type	Number allowed
<Action>	None	String	0...1 (optional)

The **Redirect** element is an optional child of the **Action** element in the **Autodiscover** response message. The client device SHOULD use the **domain** part of the address to send a new **Autodiscover** command request.

2.2.1.1.2.10 Response

The **Response** element contains the **Autodiscover** command response parameters.

Parent elements	Child elements	Data type	Number allowed
<Autodiscover>	<User>	Container	1...N (required)

	<Culture> <Action> <Error>		
--	----------------------------------	--	--

If an error occurs in the **Autodiscover** command framework that hosts the Autodiscovery implementation, then the **Response** element MUST have an **Error** child node.

The **Autodiscover** command MUST accept more than one **Response** element in a command response. The server MUST service requests in the order specified and mirror the order in the response. In this case, the order of the **Response** elements matches the order of the corresponding **Request** elements in the **Autodiscover** command request.

2.2.1.1.2.11 Server

The **Server** element encapsulates settings that apply to a particular server in the **Autodiscover** command response.

Parent elements	Child elements	Data type	Number allowed
<Settings> (response only)	<Type> <Url> <Name> <ServerData>	Container	1...N (required)

2.2.1.1.2.12 ServerData

The **ServerData** element contains the template name for the client certificate.<3>

Parent elements	Child elements	Data type	Number allowed
<Server> (response only)	None	String	0...1 (optional)

The **ServerData** element is a string that is present only when the **Type** element value is set to **CertEnroll**.

2.2.1.1.2.13 Settings

The **Settings** element contains the settings for the specified user or schema.

Parent elements	Child elements	Data type	Number allowed
-----------------	----------------	-----------	----------------

<Action> (response only)	Settings that are specific to requested service	Container	0...1 (optional)
--------------------------	---	------------------	------------------

2.2.1.1.2.14 Status

The **Status** element provides a status code that corresponds to the error.

Parent elements	Child elements	Data type	Number allowed
<Error>	None	Integer	0...1 (optional)

The following table specifies valid values for the **Status** element in the context of the **Settings** element.

Value	Meaning
1	Success
2	Protocol error

The client device can implement custom recovery logic pertaining to the status code. The client device **MUST** handle all unknown status codes.

2.2.1.1.2.15 Type

The **Type** element specifies the server type.

Parent elements	Child elements	Data type	Number allowed
<Server> (response only)	None	String	0...1 (optional)

The following are the valid values for the **Type** element:

- MobileSync. Indicates that the URL that is returned by the URL element can be accessed by clients.<4>
- CertEnroll. Indicates that the URL that is returned by the URL element can be accessed by clients that have a valid certificate over a **Secure Socket Layer (SSL)**.<5>

If the server supports both MobileSync and CertEnroll, the response buffer includes multiple **Server** elements that contain a **URL** value for each **Type** value.

2.2.1.1.2.16 Url

The **Url** element contains a URL string that conveys the protocol, port, resource location, and other information.

Parent elements	Child elements	Data type	Number allowed
<Server> (response only)	None	String	0...1 (optional)

The **Url** element is a child of the **Server** element. The value is a URL string that conveys the protocol, port, resource location, and other information. <6>

2.2.1.1.2.17 User

The **User** element encapsulates information about the user to whom this response element relates.

Parent elements	Child elements	Data type	Number allowed
<Response>	<DisplayName> <EmailAddress>	Container	1...1 (required)

2.2.1.2 FolderCreate

The **FolderCreate** command creates a new folder as a child of the specified parent folder. A parent ID of 0 signifies the **mailbox** root folder.

2.2.1.2.1 Request

The server that is implementing [MS-ASCMD] enforces the following **XML** schema definition (XSD) when processing protocol requests.

Requests that do not adhere to the schema result in the return of a status 4 to the client.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FolderCreate">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SyncKey">
          <xs:simpleType>
```

```

        <xs:restriction base="xs:string">
            <xs:maxLength value="64"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="ParentId">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="64"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="DisplayName">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="256"/>
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="Type" type="xs:unsignedByte" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.2.1.1 FolderCreate

The **FolderCreate** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **FolderCreate** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <ParentId> (request	Container	1 (required)

	only) <DisplayName> (request only) <Type> (request only) <ServerId> (response only) <Status> (response only)		
--	--	--	--

2.2.1.2.1.2 SyncKey

The **SyncKey** element specified in the **FolderCreate** command request represents the synchronization state of a **collection**. After a successful **FolderCreate** command, the server sends a synchronization key to the client in a response. The client **MUST** store this key and send it back to the server the next time the folder hierarchy is synchronized or updated. The server checks the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate>	None	String (Up to 64 characters)	Request: 1 (required) Response: 0...N

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

The **SyncKey** element is returned if the **FolderCreate** command request was successful and the element is not returned if the **FolderCreate** command request fails.

2.2.1.2.1.3 ParentId

The **ParentId** element specifies the server ID of the parent folder and is used in **FolderCreate** command requests only. The server ID of the parent folder is obtained from the **ServerId** element of a previous **FolderSync** command. A parent ID of 0 signifies the **mailbox** root folder.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate> (request only)	None	String (Up to 64 characters)	1 (required)

2.2.1.2.1.4 DisplayName

The **DisplayName** element specifies the name of the folder that is shown to the user.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate> (request only)	None	String (Between 1 and 256 characters)	1 (required)

2.2.1.2.1.5 Type

The **Type** element specifies the type of the folder to be created.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate> (request only)	None	Integer	1 (required)

The folder type values are listed in the following table. Folder types 2–11 are reserved for default folder types.

Type	Definition
1	User-created folder (generic)
12	User-created mail folder
13	User-created calendar folder
14	User-created contacts folder
15	User-created tasks folder
16	User-created journal folder
17	User-created notes folder
18	Unknown folder

2.2.1.2.2 Response

The following code shows the XSD for the **FolderCreate** command response.

```
<?xml version="1.0" ?>
```

```

<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="FolderCreate">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Status" type="xs:unsignedByte" />
        <xs:element minOccurs="0" name="SyncKey"
type="xs:string" />
        <xs:element minOccurs="0" name="ServerId"
type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

2.2.1.2.2.1 FolderCreate

The **FolderCreate** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **FolderCreate** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <ParentId> (request only) <DisplayName> (request only) <Type> (request only) <ServerId> (response only) <Status> (response only)	Container	1 (required)

2.2.1.2.2.2 ServerId

The **ServerId** element uniquely identifies a new folder on a server. The server ID of the new folder is returned to the client after a successful **FolderCreate** command request. The server ID can also be used in the **ServerId** element of future **FolderDelete** and **FolderUpdate** command requests. The client MUST store the server ID for each object and MUST be able to locate an object given a server ID.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate> (response only)	None	String (Up to 64 characters)	0...1 (optional)

The client MUST store the server ID as an opaque string of up to 64 characters.

The **ServerId** element MUST be returned if the **FolderCreate** command request was successful and the element MUST NOT be returned if the **FolderCreate** command request fails.

2.2.1.2.2.3 Status

The **Status** element indicates in the **FolderCreate** command response the success or failure of a **FolderCreate** command request. If the command failed, the **Status** element contains a code indicating the type of failure. The values are summarized in the following table.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate> (response only)	None	Integer	1 (required)

The following table shows valid values for the element.

Value	Meaning
1	Success.
2	A folder with that name already exists.
5	The specified parent folder was not found.
6	An error on the server.
7	Access denied.
8	The request timed out.

9	Synchronization key mismatch or invalid synchronization key.
10	Incorrectly formatted request.
11	An unknown error occurred.

2.2.1.2.2.4 SyncKey

The **SyncKey** element specified in the **FolderCreate** command response represents the synchronization state of a **collection**.

Parent elements	Child elements	Data type	Number allowed
<FolderCreate>	None	String (Up to 64 characters)	Request: 1 (required) Response: 0..N

After a successful **FolderCreate** command, the server **MUST** send a synchronization key to the client in a response. If the **FolderCreate** command is not successful, the server **MUST NOT** return a **SyncKey** element.

The client **MUST** store this key and send it back to the server the next time the folder hierarchy is synchronized or updated. The server **MUST** check the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

2.2.1.3 FolderDelete

The **FolderDelete** command deletes a folder from the server. The server ID of the folder **MUST** be passed to the server in the **FolderDelete** command request, which deletes the **collection** with the matching identifier. The server then sends a response indicating the status of the deletion.

2.2.1.3.1 Request

The following code shows the XSD for the **FolderDelete** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```

<xs:element name="FolderDelete">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SyncKey" type="xs:string" />
      <xs:element name="ServerId" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.3.1.1 FolderDelete

The **FolderDelete** element is the top-level element in the **XML** document. It identifies the body of the **HTTP** Post as containing a **FolderDelete** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <ServerId> (request only) <Status> (response only)	Container	1 (required)

2.2.1.3.1.2 SyncKey

The **SyncKey** element represents the synchronization state of a folder hierarchy.

Parent elements	Child elements	Data type	Number allowed
<FolderDelete>	None	String (Up to 64 characters)	Request: 1 (required) Response: 0...N

After a successful **FolderDelete** command request, the server **MUST** send a synchronization key to the client in the response. If the **FolderDelete** command request is unsuccessful, the server **MUST NOT** return a **SyncKey** element.

The client **MUST** store this key and send it back to the server the next time the folder hierarchy is synchronized or updated. The server **MUST** check the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

2.2.1.3.1.3 **ServerId**

The **ServerId** element specifies the folder on the server to be deleted, and it is a unique identifier assigned by the server to each object that can be synchronized.

Parent elements	Child elements	Data type	Number allowed
<FolderDelete> (request only)	None	String (Up to 64 characters)	1 (required)

The server ID of the folder to be deleted is returned to the client in the **ServerId** element of a previous **FolderSync** or **FolderCreate** command. The client **MUST** store the server ID for each object and **MUST** be able to locate an object given a server ID.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

2.2.1.3.2 **Response**

The following code shows the XSD for the **FolderDelete** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FolderDelete">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Status" type="xs:unsignedByte" />
        <xs:element minOccurs="0" name="SyncKey"
type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.2.1.3.2.1 **FolderDelete**

The **FolderDelete** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **FolderDelete** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <ServerId> (request only) <Status> (response only)	Container	1 (required)

2.2.1.3.2.2 SyncKey

The **SyncKey** element is used by the server to mark the synchronization state of a folder hierarchy.

Parent elements	Child elements	Data type	Number allowed
<FolderDelete>	None	String (Up to 64 characters)	Request: 1 (required) Response: 0...N

After a successful **FolderDelete** command, the server MUST send a synchronization key to the client in a response. If the **FolderDelete** command is not successful, the server MUST NOT return a **SyncKey** element.

The client MUST store this key and send it back to the server the next time the folder hierarchy is synchronized or updated. The server MUST check the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server MUST provide an error if the **SyncKey** values do not match.

2.2.1.3.2.3 Status

The **Status** element indicates the success or failure of the **FolderDelete** command request. If the command failed, the Status element in the server response contains a code indicating the type of failure.

Parent elements	Child elements	Data type	Number allowed
<FolderDelete> (response)	None	Integer	1 (required)

The following table lists the valid values for this element.

Value	Meaning
1	Success.
3	The specified folder is the Inbox , Outbox , Contacts , or Drafts folder.
4	The specified folder does not exist.
6	An error occurred on the server.
7	Access denied.
8	The request timed out.
9	Synchronization key mismatch or invalid synchronization key.
10	Incorrectly formatted request.
11	An unknown error occurred.

2.2.1.4 FolderSync

The **FolderSync** command synchronizes the **collection** hierarchy but does not synchronize the items in the collections themselves.

This command works similarly to the **Sync** command. An initial **FolderSync** command with a synchronization key of 0 (value of 0 in **SyncKey** element) is required in order to obtain the list of folders and the synchronization key associated with that list. The synchronization key **MUST** be returned in the **SyncKey** element of the response. This synchronization key **MUST** be used in subsequent **FolderSync** commands to obtain folder hierarchy changes.

Unlike a **Sync** request, there is no **GetChanges** element submitted in the **FolderSync** request to get changes from the server. All folders **MUST** be returned to the client when initial folder synchronization is done with a synchronization key of 0.

2.2.1.4.1 Request

The following code shows the XSD for the **FolderSync** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified">
```

```

targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FolderSync">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SyncKey" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

2.2.1.4.1.1 FolderSync

The **FolderSync** element is the top-level element in the **XML** stream. It indicates that the body of the **HTTP POST** contains a **FolderSync** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <Status> (response only) <Changes> (response only)	Container	1 (required)

2.2.1.4.1.2 SyncKey

The **SyncKey** element is used by the server to track the current state of the client.

Parent elements	Child elements	Data type	Number allowed
<FolderSync>	None	String (Up to 64 characters)	1 (required)

After successful folder synchronization, the server **MUST** send a synchronization key to the client. The client **MUST** store this key and send the key back to the server the next time the folder hierarchy is synchronized or updated. The server **MUST** check the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

If a synchronization error occurs, the client SHOULD restart the synchronization process with a synchronization key of 0. The client data can then be merged with the data returned by the server, or the client data can be completely deleted and replaced with the data from the server.

2.2.1.4.2 *Response*

The following code shows the XSD for the **FolderSync** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FolderSync">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Status" type="xs:unsignedByte" />
        <xs:element minOccurs="0" name="SyncKey"
type="xs:string" />
        <xs:element minOccurs="0" name="Changes">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Count"
type="xs:unsignedByte" />
              <xs:element minOccurs="0"
name="Update">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element
name="ServerId" type="xs:string" />
                    <xs:element
name="ParentId" type="xs:string" />
                    <xs:element
name="DisplayName" type="xs:string" />
                    <xs:element
name="Type" type="xs:unsignedByte" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

        </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0"
name="Delete">
        <xs:complexType>
            <xs:sequence>
                <xs:element
name="ServerId" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0"
maxOccurs="unbounded" name="Add">
        <xs:complexType>
            <xs:sequence>
                <xs:element
name="ServerId" type="xs:string" />
                <xs:element
name="ParentId" type="xs:string" />
                <xs:element
name="DisplayName" type="xs:string" />
                <xs:element
name="Type" type="xs:unsignedByte" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.4.2.1 FolderSync

The **FolderSync** element is the top-level element in the **XML** stream. It indicates that the body of the **HTTP POST** contains a **FolderSync** command.

Parent elements	Child elements	Data type	Number allowed
None	SyncKey> <Status> (response only) <Changes> (response only)	Container	1 (required)

2.2.1.4.2.2 Status

The **Status** element indicates the success or failure of a **FolderSync** command request.

Parent elements	Child elements	Data type	Number allowed
<FolderSync> (response only)	None	Integer (See values in the following table)	1 (required)

If the command fails, the **Status** element contains a code that indicates the type of failure. The **Status** element is global for all returned **Collection** elements. If one **collection** fails, a failure status **MUST** be returned for all collections.

The following table lists the valid values for this element.

Value	Meaning
1	Success.
6	An error occurred on the server.
7	Access denied.
8	The request timed out.
9	Synchronization key mismatch or invalid synchronization key.
10	Incorrectly formatted request.

11	An unknown error occurred.
----	----------------------------

2.2.1.4.2.3 SyncKey

The **SyncKey** element is used by the server to track the current state of the client.

Parent elements	Child elements	Data type	Number allowed
<FolderSync>	None	String (Up to 64 characters)	1 (required)

After a successful folder synchronization, the server **MUST** send a synchronization key to the client. The client **MUST** store this key and send the key back to the server the next time the folder hierarchy is synchronized or updated. The server **MUST** check the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

If a synchronization error occurs, the client **SHOULD** restart the synchronization process with a synchronization key of 0. The client data can then be merged with the data returned by the server, or the client data can be completely deleted and replaced with the data from the server.

2.2.1.4.2.4 Changes

The **Changes** element is a container for changes to the folder hierarchy. It is used in the **FolderSync** command response to update the client with folder additions, deletions, and updates on the server.

The server **SHOULD** maintain the same set of folder data being returned across synchronization key 0, in terms of **ServerId** and **DisplayName** mapping. However, if an error occurs, the server can return a totally different set.

Parent elements	Child elements	Data type	Number allowed
<FolderSync> (response only)	<Count> <Add> <Delete> <Update>	Container	1 (required)

2.2.1.4.2.5 Count

The **Count** element is used in the **FolderSync** command response to list the number of added, deleted, and updated folders on the server since the last folder synchronization. These changes are listed in the **Changes** element. If there are no changes since the last folder synchronization, a count of 0 is returned.

Parent elements	Child elements	Data type	Number allowed
<Changes> (response only)	None	Unsigned Integer	1 (required)

2.2.1.4.2.6 Delete

The **Delete** element is used in the **FolderSync** command response to specify that a folder on the server was deleted since the last folder synchronization.

Parent elements	Child elements	Data type	Number allowed
<Changes> (response only)	<ServerId>	Container	0...N (optional)

2.2.1.4.2.7 Add

The **Add** element is used in a **FolderSync** command response to create a new folder on the client. Child elements of the **Add** element specify the server ID of the folder, the server ID of the parent folder, the display name of the folder, and the type of folder.

Parent elements	Child elements	Data type	Number allowed
<Changes> (response only)	<ServerId/> <ParentId/> <DisplayName/> <Type/>	Container	0...N (optional)

2.2.1.4.2.8 ServerId

The **ServerId** element specifies the server-unique identifier for a folder on the server.

Parent elements	Child elements	Data type	Number allowed
-----------------	----------------	-----------	----------------

<Add> (response only) <Delete> (response only) <Update> (response only)	None	String (Up to 64 characters)	1 (required)
---	------	-------------------------------------	--------------

The **ServerId** element is used to identify folders that have been added, deleted, or updated on the server in the **FolderSync** command response.

The client **MUST** store the server ID as an opaque string of up to 64 characters.

2.2.1.4.2.9 ParentId

The **ParentId** element specifies the server ID of the parent of the folder on the server that has been added or updated.

Parent elements	Child elements	Data type	Number allowed
<Add> (response only) <Update> (response only)	None	String (Up to 64 characters)	1 (required)

The client **MUST** store the parent ID as an opaque string of up to 64 characters.

2.2.1.4.2.10 DisplayName

The **DisplayName** element specifies the name of the folder that is shown to the user.

Parent elements	Child elements	Data type	Number allowed
<Add> (response only) <Update> (response only)	None	String	1 (required)

The **DisplayName** element is used in the **Add** and **Update** elements of **FolderSync** responses when a folder has been added or updated on the server. Subfolder display names **MUST** be unique within a folder.

2.2.1.4.2.11 Type

The **Type** element specifies the type of the folder that was added or updated (renamed or moved) on the server.

Parent elements	Child elements	Data type	Number allowed
<Add> (response only) <Update> (response only)	None	Integer	1 (required)

The folder type values are listed in the following table.

Value	Meaning
1	User-created folder (generic)
2	Default Inbox folder
3	Default Drafts folder
4	Default Deleted Items folder
5	Default Sent Mail folder
6	Default Outbox folder
7	Default Tasks folder
8	Default Calendar folder
9	Default Contacts folder
10	Default Notes folder
11	Default Journal folder
12	User-created Mail folder
13	User-created Calendar folder
14	User-created Contacts folder
15	User-created Tasks folder

16	User-created Journal folder
17	User-created Notes folder
18	Unknown folder type

2.2.1.4.2.12 Update

The **Update** element is used in a **FolderSync** command response to identify a folder on the server that has been updated (renamed or moved).

Parent elements	Child elements	Data type	Number allowed
<Changes> (response only)	<ServerId/> <ParentId/> <DisplayName/> <Type/>	Container	0...N (optional)

The child elements of the **Update** element identify the server ID of the folder that was updated, the server ID of its parent folder, the new display name of the updated folder, and the folder type.

2.2.1.5 FolderUpdate

The **FolderUpdate** command moves a folder from one location to another on the server. The command is also used to rename a folder.

2.2.1.5.1 Request

The following code shows the XSD for the **FolderUpdate** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FolderUpdate">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SyncKey" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

        <xs:element name="ServerId" type="xs:string" />
        <xs:element name="ParentId" type="xs:string" />
        <xs:element name="DisplayName" type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.5.1.1 FolderUpdate

The **FolderUpdate** element is the top-level element in the **XML** stream. It indicates that the body of the **HTTP POST** contains a **FolderUpdate** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <ServerId> (request only) <ParentId> (request only) <DisplayName> (request only) <Status> (response only)	Container	1 (required)

2.2.1.5.1.2 SyncKey

The **SyncKey** element is used by the server to track the current state of the client.

Parent elements	Child elements	Data type	Number allowed
<FolderUpdate>	None	String (Up to 64 characters)	1 (required)

After a successful **FolderUpdate** command, the server **MUST** send a new synchronization key to the client. If the **FolderUpdate** command was not successful, the server **MUST NOT** return a **SyncKey** element.

The client **MUST** store this key and send the key back to the server the next time the folder hierarchy is synchronized or updated. The server **MUST** check the value of the key to make

sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

2.2.1.5.1.3 ServerId

The **ServerId** element identifies the folder on the server to be renamed or moved.

Parent elements	Child elements	Data type	Number allowed
<FolderUpdate> (request only)	None	String (Up to 64 characters)	1 (required)

The server ID is obtained from the **ServerId** element of a previous **FolderSync** or **FolderUpdate** command. The server ID specifies a unique identifier assigned by the server to each object that can be synchronized. The client **MUST** store the server ID for each object and **MUST** be able to locate an object given a server ID.

The client **MUST** store the server ID as an opaque string of up to 64 characters.

2.2.1.5.1.4 ParentId

The **ParentId** element specifies the server ID of the parent of the folder to be renamed or the destination folder of the folder to be moved.

Parent elements	Child elements	Data type	Number allowed
<FolderUpdate> (request only)	None	String (Up to 64 characters)	1 (required)

The parent ID is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command. The client **MUST** store the parent ID as an opaque string of up to 64 characters.

A parent ID of 0 signifies the **mailbox** root folder.

2.2.1.5.1.5 DisplayName

The **DisplayName** element specifies the name of the folder that is shown to the user.

Parent elements	Child elements	Data type	Number allowed
<FolderUpdate> (request only)	None	String	1 (required)

2.2.1.5.2 Response

The following code shows the XSD for the **FolderUpdate** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="FolderHierarchy:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="FolderHierarchy:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FolderUpdate">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Status" type="xs:unsignedByte" />
        <xs:element minOccurs="0" name="SyncKey"
type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.2.1.5.2.1 FolderUpdate

The **FolderUpdate** element is the top-level element in the **XML** stream. It indicates that the body of the **HTTP POST** contains a **FolderUpdate** command.

Parent elements	Child elements	Data type	Number allowed
None	<SyncKey> <ServerId> (request only) <ParentId> (request only) <DisplayName> (request only) <Status> (response only)	Container	1 (required)

2.2.1.5.2.2 Status

The **Status** element indicates the success or failure of a **FolderSync** command request.

Parent elements	Child elements	Data type	Number allowed
<FolderUpdate> (response only)	None	Integer (See values in the following table)	1 (required)

If the command fails, the **Status** element contains a code that indicates the type of failure. The **Status** element is global for all returned **Collection** elements. If one **collection** fails, a failure status is returned for all collections.

The following table lists the valid values for this element.

Value	Meaning
1	Success.
2	A folder with that name already exists.
3	The specified folder is the Inbox , Outbox , Contacts , or Drafts folder.
4	The specified folder does not exist.
5	The specified parent folder was not found.
6	An error occurred on the server.
7	Access denied.
8	The request timed out.
9	Synchronization key mismatch or invalid synchronization key.
10	Incorrectly formatted request.
11	An unknown error occurred.

2.2.1.5.2.3 SyncKey

The **SyncKey** element is used by the server to track the current state of the client.

Parent elements	Child elements	Data type	Number allowed
<FolderUpdate>	None	String (Up to 64 characters)	1 (required)

After a successful **FolderUpdate** command, the server **MUST** send a new synchronization key to the client. If the **FolderUpdate** command was not successful, the server **MUST NOT** return a **SyncKey** element.

The client **MUST** store this key and send the key back to the server the next time the folder hierarchy is synchronized or updated. The server **MUST** check the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

2.2.1.6 GetAttachment

The **GetAttachment** command retrieves an e-mail **attachment** from the server.

Attachments are not automatically included with e-mail messages in a synchronization; they are explicitly retrieved by using the **GetAttachment** command.

This command is issued within the **HTTP POST** command, and does not require any additional information in an **XML** body. The name of the attachment to be retrieved is specified in the *AttachmentName* command parameter. The content of the attachment is returned in the response body with the content type being specified in the Content-Type header of the response. When the Content Type header is missing, this indicates that the default encoding of 7-bit **ASCII** has been used.

If the **GetAttachment** command is used to retrieve an attachment that has been deleted on the server, a 500 status code is returned in the **HTTP POST** response.

2.2.1.6.1 Request

No **XML** body is included in the **GetAttachment** command request.

2.2.1.6.2 Response

No **XML** body is included in the **GetAttachment** command response.

2.2.1.7 GetItemEstimate

This command gets an estimate of the number of items in a **collection** or folder on the server that have to be synchronized.

2.2.1.7.1 Request

The following code shows the XSD for the **GetItemEstimate** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="GetItemEstimate:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="GetItemEstimate:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="AirSync:" />
  <xs:element name="GetItemEstimate">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Collections">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Collection">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element
name="CollectionId" type="xs:string" />
                    <xs:element
xmlns:q1="AirSync:" ref="q1:FilterType" />
                    <xs:element
xmlns:q2="AirSync:" ref="q2:SyncKey" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.2.1.7.1.1 GetItemEstimate

The **GetItemEstimate** element is the top-level element in the **XML** stream. It indicates that the body of the **HTTP POST** contains a **GetItemEstimate** command.

Parent elements	Child elements	Data type	Number allowed
None	<Collections> (request only) <Response> (response only)	Container	1 (required)

2.2.1.7.1.2 Collections

The **Collections** element serves as a container for one to 300 **Collection** elements.

Parent elements	Child elements	Data type	Number allowed
<GetItemEstimate> (request only)	<Collection>	Container	1 (required)

2.2.1.7.1.3 Collection

The **Collection** element wraps elements that apply to a particular **collection**. A maximum of 300 **Collection** elements can be included in a single **Collections** element.

Parent elements	Child elements	Data type	Number allowed
<Collections> (request only) <Response> (response only)	<SyncKey/> (request only) <CollectionId/> <FilterType/> (request only) <Estimate/> (response only)	Container	1...n (required)

2.2.1.7.1.4 CollectionId

The required element **CollectionId** specifies the server ID of the **collection** from which the item estimate is being obtained.

Parent elements	Child elements	Data type	Number allowed
<Collection>	None	String (Up to 64 characters)	1...1 (required)

The collection ID is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command. The **CollectionId** element is used in both **GetItemEstimate** command requests and responses.

2.2.1.7.1.5 FilterType

The **FilterType** element specifies an optional time window in the **GetItemEstimate** command request for the objects sent from the server to the client.

Parent elements	Child elements	Data type	Number allowed
<Collection> (request only)	None	Integer	0...1 (optional)

The filter type applies to e-mail and calendar collections. If a filter type is specified, the server sends an estimate of the items within the filter specifications.

If the **FilterType** element is present in the request, the server manages objects on the client to maintain the specified time window. New objects are added to the client when they are within the time window. If the **FilterType** element is omitted, all objects are sent from the server.

Calendar items that are in the future or that have recurrence, but no end date, are sent to the client regardless of the filter type value

The valid values are listed in the following table.

Value	Meaning	Applies to E-mail	Applies to calendar
0	No filter	Yes	Yes
1	1 day ago	Yes	No
2	3 days ago	Yes	No
3	1 week ago	Yes	No
4	2 weeks ago	Yes	Yes

5	1 month ago	Yes	Yes
6	3 months ago	No	Yes
7	6 months ago	No	Yes

2.2.1.7.1.6 SyncKey

The **SyncKey** element represents the current state of a **collection**. The value of the element is examined by the server to determine the state of the synchronization process.

After a successful **GetItemEstimate** command, the server **MUST** send a new synchronization key to the client. If the **GetItemEstimate** command was not successful, the server **MUST NOT** return a **SyncKey** element.

The client **MUST** store this key and send the key back to the server the next time the folder hierarchy is synchronized or updated. The server checks the value of the key to make sure the value of the **SyncKey** provided in the request matches a **SyncKey** value on the server. The server **MUST** provide an error if the **SyncKey** values do not match.

Parent elements	Child elements	Data type	Number allowed
<Collection> (request only)	None	String (Up to 64 characters)	1 (required)

2.2.1.7.2 Response

The following code shows the XSD for the **GetItemEstimate** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="GetItemEstimate:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="GetItemEstimate:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GetItemEstimate">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Response">
          <xs:complexType>
            <xs:sequence>
```

```

type="xs:unsignedByte" />
name="Collection">
    <xs:element name="Status"
    <xs:element minOccurs="0"
    <xs:complexType>
        <xs:sequence>
            <xs:element
name="CollectionId" type="xs:string" />
            <xs:element
name="Estimate" type="xs:unsignedByte" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.7.2.1 GetItemEstimate

The **GetItemEstimate** element is the top-level element in the **XML** stream. It indicates that the body of the **HTTP POST** contains a **GetItemEstimate** command.

2.2.1.7.2.2 Response

The **Response** element wraps elements that describe estimated changes. Its child elements specify the status of the **GetItemEstimate** command request and information about the collection on which the estimate was made.

Parent elements	Child elements	Data type	Number allowed
<GetItemEstimate> (response only)	<Collection> <Status/>	Container	1 (required)

2.2.1.7.2.3 Status

The **Status** element indicates the success or failure of a **GetItemEstimate** command request.

Parent elements	Child elements	Data type	Number allowed
<Response> (response only)	None	Integer	1 (required)

If the command fails, the **Status** element contains a code that indicates the type of failure. The **Status** element is global for all returned **Collection** elements. If one **collection** fails, a failure status is returned for all collections.

The following table lists the valid values for the element.

Value	Meaning
1	Success.
2	A collection was invalid or one of the specified collection IDs was invalid.
3	Synchronization state has not been primed yet. The Sync command MUST be performed first.
4	The specified synchronization key was invalid

2.2.1.7.2.4 Collection

The **Collection** element wraps elements that apply to a particular **collection**. A maximum of 300 **Collection** elements can be included in a single **Collections** element.

Parent elements	Child elements	Data type	Number allowed
<Collections> (request only) <Response> (response only)	<SyncKey/> (request only) <CollectionId/> <FilterType/> (request only) <Estimate/>	Container	1...n (required)

	(response only)		
--	-----------------	--	--

2.2.1.7.2.5 Class

The **Class** element specifies the type of a **collection**. The valid values are listed in the following table.

Value	Meaning
"Tasks"	A collection of tasks.
"Email"	A collection of e-mail messages.
"Calendar"	A collection of calendar items.
"Contacts"	A collection of contacts .
"Document"	A collection of file-share documents.

2.2.1.7.2.6 CollectionId

The **CollectionId** element specifies the server ID of the **collection** from which the item estimate is being obtained.

The collection ID is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command. The **CollectionId** element is used in both **GetItemEstimate** command requests and responses.

2.2.1.7.2.7 Estimate

The **Estimate** element specifies the estimated number of items in the **collection**/folder that have to be synchronized.

Parent elements	Child elements	Data type	Number allowed
<Collection> (response only)	None	Integer	1 (required)

2.2.1.8 ItemOperations

The **ItemOperations** command acts as a container for the **Fetch** command and the **EmptyFolderContents** command to provide batched online operations of these commands against the server.

Operations that are contained within the **ItemOperations** node **MUST** be executed by the server in the specified order. The server **MUST** report the status per operation to the client. Accordingly, the client correlates these responses to the initial operation and proceeds appropriately.

The **Fetch** operation is intended to be used on Microsoft Windows® SharePoint® Services technology or **Universal Naming Convention (UNC)** document metadata, search results, and items and **attachments**.

The **EmptyFolderContents** operation enables the client to empty a folder of all its items. Clients use **EmptyFolderContents** specifically to clear out all items in the Deleted Items folder if the user runs out of storage quota.

2.2.1.8.1 Delivery of Content Requested by Fetch

Because the **ItemOperations** response potentially contains large amounts of binary data, the client can choose a delivery method that is most efficient for its implementation by providing the following two methods for delivering the content that is requested by the **Fetch** command:

- Inline
- Multipart

Inline

The inline method of delivering binary content is including Base64-encoded data inside the WBXML. The inline approach generally requires the client to read the whole response into memory in order to parse it, thereby consuming a large amount of memory.

Multipart

The multipart method of delivering content is a multipart structure with the WBXML being the first part, and the requested data populating the subsequent parts. This format enables a client to handle large files without consuming large amounts of RAM, because a file is read in pieces, one piece at a time.

The multipart approach enables the client to parse the small WBXML part, obtain references to the binary parts, and handle the binary parts as necessary, without reading the entire response into memory.

Multipart Request

If the client wants to have the document or documents returned in multipart format, the only difference between this request and the inline content request is the addition of the following **HTTP** header:

MS-ASAcceptMultiPart: T

If this header is not present, then the server uses the default of **FALSE**, and returns content inline. If the header is set to **TRUE**, then the server returns the document contents by using the multipart format.

The following is a sample request for the test.txt document in a **UNC** share:

```
POST /Microsoft-Server-
ActiveSync?Cmd=ItemOperations&User=administrator&DeviceId=v120Device&De
viceType=PocketPC
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1
MS-ASAcceptMultiPart: T
<ItemOperations>
  <Fetch>
    <Store>DocumentLibrary</Store>
    <LinkId>\\feed31\public\test.txt</LinkId>
  </Fetch>
</ItemOperations>
```

Multipart Response

At a high level, the multipart response consists of several key elements:

- HTTP headers that specify the content type (HTTP 'Content-Type' header) of the multipart response: application/vnd.ms-sync.multipart.
- Metadata consisting of a list of [integer, integer] tuples that specify the start and count of bytes, respectively, of each body part. The following is the format of the metadata:

'Number of Parts :' <number of body parts, including WBXML>

'Part' <part #> ':' <range>

Range specifies a [start, count] value that indicates the start and count of bytes for each body part. There is always at least one tuple, pointing to the WBXML response.

- The WBXML response, which contains status and application data for all requested items. The WBXML response is always the first part in the response. Items composed of binary content have a **Part** element that indicates the index (base 0) of the body part that corresponds to that item in the multipart structure. This index is used by the client to find the appropriate [start, count] entry in the metadata.
- Binary application data, which includes one or more binary data parts, the start and end byte of each of which is indicated in the WBXMLEX-Ranges header.

The following figure shows the elements of the multipart response.

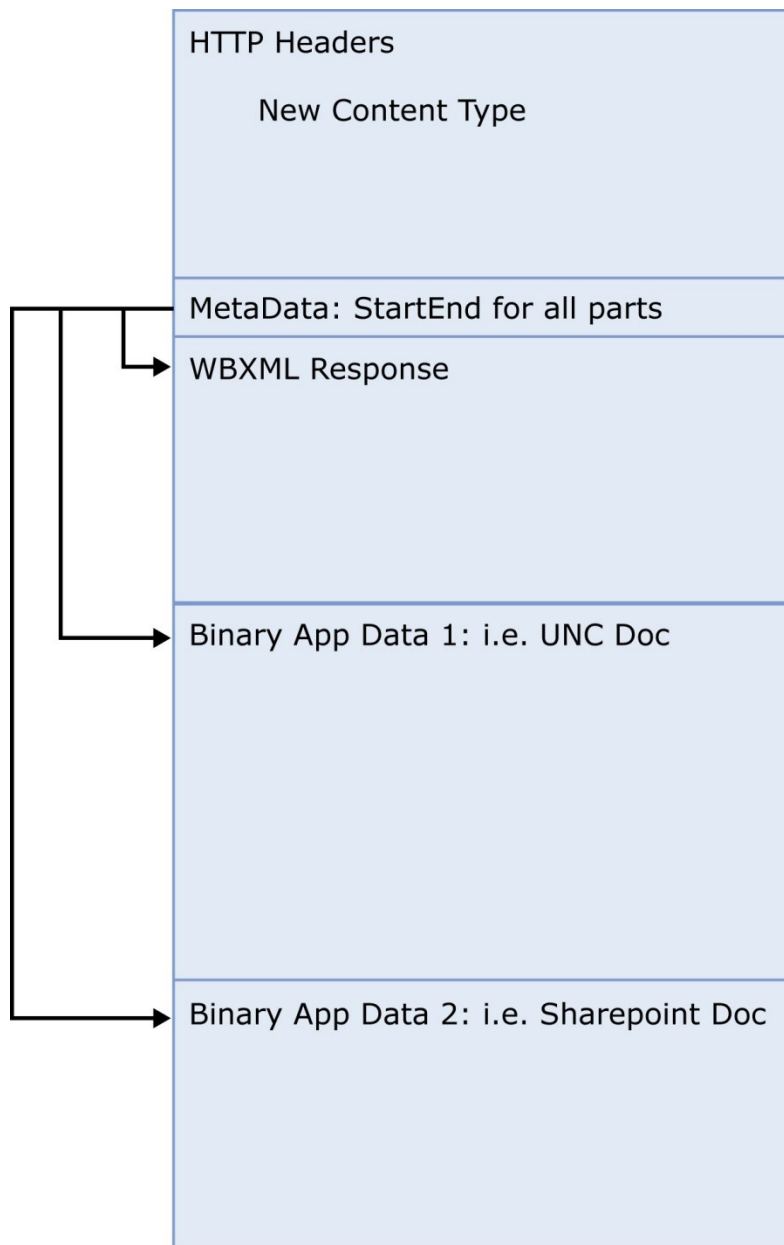


Figure 1: ItemOperations command multipart response

2.2.1.8.2 Request

The server that is implementing [MS-ASCMD] enforces the following XSD when it processes protocol requests.

Requests that do not follow the schema result in the return of a status 4 to the client. The following XSD is not applied by the server when it sends responses to the client.

```

<?xml version="1.0"?>
<xs:schema id="ItemOperations" targetNamespace="ItemOperations:"
xmlns:search="Search:" xmlns:calendar="Calendar:"
xmlns:contacts2="Contacts2:" xmlns:contacts="Contacts:"
xmlns:email="Email:" xmlns:mstns="ItemOperations:"
xmlns:airsyncbase="AirSyncBase:"
xmlns:documentLibrary="DocumentLibrary:" xmlns:airsync="AirSync:"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="qualified" elementFormDefault="qualified">
  <xs:import namespace="DocumentLibrary:"/>
  <xs:import namespace="AirSync:"/>
  <xs:import namespace="AirSyncBase:"/>
  <xs:import namespace="Email:"/>
  <xs:element name="ItemOperations">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="EmptyFolderContents">
          <xs:complexType>
            <xs:all>
              <xs:element ref="airsync:CollectionId" minOccurs="1"
maxOccurs="1"/>
              <xs:element name="Options" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="DeleteSubFolders"/>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
        <xs:element name="Fetch">
          <xs:complexType>
            <xs:all>

```

```

    <xs:element name="Store">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="1"/>
          <xs:maxLength value="256"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>

    <xs:element ref="airsync:ServerId" minOccurs="0"
maxOccurs="1"/>

    <xs:element ref="airsync:CollectionId" minOccurs="0"
maxOccurs="1"/>

    <xs:element ref="documentLibrary:LinkId" minOccurs="0"
maxOccurs="1"/>

    <xs:element ref="search:LongId" minOccurs="0"
maxOccurs="1"/>

    <xs:element ref="airsyncbase:FileReference" minOccurs="0"
maxOccurs="1"/>

    <xs:element name="Options" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="1"
name="Schema">

            <xs:complexType>
              <xs:choice maxOccurs="unbounded">
                <xs:group ref="email:TopLevelSchemaProps"/>
                <xs:group
ref="airsyncbase:TopLevelSchemaProps"/>
                <xs:group
ref="calendar:TopLevelSchemaProps"/>
                <xs:group
ref="contacts:TopLevelSchemaProps"/>
                <xs:group
ref="contacts2:TopLevelSchemaProps"/>
              </xs:choice>
            
```

```

        </xs:complexType>
    </xs:element>
    <xs:element name="Range" minOccurs="0"
maxOccurs="1">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:pattern value="[0-9]{1,9}-[0-9]{1,9}"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
<xs:element name="UserName" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="100" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element
name="Password" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="100" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>

```

```

        <xs:element ref="airsync:MIMESupport" minOccurs="0"
maxOccurs="1" />
        <xs:element ref="airsyncbase:BodyPreference"
minOccurs="0" maxOccurs="256" >
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="airsyncbase:Type"
minOccurs="0" maxOccurs="1" />
                    <xs:element ref="airsyncbase:TruncationSize"
minOccurs="0" maxOccurs="1" />
                    <xs:element ref="airsyncbase:AllOrNone"
minOccurs="0" maxOccurs="1" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:all>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.8.2.1 ItemOperations

The **ItemOperations** element is the top-level element in the **XML** document. The element identifies the body of the **HTTP** Post as containing a **ItemOperations** command.

Parent elements	Child elements	Data type	Number allowed
None	<Fetch> (request only) <EmptyFolderContents> (request only)	Container	1...1 (required)

	<Response> <response only> <Status> (response only)		
--	--	--	--

2.2.1.8.2.2 Fetch

The **Fetch** element retrieves an item from the server.

Parent elements	Child elements	Data type	Number allowed
<ItemOperations> (request) <Response> (response)	<Store> (request only) <LinkID/> (optional) <CollectionId> (optional) <ServerId> (optional) <Options/> <Status/> (response only) <Class/> (response only) <Properties/> (response only) <FileReference> (request only) <LongId> (request only)	Container	0...N (Optional)

The **Fetch** operation is intended to be used on Microsoft Windows® SharePoint® Services technology or **UNC** document metadata, search results, and items and **attachments**.

Because the **ItemOperations** response potentially contains large amounts of binary data, the client can choose a delivery method that is most efficient for its implementation by providing the following two methods to deliver content that is requested by the **Fetch** command:

- **Inline**—The binary content is Base64-encoded and is included inside the **WBXML**.

- **Multipart**—This method involves a multipart structure in which the WBXML is the first part, and the requested data populates the subsequent parts. This format enables a client to handle large files without consuming large amounts of RAM.

The inline approach generally requires the client to read the WBXML part into memory in order to parse it, thereby consuming a large amount of memory. The multipart approach enables the client to parse the small WBXML part, obtain references to the binary parts, and handle the binary parts as necessary, without reading the whole response into memory.

In the request, the client specifies the location and a byte range for the item. The location is indicated by either a link ID (**LinkId** element) if the target item is identified by a **Uniform Resource Identifier (URI)**, or a file reference (**FileReference** element) if the client is retrieving an e-mail **attachment**.<7>

The **Fetch** command supports several options, including:

- **Byte ranges**—The range of bytes for an item that is contained in a given **Fetch** command response. The specified range facilitates a checkpoint to improve the reliability of large data downloads. This option is supported for document library items and attachments; it is not supported for other item types.
- **Body preference**—Per-class settings on preferred body format. It is supported only for e-mail, **contact**, calendar, and task items; it is not supported for document library items or **attachments**.
- **Schema**—Per-class settings on format for search results. It is supported only for e-mail, contact, calendar, and task items; it is not supported for document library items or attachments.

The response contains either the requested byte range of the item, or an error code that indicates why the fetch failed. If the client tries to fetch a resource that is not an item (such as a document folder), the client receives an error.

Multiple **Fetch** operations can be included within one **ItemOperations** request. In this case, the **Fetch** operations are executed in the order that is specified.

2.2.1.8.2.3 EmptyFolderContents

The **EmptyFolderContents** element identifies the body of the request or response as containing the operation that deletes the contents of a folder.

Parent elements	Child elements	Data type	Number allowed
<ItemOperations> (request only) <Response> (response only)	<CollectionId> <Options> (request only) <Status> (response	Container	0...N (optional)

	only)		
--	-------	--	--

The **EmptyFolderContents** element enables the client to empty a folder of all its items. The element supports a single option, which is whether to delete subfolders contained in the folder (the default is not to delete subfolders).

Specifically, clients use **EmptyFolderContents** to empty the Deleted Items folder. The client can clear out all items in the Deleted Items folder when the user runs out of storage quota (generally indicated by the return of an **HTTP 507** status code from the server).

2.2.1.8.2.4 CollectionId

The **CollectionId** element enables a client to specify the folder to be emptied or the item to be fetched.

Parent elements	Child elements	Data type	Number allowed
<EmptyFolderContents> <Fetch> (request only)	None	String	0...1

2.2.1.8.2.5 Options

The **Options** element contains the options for its parent element. The child elements of **Options**, therefore, depend on its parent element and the store/item type that is being acted upon.

Parent elements	Child elements, <EmptyFolderContents> parent	Child elements, <Fetch> parent	Data type	Number allowed
<EmptyFolderContents> <Fetch>	<Range> <DeleteSubFolders>	<Range> <Schema> <airsyncbase:BodyPreference> <airsync:MIMESupport>	Container	0...1 (optional)

The following options are supported for **Fetch**:

- Byte ranges
 - Facilitates a checkpoint to improve the reliability of large data downloads.

- [MS-ASCMD] supports ranges for document library items and **attachments**; it does not support ranges for other item types—that is, **Personal Information Manager (PIM)** items, such as e-mail, **contact**, calendar, or task items.
- For attachments, the range applies to the file content.
- For document library items, this applies to the file content.
- Body preference
 - Per-**class** settings on preferred body format.
 - [MS-ASCMD] supports body preferences for PIM items only; it does not support body preferences for document library items or attachments.
- Schema
 - Per-class settings on format for search results.
 - [MS-ASCMD] supports schemas for PIM items only; it does not support schemas for document library items or attachments.
 - Supports all top-level property nodes.

If you specify an option that is invalid for the parent command, the server returns a protocol error.

2.2.1.8.2.6 DeleteSubFolders

The **DeleteSubFolders** element is a flag that indicates whether to delete the subfolders of the specified folder.

Parent elements	Child elements	Data type	Number allowed
<Options>	None	Flag	1...1 (required)

If the **DeleteSubFolders** element is not present in the request, the default behavior is to not delete subfolders.

2.2.1.8.2.7 Store

The **Store** element specifies the name of the store to which the parent operation applies.

Parent elements	Child elements	Data type	Number allowed
<Fetch> (request only)	None	Container	1...1 (optional)

The following values are valid for the **Store** element:

- Document Library (SharePoint and **UNC** links)
- **Mailbox** (items and **attachments**)

2.2.1.8.2.8 MIMESupport

The **MIMESupport** element is included in the **Options** element of a client **Fetch** command request to enable **MIME** support for e-mail items that are sent from the server to the client. For an example, see section 4.2.2.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	0...1 (optional)

The following table lists the valid values for this element.

Value	Meaning
0	Never send MIME data.
1	Send MIME data for S/ MIME messages only. Send regular body for all other messages.
2	Send MIME data for all messages. This flag could be used by clients to build a more rich and complete Inbox solution.

To support fetching of the full S/**MIME** message, the **Fetch** request **MUST** include the following elements in the **Options** element:

- The **MIMESupport** element to indicate to the server to return MIME for S/**MIME**-only/All/None messages.
- The **BodyPreference** element with its child element, **Type** having a value of 4 to inform the server that the device can read the MIME **binary large object (BLOB)**.

The server's response **MUST** include the **Body** element, which is a child of the **Properties** element. The **Body** element is a complex element and **MUST** contain the following child nodes in an S/**MIME** fetch response:

- The **Type** element with a value of 4 to inform the device that the data is a MIME BLOB.
- The **EstimatedDataSize** element to specify the rough total size of the data.
- The **Data** element that contains the full MIME BLOB.

For more details about the **Body** element or the **BodyPreference** element, see [MS-ASAIRS] sections 2.2.3.3 or 2.2.3.4, respectively.

2.2.1.8.2.9 LinkId

The **LinkId** element specifies a **URI** that is assigned by the server to certain resources, such as Windows SharePoint Services or **UNC** documents.

Parent elements	Child elements	Data type	Number allowed
<Fetch>	None	URI	0...1 (optional)

The client **MUST** store the **LinkID** that is retrieved by the **Sync** or **Search** command if the client will make calls by using the **LinkID** in the future. In an **ItemOperations** request, the **LinkId** element can be used by the **Fetch** command to refer to the location of an item.

2.2.1.8.2.10 ServerId

The **ServerId** element specifies a unique identifier that is assigned by the server to each object that can be synchronized or have an item operation applied to it.

Parent elements	Child elements	Data type	Number allowed
<Fetch>	None	String	0...1 (optional)

The client **MUST** store the server ID for any item that is retrieved by means of the **Sync** or **Search** command. In an **ItemOperations** request, the **ServerId** element can be used by the **Fetch** command to refer to the location of the item in question.

2.2.1.8.2.11 FileReference

The **FileReference** element specifies a unique identifier that is assigned by the server to each **attachment** to a given item.

Parent elements	Child elements	Data type	Number allowed
<Fetch> (request only)	None	String	0...1 (optional)

The client **MUST** store the file reference for any item that is retrieved by means of the **Sync** or **Search** command. In an **ItemOperations** request, only one **FileReference** identifier can exist per **Fetch** node. Violation of this constraint results in an error from the server. The client can, however, retrieve multiple attachments by using one **Fetch** node per attachment.

2.2.1.8.2.12 Schema

The **Schema** element specifies the schema of the item to be fetched.

Parent elements	Child elements	Data type	Number allowed
-----------------	----------------	-----------	----------------

<Options> (request only)	None	Container	0...1 (optional)
--------------------------	------	------------------	------------------

The **Schema** element is supported within options for PIM **Fetch** requests. It is not supported when the client is retrieving items from a document library or retrieving an **attachment**.

If **Schema** is not specified, the server allows all properties to be retrieved.

2.2.1.8.2.13 Range

In an **ItemOperations** request, the **Range** element specifies the range of bytes that the client can receive in response to the **Fetch** operation for a document library item. In an **ItemOperations** response, the **Range** element specifies the actual range of bytes for an item that is contained in a given **Fetch** operation.

Parent elements	Child elements	Data type	Number allowed
<Options> (request) <properties> (response)	None	String in the format 0-n, where n is the maximum value	0...1 (optional)

The server provides a best effort at fulfilling the request. Therefore, the client cannot assume that the byte-range that is specified in the request exactly matches the byte-range that is returned in the response. The byte-range that is specified by the server in the response is the authoritative value.

If **Range** is omitted in the **Fetch** request, the whole item is fetched.

2.2.1.8.3 Response

The following code shows the XSD for the **ItemOperations** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="ItemOperations:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="ItemOperations:" xmlns:DocumentLibrary
="DocumentLibrary:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="DocumentLibrary:" />
  <xs:element name="ItemOperations">
    <xs:complexType>
      <xs:sequence>
```

```

        <xs:element name="Status" type="xs:integer " minOccurs="1"
maxOccurs="1" />
        <xs:element name="Response" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="EmptyFolderContents" minOccurs="0"
maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Status" type="xs:integer"
minOccurs="1" maxOccurs="1" />
                                <xs:element ref="AirSync:CollectionId"
minOccurs="0" maxOccurs="1" />
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Fetch" minOccurs="0"
maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Status" type="xs:integer"
minOccurs="1" maxOccurs="1"/>
                                <xs:element ref="appl:LinkId" minOccurs="0"
maxOccurs="1"/>
                                <xs:element name="Properties" minOccurs="0"
maxOccurs="1">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="Range" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                            <xs:element name="Total" type="xs:integer"
minOccurs="0" maxOccurs="1" />
                                            <xs:element name="Data" type="xs:string"
minOccurs="0" maxOccurs="1" />

```

```

        <xs:element name="Version" type="xs:datetime"
minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.8.3.1 ItemOperations

The **ItemOperations** element is the top-level element in the **XML** document. The element identifies the body of the **HTTP** Post as containing an **ItemOperations** command.

Parent elements	Child elements	Data type	Number allowed
None	<Fetch> (request only) <EmptyFolderContents> (request only) <Response> <response only> <Status> (response only)	Container	1...1 (required)

2.2.1.8.3.2 Response

The **Response** element is a container for the operation responses.

Parent elements	Child elements	Data type	Number allowed
<ItemOperations> (response only)	<EmptyFolderContents>	Container	1...1 (required)

	<Fetch>		
--	---------	--	--

2.2.1.8.3.3 EmptyFolderContents

The **EmptyFolderContents** element identifies the body of the request or response as containing the operation that deletes the contents of a folder.

Parent elements	Child elements	Data type	Number allowed
<ItemOperations> (request only) <Response> (response only)	<CollectionId> <Options> (request only) <Status> (response only)	Container	0...N (optional)

2.2.1.8.3.4 CollectionId

The **CollectionId** element enables a client to specify the folder to be emptied.

Parent elements	Child elements	Data type	Number allowed
<EmptyFolderContents>	None	String	1...1

2.2.1.8.3.5 Fetch

The **Fetch** element retrieves an item from the server.

Parent elements	Child elements	Data type	Number allowed
<ItemOperations> (request) <Response> (response)	<Store> (request only) <LinkID/> (optional) <CollectionId> (optional) <ServerId> (optional) <Options/> <Status/> (response only) <Class/> (response	Container	0...N (optional)

	only) <Properties/> (response only) <FileReference> (request only)		
--	--	--	--

The **Fetch** operation is intended to be used on Microsoft Windows® SharePoint® Services technology or **UNC** document metadata, search results, and items and **attachments**.

Because the **ItemOperations** response potentially contains large amounts of binary data, the [MS-ASCMD] protocol enables the client to choose a delivery method that is most efficient for its implementation by providing the following two methods to deliver content that is requested by the **Fetch** command:

- **Inline**—The binary content is Base64-encoded and is included inside the **WBXML**.
- **Multipart**—This method involves a multipart structure in which the WBXML is the first part, and the requested data populates the subsequent parts. This format enables a client to handle large files without consuming large amounts of RAM.

The inline approach generally requires the client to read the WBXML part into memory in order to parse it, thereby consuming a large amount of memory. The multipart approach enables the client to parse the small WBXML part, obtain references to the binary parts, and handle the binary parts as necessary, without reading the whole response into memory.

In the request, the client specifies the location and a byte range for the item. The location is indicated by either a link ID (**LinkId** element) if the target item is identified by a **URI**, or a file reference (**FileReference** element) if the client is retrieving an e-mail **attachment**.<8>

The **Fetch** command supports several options, including:

- **Byte ranges**—The range of bytes for an item that is contained in a given **Fetch** command response. The specified range facilitates a checkpoint to improve the reliability of large data downloads. This option is supported for document library items and attachments; it is not supported for other item types.
- **Body preference**—Per-**class** settings on preferred body format. It is supported only for e-mail, **contact**, calendar, and task items; it is not supported for document library items or attachments.
- **Schema**—Per-class settings on format for search results. It is supported only for e-mail, contact, calendar, and task items; it is not supported for document library items or attachments.

The response contains either the requested byte range of the item, or an error code that indicates why the fetch failed. If the client tries to fetch a resource that is not an item (such as a document folder), it receives an error.

Multiple **Fetch** operations can be included within one **ItemOperations** request. In this case, the **Fetch** operations are executed in the order that is specified.

2.2.1.8.3.6 LinkId

The **LinkId** element specifies a **URI** that is assigned by the server to certain resources, such as Windows SharePoint Services or **UNC** documents.

Parent elements	Child elements	Data type	Number allowed
<Fetch>	None	URI	0...1 (optional)

The client **MUST** store the **LinkID** that is retrieved by the **Sync** or **Search** command if the client will make calls by using the **LinkID** in the future. In an **ItemOperations** request, the **LinkId** element can be used by the **Fetch** command to refer to the location of an item.

2.2.1.8.3.7 Class

In a response, the **Class** element indicates the class of the content of the fetched item.

Parent elements	Child elements	Data type	Number allowed
<Schema> (request only) <Fetch> (request only) <Content> (response only)	None	String	0...1 (optional)

The following are valid values for the **Class** element in a request or response:

- E-mail
- Contacts
- Calendar
- Tasks

2.2.1.8.3.8 Properties

The **Properties** element contains a list of the schema properties for the item that the client wants to have returned in the **Fetch** response.

Parent elements	Child elements	Data type	Number allowed
<Schema> (request	The schema	Container	1...1 (required)

only) <Fetch> (response only)	properties of the item being fetched (request only) <Data> (response only) <Part> (response only) <Version> (response only) <Total> (response only) <airsynbase:Body> (response only)		
----------------------------------	--	--	--

If an unsupported property is specified by the client, the server returns an error. If **Properties** is not specified, the server uses the synchronized schema for that item class for **Fetch** results.

2.2.1.8.3.9 Range

In an **ItemOperations** request, the **Range** element specifies the range of bytes that the client can receive in response to the **Fetch** operation for a document library item. In an **ItemOperations** response, the **Range** element specifies the actual range of bytes for an item that is contained in a given **Fetch** operation.

Parent elements	Child elements	Data type	Number allowed
<Options> (request) <Properties> (response)	None	String in the format 0-n, where n is the maximum value	0...1 (optional)

The server provides a best effort at fulfilling the request. Therefore, the client cannot assume that the byte-range that is specified in the request exactly matches the byte-range that is returned in the response. The byte-range that is specified by the server in the response is the authoritative value.

If **Range** is omitted in the **Fetch** request, the whole item is fetched.

2.2.1.8.3.10 Status

The **Status** element contains a code that indicates the success or failure of the **ItemOperations** command and the operations within the **ItemOperations** command.

Parent elements	Child elements	Data type	Number allowed
<EmptyFolderContents> <Fetch> <ItemOperations>	None	Integer	0...1

The following table lists the different status codes.

Status Code	Meaning
1	Success.
2	Protocol error - protocol violation/ XML validation error.
3	Server error.
4	Document library access - The specified URI is bad.
5	Document library - Access denied.
6	Document library - The object was not found.
7	Document library - Failed to connect to the server.
8	Document library - The byte-range is invalid or too large.
9	Document library - The store is unknown or unsupported.
10	Document library - The file is empty.
11	Document library - The requested data size is too large.
12	Document library - Failed to download file because of input/output (I/O) failure.
13	Mailbox fetch provider - The body preference option is invalid.
14	Mailbox fetch provider - The item failed conversion.
15	Attachment fetch provider - Attachment or attachment ID is invalid.
16	Policy-related - Server blocked access.

17	Empty folder contents - Partial success; the command completed partially.
18	Credentials required.

The status is specified for the **ItemOperations** response, and for each fetch operation or empty-folder-contents operation within the **ItemOperations** command. If the status is not returned for an operation, the client assumes success.

2.2.1.8.3.11 Data

The **Data** element is part of the response for the **Fetch** command and contains the item content for inline content responses.

Parent elements	Child elements	Data type	Number allowed
<Properties> (response only)	None	String	0...1 (optional)

The content of the **Data** element is a Base64-encoding of the binary document, **attachment**, or body data. The size of the data (in bytes) that is contained within the **Data** element is indicated by the **Range** element in the fetch response. The total size of the item (in bytes) is indicated by the **Total** element.

2.2.1.8.3.12 Part

The **Part** element specifies an integer index into the metadata of the multipart response.

Parent elements	Child elements	Data type	Number allowed
<Properties> (response only)	None	Integer	0...1 (optional)

The **Part** element is present only in a multipart **ItemOperations** response.

The **Part** element can be used to locate the [start, end] tuple that specifies the starting byte and ending byte for this item's binary content in the command response.

2.2.1.8.3.13 Version

The **Version** element is a date/time stamp that indicates the time at which a document item was last modified.

Parent elements	Child elements	Data type	Number allowed
<Properties>	None	DateTime	0...1 (optional)

(response only)			
-----------------	--	--	--

The **Version** element is present only in the response and only when **ItemOperations** is used to access a Windows SharePoint Services or **UNC** resource.

2.2.1.8.3.14 Total

The **Total** element indicates the total size of an item on the server, in bytes.

Parent elements	Child elements	Data type	Number allowed
<Properties> (response only)	None	Integer	0...1 (optional)

2.2.1.9 MeetingResponse

The **MeetingResponse** command is used to accept, tentatively accept, or decline a **meeting** request in the user's Inbox.

The **MeetingResponse** command can only be used when the **CollectionId** element is being used to synchronize the folder that contains the meeting request item.

The **SendMail** command can be used to send a message back to the meeting **organizer**, notifying him or her that the meeting request was accepted or declined.

2.2.1.9.1 Request

The following code shows the XSD for the **MeetingResponse** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="MeetingResponse:"
  attributeFormDefault="unqualified" elementFormDefault="qualified"
    targetNamespace="MeetingResponse:"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MeetingResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Request">
          <xs:complexType>
            <xs:all>
```

```

<xs:element name="UserResponse">
    <xs:simpleType>
        <xs:restriction
base="xs:unsignedByte">
            <xs:enumeration value="3"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element minOccurs="0" name="CollectionId">
    <xs:simpleType>
        <xs:restriction
base="xs:string">
            <xs:maxLength value="64"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element minOccurs="0" name="RequestId">
    <xs:simpleType>
        <xs:restriction
base="xs:string">
            <xs:maxLength value="64"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element><xs:element
ref="search:LongId" minOccurs="0" maxOccurs="1"/>
    </xs:all>
</xs:complexType>
</xs:element>
</xs:sequence>

```



```

        </xs:complexType>
    </xs:element>
</xs:schema>

```

2.2.1.9.1.1 CollectionId

The **ElementName** element specifies the folder that contains the **meeting** request.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only)	None	String (Up to 64 characters)	0...1 (Required, or optional if LongId is specified)

The **CollectionId** element specifies the **Inbox folder** in most implementations.

The **collection** ID is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command.

2.2.1.9.1.2 Request

The **Request** element is a container for elements in a **MeetingResponse** command request. Its child elements specify the **meeting** request that is being responded to, the response to that meeting request, and the folder on the server that the meeting request is located in.

Parent elements	Child elements	Data type	Number allowed
<MeetingResponse> (request only)	<UserResponse/> <CollectionId/> <RequestId/> <LongId/>	Container	1...n (required)

2.2.1.9.1.3 RequestId

The **RequestId** element specifies the server ID of the **meeting** request message item.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only) <Result> (response only)	None	String (Up to 64 characters)	0...1 (Required, or optional if LongId is specified)

When the client sends a **MeetingResponse** command request, the client includes a **RequestId** element to identify which **meeting** request is being responded to. The **RequestId** element is also returned in the response to the client along with the status of the user's response to the meeting request.

2.2.1.9.1.4 UserResponse

The **UserResponse** element indicates in the **MeetingResponse** command request whether the **meeting** is being accepted, tentatively accepted, or declined.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only)	None	Integer	1 (required)

The following table shows valid values for the element.

Value	Meaning
1	Accepted
2	Tentatively accepted
3	Declined

2.2.1.9.2 Response

The following code shows the XSD for the **MeetingResponse** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="MeetingResponse:"
  attributeFormDefault="unqualified" elementFormDefault="qualified"
    targetNamespace="MeetingResponse:"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MeetingResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Result">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId"
                type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

        <xs:element name="Status"
type="xs:unsignedByte" />

        <xs:element name="CalendarId"
type="xs:string" minOccurs="0" />
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.9.2.1 CalendarId

The **CalendarId** element specifies the server ID of the calendar item.

Parent elements	Child elements	Data type	Number allowed
<Result> (response only)	None	String (Up to 64 characters)	0...1

The following table shows valid values for the element.

Value	Meaning
1	Success

The **CalendarId** element is included in the **MeetingResponse** command response that is sent to the client if the **meeting** request was not declined. If the meeting is accepted or tentatively accepted, the server adds a new item to the calendar and returns its server ID in the **CalendarId** element in the response. If the client created a tentative meeting calendar item, the client **SHOULD** update that item with the new server ID. The client **SHOULD** also change the busy status from tentative to busy. When a meeting is accepted, the server also creates a new calendar item with the same server ID. This means there is a conflict that will be resolved the next time the calendar is synchronized.

If the meeting is declined, the response does not contain a calendar ID.

2.2.1.9.2.2 RequestId

The **RequestId** element specifies the server ID of the **meeting** request message item.

Parent elements	Child elements	Data type	Number allowed
<Request> (request only) <Result> (response only)	None	String (Up to 64 characters)	1 (required)

When the client sends a **MeetingResponse** command request, the client includes a **RequestId** element to identify which meeting request is being responded to. The **RequestId** element is also returned in the response to the client along with the status of the user's response to the meeting request.

2.2.1.9.2.3 Result

The **Result** element is a container for elements that are sent to the client in a **MeetingResponse** command response.

Parent elements	Child elements	Data type	Number allowed
<MeetingResponse> (response only)	<RequestId/> <Status/> <CalendarId/>	Container	1...n (required)

The **Result** element's child elements identify the **meeting** request message item on the server and the status of the response to the meeting request. If the meeting request is accepted, the server ID of the calendar item is also returned.

2.2.1.9.2.4 Status

The **Status** element indicates the success or failure of the **MeetingResponse** command request.

Parent elements	Child elements	Data type	Number allowed
<Result> (response only)	None	Integer	1...n (required)

The following table shows valid values for the element.

Value	Meaning
1	Success.

2	Invalid meeting request.
3	An error occurred on the mailbox .
4	Error occurred on the server.

The values for the **Status** element when sent in a response to the **MeetingResponse** command are the same as for the **Sync** command.

2.2.1.10 MoveItems

The **MoveItems** command moves an item or items from one folder on the server to another.

The item to be moved is identified by its server ID in the **MoveItems** command request. The source and destination folders are also identified by their server IDs in the command request. The **MoveItems** command response shows the status of the move, the message that was moved, and the new message ID.

When items are moved between folders on the server, the client receives **Delete** and **Add** operations the next time the client synchronizes the affected folders.

An item that has been successfully moved to a different folder can be assigned a new server ID by the server.

2.2.1.10.1 Request

The following code shows the XSD for the **MoveItems** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="Move:" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="Move:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MoveItems">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Move">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="SrcMsgId"
type="xs:string" />
              <xs:element name="SrcFldId"
type="xs:string" />
            

```

```

                                <xs:element name="DstFldId"
type="xs:string" />
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
</xs:schema>

```

2.2.1.10.1.1 DstFldId

The **DstFldId** element specifies the server ID of the destination folder (that is, the folder to which the items are moved).

Parent elements	Child elements	Data type	Number allowed
<Move> (request only)	None	String (Up to 64 characters)	1 (required)

The server ID of the destination folder is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command.

2.2.1.10.1.2 MoveItems

The **MoveItems** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **MoveItems** command.

Parent elements	Child elements	Data type	Number allowed
None	<Move> (request only) <Response> (response only)	Container	1 (required)

2.2.1.10.1.3 Move

The **Move** element is a container for elements that describe details of the items to be moved.

Parent elements	Child elements	Data type	Number allowed
-----------------	----------------	-----------	----------------

<MoveItems> (request only)	<SrcMsgId/> <SrcFldId/> <DstFldId/>	Container	1...N (required)
-------------------------------	---	------------------	------------------

The **Move** element's child elements specify the item to be moved, the folder it's currently located in, and the folder it will be moved to.

2.2.1.10.1.4 SrcFldId

The **SrcFldId** element specifies the server ID of the source folder (that is, the folder that contains the items to be moved).

Parent elements	Child elements	Data type	Number allowed
<Move>	None	String (Up to 64 characters)	1 (required)

The server ID of the source folder is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command.

2.2.1.10.2 Response

The following code shows the XSD for the **MoveItems** command response.

```
<?xml version="1.0" ?>

<xs:schema xmlns:tns="Move:" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="Move:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="MoveItems">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Response">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="SrcMsgId"
type="xs:string" />
              <xs:element name="Status"
type="xs:unsignedByte" />
              <xs:element minOccurs="0"
name="DstMsgId" type="xs:string" />
            

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.10.2.1 DstMsgId

The **DstMsgId** element specifies the new server ID of the item after the item is moved to the destination folder.

Parent elements	Child elements	Data type	Number allowed
<Response> (response only)	None	String (Up to 64 characters)	1 (required)

2.2.1.10.2.2 MoveItems

The **MoveItems** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **MoveItems** command.

Parent elements	Child elements	Data type	Number allowed
None	<Move> (request only) <Response> (response only)	Container	1 (required)

2.2.1.10.2.3 Response

The **Response** element serves as a container for elements that describe the moved items.

Parent elements	Child elements	Data type	Number allowed
<MoveItems> (response only)	<SrcMsgId/> <Status/> <DstMsgId/>	Container	1 (required)

2.2.1.10.2.4 SrcMsgId

The **SrcMsgId** element specifies the server ID of the item to be moved.

Parent elements	Child elements	Data type	Number allowed
<Move> (request only) <Response> (response only)	None	String (Up to 64 characters)	1 (required)

2.2.1.10.2.5 Status

The **Status** element indicates the success or failure of an item moved. If the command failed, **Status** contains a code indicating the type of failure.

Parent elements	Child elements	Data type	Number allowed
<Response> (response only)	None	Integer	1 (required)

The following table shows valid values for the element.

Value	Meaning
1	Invalid source collection ID.
2	Invalid destination collection ID.
3	Success.
4	Source and destination collection IDs are the same.
5	A failure occurred during the MoveItem operation.
6	An item with that name already exists at the destination.
7	Source or destination item was locked.

2.2.1.11 Ping

The **Ping** command is used to request that the server monitor specified folders for changes that would require the client to resynchronize.

The body of the request contains a list of folders on the server about which the client is requesting notifications and an interval of time that specifies how long the server **SHOULD** wait before responding.

The server does not immediately issue a response to the client's **Ping** request. Instead, the server waits until one of two events occur: either the time-out that is specified by the client elapses, or changes occur in one of the folders that the client specifies. The response that the server issues indicates which of these events has happened so that the client can react appropriately.

In the case of no changes on the server, the client reissues the **Ping** request. In the case of changes, the response indicates in which folders those changes occurred so that the client can resynchronize these folders.

Note that if no changes occur in any of the folders that are specified by the client for a significant length of time, the client runs in a loop in which it issues a **Ping** request, receives a response indicating that there are no changes, and then reissues the **Ping** request. This loop is called the heartbeat. The length of time that the server **SHOULD** wait before issuing a response is called the heartbeat interval.

To reduce the amount of data sent in a **Ping** command request, the server caches the heartbeat interval and folder list. The client can omit the heartbeat interval, the folder list, or both from subsequent **Ping** requests if those parameters have not changed from the previous **Ping** request. If neither the heartbeat interval nor the folder list has changed, the client can issue a **Ping** request that does not contain an **XML** body. If the **Ping** element is specified in an **XML** request body, either the **HeartbeatInterval** element or the **Folders** element or both **MUST** be specified.

2.2.1.11.1 Request

A **Ping** command can be sent with no body, in which case the cached version is used. This XSD is applied only to requests with a body.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="Ping:" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="Ping:" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Ping">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1"
name="HeartbeatInterval" type="xs:unsignedShort" />
        <xs:element minOccurs="0" maxOccurs="1"
name="Folders">
```

```

        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="1"
maxOccurs="unbounded" name="Folder">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Id"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                                <xs:element
name="Class" type="xs:string" minOccurs="1" maxOccurs="1" />
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>

```

2.2.1.11.1.1 Class

The **Class** element specifies the content **class** of the folder to be monitored. The possible content classes are **Email**, **Calendar**, **Contacts**, and **Tasks**.

Parent elements	Child elements	Data type	Number allowed
<Folder> (request only)	None	String	1 (required)

2.2.1.11.1.2 Folder

The **Folder** element contains the **Id** and **Class** elements in the **Ping** command request, which identifies the folder and folder type to be monitored by the client. The **Folder** element is also returned by the server with the **Status** element, where the element identifies the folder that is being described by the returned status code.

Parent elements	Child elements	Data type	Number allowed
<Folders>	<Id> (request only) <Class> (request only) None (response only)	Container (request only) String (response only)	1...N (optional)

2.2.1.11.1.3 Folders

The **Folders** element serves as a container for the **Folder** element.

Parent elements	Child elements	Data type	Number allowed
<Ping>	<Folder>	Container	0...1 (optional)

2.2.1.11.1.4 HeartbeatInterval

The **HeartbeatInterval** element specifies the length of time, in seconds, that the server SHOULD wait before notifying the client of changes in a folder on the server. The **HeartbeatInterval** element is also returned by the server with a status code of 5 and specifies either the minimum or maximum interval that is allowed when the client has requested a heartbeat interval that is outside the acceptable range.

Parent elements	Child elements	Data type	Number allowed
<Ping>	None	Integer	Request- 1 (Required in first request only) Response- 0...1 (optional)

The **HeartbeatInterval** element is only required in the first **Ping** command request from a device by a given user. The server then caches the heartbeat interval value so that in later requests the **HeartbeatInterval** element is necessary only if the client is changing the interval.

2.2.1.11.1.5 Id

The **Id** element specifies the server ID of the folder to be monitored.

Parent elements	Child elements	Data type	Number allowed
<Folder> (request	None	String (Up to 64	1 (required)

only)		characters)	
-------	--	-------------	--

The server ID of the folder is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command.

2.2.1.11.1.6 Ping

The **Ping** element is the top-level element in the XML document. It identifies the body of the **HTTP POST** as containing a **Ping** command.

Parent elements	Child elements	Data type	Number allowed
None	<Folders> <HeartbeatInterval> <MaxFolders> (response only) <Status> (response only)	Container	1 (required)

The **Ping** element can also include one or more explicit namespace attributes.

2.2.1.11.2 Response

The following code shows the XSD for the **Ping** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="Ping:" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="Ping:" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Ping">
    <xs:complexType>
      <xs:choice>
        <xs:element minOccurs="1" name="Status"
type="xs:unsignedByte" />
        <xs:element minOccurs="0" maxOccurs="1"
name="Folders">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="1"
maxOccurs="unbounded" name="Folder" type="xs:string" />

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>
    <xs:element minOccurs="0" name="MaxFolders"
type="xs:integer" />
    <xs:element minOccurs="0" name="HeartbeatInterval"
type="xs:integer" />
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.11.2.1 Folder

The **Folder** element contains the **Id** and **Class** elements in the **Ping** command request, which identifies the folder and folder type to be monitored by the client. The **Folder** element is also returned by the server with the **Status** element, where the element identifies the folder that is being described by the returned status code.

Parent elements	Child elements	Data type	Number allowed
<Folders>	<Id> (request only) <Class> (request only) None (response only)	Container (request only) String (response only)	1...N (optional)

2.2.1.11.2.2 Folders

The **Folders** element serves as a container for the **Folder** element.

Parent elements	Child elements	Data type	Number allowed
<Ping>	<Folder>	Container	0...1 (optional)

2.2.1.11.2.3 HeartbeatInterval

The **HeartbeatInterval** element specifies the length of time, in seconds, that the server SHOULD wait before notifying the client of changes in a folder on the server. The **HeartbeatInterval** element is also returned by the server with a status code of 5 and specifies

either the minimum or maximum interval that is allowed when the client has requested a heartbeat interval that is outside the acceptable range.

Parent elements	Child elements	Data type	Number allowed
<Ping>	None	Integer	Request- 1 (Required in first request only) Response- 0...1 (optional)

The **HeartbeatInterval** element is only required in the first **Ping** command request from a device by a given user. The server then caches the heartbeat interval value so that in later requests the **HeartbeatInterval** element is necessary only if the client is changing the interval.

2.2.1.11.2.4 MaxFolders

The **MaxFolders** element specifies the maximum number of folders that can be monitored.

Parent elements	Child elements	Data type	Number allowed
<Ping> (response only)	None	Integer	0...1 (optional)

The **MaxFolders** element is returned in a response with a status code of 6.

2.2.1.11.2.5 Ping

The **Ping** element is the top-level element in the **XML** document. It identifies the body of the **HTTP POST** as containing a **Ping** command.

Parent elements	Child elements	Data type	Number allowed
None	<Folders> <HeartbeatInterval> <MaxFolders> (response only) <Status> (response only)	Container	1 (required)

The **Ping** element can also include one or more explicit namespace attributes.

2.2.1.11.2.6 Status

The **Status** element indicates the success or failure of the **Ping** command request. If the command failed, the **Status** element contains a code that indicates the type of failure. Certain status codes have additional information that is included in the response.

Parent elements	Child elements	Data type	Number allowed
<Ping> (response only)	None	Integer	1 (required)

The following table lists valid values for the element.

Value	Meaning
1	The heartbeat interval expired before any changes occurred in the folders that are being monitored. The client SHOULD reissue the Ping command request.
2	Changes occurred in at least one of the folders that were being monitored. The response includes the folders in which these changes have occurred.
3	The client Ping command request did not specify all the necessary parameters. The client MUST issue a Ping request that includes both the heartbeat interval and the folder list.
4	There is a general error in the Ping request that was issued by the client, which can be caused by poorly formatted WBXML .
5	The heartbeat interval that was specified by the client is outside the range that was set by the server administrator. If the specified interval was too great, the returned interval is the maximum allowed value. If the specified interval was too low, the returned interval is the minimum allowed value.
6	The Ping command request specified more folders to monitor for changes than is allowed by the limit that was configured by the server administrator. The response specifies the limit in the MaxFolders element.
7	The client specified a folder that has been moved or deleted. The client SHOULD issue a FolderSync request.
8	An error has occurred. The client SHOULD reissue the Ping request.

2.2.1.12 Provision

The **Provision** command enables client devices to request from the server the security policy settings that the administrator sets, such as the minimum personal identification number (PIN)

password length requirement. The **Provision** command is one of the first commands sent to the server by the client. The **Provision** command is specified in [MS-ASPROV].

2.2.1.13 ResolveRecipients

The **ResolveRecipients** command is used by clients to resolve a list of supplied **recipients** and, optionally, to retrieve their **S/MIME** certificates so that clients can send encrypted **S/MIME** e-mail messages.

2.2.1.13.1 Request

The following code shows the XSD for the **ResolveRecipients** command request.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="ResolveRecipients:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="ResolveRecipients:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="ResolveRecipients">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1000" name="To"
type="xs:string" />
      <xs:element name="Options" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1"
name="CertificateRetrieval" type="xs:integer" />
            <xs:element minOccurs="0" maxOccurs="1"
name="MaxCertificates" type="xs:integer">
            <xs:element minOccurs="0" maxOccurs="1"
name="MaxAmbiguousRecipients" type="xs:integer">
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

</xs:schema>

2.2.1.13.1.1 CertificateRetrieval

The **CertificateRetrieval** element specifies whether S/MIME certificates SHOULD be returned by the server for each resolved **recipient**.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	1...1

The following table shows valid values for the **CertificateRetrieval**.

Value	Meaning
1	Do not retrieve certificates for the recipient (default).
2	Retrieve x509 certificate(s) for each resolved recipient.
3	Retrieve the certificate for each resolved recipient.

2.2.1.13.1.2 MaxAmbiguousRecipients

The **MaxAmbiguousRecipients** element limits the number of suggestions that are returned for each ambiguous **recipient** node in the response.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	0...1

The value of **MaxAmbiguousRecipients** is limited to a range of 0–9999. Each ambiguous recipient node receives only this many suggestions and no more. The recipient count, returned in the **RecipientCount** element, can be used by the client to determine the total number of suggestions available for that recipient.

2.2.1.13.1.3 Options

The **Options** element contains the options for resolving the list of **recipients**.

Parent elements	Child elements	Data type	Number allowed
<ResolveRecipients>	<CertificateRetrieval>, <MaxCertificates>,	Container	0...1 (optional)

(request only)	<MaxAmbiguousRecipients>		
----------------	--------------------------	--	--

2.2.1.13.1.4 MaxCertificates

The **MaxCertificates** element limits the total number of certificates that is returned by the server.

Parent elements	Child elements	Data type	Number allowed
<Options>	None	Integer	0...1

The value of **MaxCertificates** is limited to a range of 0–9999. This limit ensures that no individual **recipient** receives an incomplete set of certificates. For example, if the limit is reached while enumerating certificates for an **address list**, that address list won't get back any certificates and an appropriate certificate status code is returned. The client can then use the certificate count returned to determine the number of certificates that are available for that recipient node.

2.2.1.13.1.5 ResolveRecipients

The **ResolveRecipients** element is the top-level element in the **XML** document. It identifies the body of the **HTTP** Post as containing a **ResolveRecipients** command.

Parent elements	Child elements	Data type	Number allowed
None	<To>, <Options> (request only) <Status>, <Response> (response only)	Container	1 (required)

The **ResolveRecipients** element can also include one or more explicit namespace attributes.

2.2.1.13.1.6 To

The **To** element specifies one or more **recipients** to be resolved. The **To** element is also an **ambiguous name resolution (ANR)** search field.

Parent elements	Child elements	Data type	Number allowed
<ResolveRecipients> (request only) <Response>	None	String , limited to 256 characters.	0...1000 (optional)

(response only)			
-----------------	--	--	--

The server attempts to match the **To** value specified in the request to common directory service user attributes, and then return the matches. The **To** element(s) that are returned in the response corresponds directly to the **To** element(s) that are specified in the request.<9>

2.2.1.13.2 Response

The following code shows the XSD for the **ResolveRecipients** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="ResolveRecipients:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="ResolveRecipients:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="ResolveRecipients">
  <xs:complexType>
    <xs:choice>
      <xs:element minOccurs="1" name="Status" type="xs:unsignedByte"
/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
name="Response">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" name="To"
type="xs:string"/>
            <xs:element minOccurs="1" maxOccurs="1" name="Status"
type="xs:string"/>
            <xs:element minOccurs="1" maxOccurs="1"
name="RecipientCount" type="xs:integer"/>
            <xs:element minOccurs="1" maxOccurs="unbounded"
name="Recipient" type="xs:string">
              <xs:complexType>
                <xs:sequence>
                  <xs:element minOccurs="1" maxOccurs="1"
name="Type" type="xs:unsignedbyte"/>
                  <xs:element minOccurs="1" maxOccurs="1"
name="DisplayName" type="xs:string"/>

```

```

        <xs:element minOccurs="1" maxOccurs="1"
name="EmailAddress" type="xs:string"/>
        <xs:element minOccurs="0" maxOccurs="unbounded"
name="Certificates">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="1" maxOccurs="1"
name="Status" type="xs:unsignedbyte"/>
                    <xs:element minOccurs="1" maxOccurs="1"
name="CertificateCount" type="xs:integer"/>
                    <xs:element minOccurs="1" maxOccurs="1"
name="RecipientCount" type="xs:integer"/>
                    <xs:element minOccurs="0" maxOccurs="1"
name="MiniCertificate" type="xs:string"/>
                </xs:complexType>
            </xs:sequence>
        </xs:element>
    </xs:complexType>
</xs:sequence>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.13.2.1 Certificate

The **Certificate** element contains the Base64-encoded x509 certificate **BLOB**.

Parent elements	Child elements	Data type	Number allowed
<Certificates>	None	String (Base64-encoded)	0...N

This element is returned by the server only if the client specified a value of 2 in the **CertificateRetrieval** element in the request.

2.2.1.13.2.2 CertificateCount

The **CertificateCount** element specifies the number of valid certificates that were found for the **recipient**.

Parent elements	Child elements	Data type	Number allowed
<Certificates> (response only)	None	Integer	1...N

If a status code of 8 is returned with the **Certificates** element, the **CertificateCount** element specifies the number of recipient certificates that was not returned.

2.2.1.13.2.3 Certificates

The **Certificates** element contains information about the certificates for a **recipient**.

Parent elements	Child elements	Data type	Number allowed
<Recipient> (response only)	<Status>, <CertificateCount>, <RecipientCount>, <MiniCertificate>, <Certificate>	Container	0...N

2.2.1.13.2.4 DisplayName

The **DisplayName** element contains the display name of the **recipient**.

Parent elements	Child elements	Data type	Number allowed
<Recipient> (response only)	None	String	1...N

2.2.1.13.2.5 EmailAddress

The **EmailAddress** element contains the e-mail address, in **SMTP** format, of the **recipient**.

Parent elements	Child elements	Data type	Number allowed
<Recipient>	None	String	1...N

(response only)			
-----------------	--	--	--

2.2.1.13.2.6 MiniCertificate

The **MiniCertificate** element contains the Base64-encoded mini-certificate **BLOB**.

Parent elements	Child elements	Data type	Number allowed
<Certificates>	None	String (Base64-encoded)	0...N

This element is returned only if the client specified a value of 3 in the **CertificateRetrieval** element in the request.

2.2.1.13.2.7 Recipient

The **Recipient** element represents a single **recipient** that has been resolved.

Parent elements	Child elements	Data type	Number allowed
<Response>	<Type>, <DisplayName>, <EmailAddress>, <Certificates>	Container	0...N

One or more **Recipient** elements are returned to the client in a **Response** element by the server if the **To** element specified in the request was either resolved to a **distribution list** or found to be ambiguous. The status code returned in the **Response** element can be used to determine if the **recipient** was found to be ambiguous. The recipient would be a suggested match if the recipient specified in the request was found to be ambiguous.

A **Certificates** element is returned as a child of **Recipient** if the client requested certificates to be returned in the response.

2.2.1.13.2.8 RecipientCount

The **RecipientCount** element specifies the number of **recipients** that are returned in the **ResolveRecipients** command response or the count of members belonging to a **distribution list**.

Parent elements	Child elements	Data type	Number allowed
<Response>, <Certificates>	None	Integer	1...1 (required)

As a child of the **Response** element, the recipient count specifies the number of recipients that are returned in the **ResolveRecipients** command response. As a child of the **Certificates** element, the recipient count specifies the number of members belonging to a distribution list.

When returned in the **Certificates** element, the recipient count can be used to determine whether all recipients belonging to a distribution list have valid certificates by comparing values of the **CertificateCount** and **RecipientCount** elements.

The **ResolveRecipients** element is the top-level element in the **XML** document. It identifies the body of the **HTTP** Post as containing a **ResolveRecipients** command.

Parent elements	Child elements	Data type	Number allowed
None	<To>, <Options> (request only) <Status>, <Response> (response only)	Container	1 (required)

The **ResolveRecipients** element can also include one or more explicit namespace attributes.

2.2.1.13.2.9 Response

The **Response** element contains information as to whether the **recipient** was resolved; if the recipient was resolved, the element also contains the type of recipient, the e-mail address that the recipient resolved to, and, optionally, the **S/MIME** certificate for the recipient.

Parent elements	Child elements	Data type	Number allowed
<ResolveRecipients>	<To>, <Status>, <RecipientCount>, <Recipient>	Container	1...N

2.2.1.13.2.10 Status

The **Status** element provides the status of the response. The meaning of the returned status code depends on whether the **Status** element was returned as a child of the **ResolveRecipients** element, the **Response** element, or the **Certificates** element.

Parent elements	Child elements	Data type	Number allowed
<ResolveRecipients>, <Response>, <Certificates>	None	Integer	1...N

(response only)			
-----------------	--	--	--

The following table shows valid values for the **Status** element when it is returned as a child of the **ResolveRecipients** element.

Value	Meaning
1	Success.
5	Protocol error. Either an invalid parameter was specified or the range exceeded limits.
6	An error occurred on the serverThe client SHOULD retry the request.

The following table shows valid values for the **Status** element when it is returned as a child of the **Response** element.

Value	Meaning
1	The recipient was resolved successfully. For more details about the Recipient element, see section 2.2.1.13.2.7.
2	The recipient was found to be ambiguous. The returned list of recipients are suggestions. No certificate nodes were returned.
3	The recipient was found to be ambiguous. The returned list is a partial list of suggestions. The total count of recipients can be obtained from the RecipientCount element. No certificate nodes were returned.
4	The recipient did not resolve to any contact or GAL entry.

The following table shows valid values for the **Status** element when it is returned as a child of the **Certificates** element.

Value	Meaning
1	One or more certificates were successfully returned.
7	The recipient does not have a valid S/MIME certificate. No certificates were returned.

8	The global certificate limit was reached and the recipient's certificate could not be returned. The count of certificates not returned can be obtained from the CertificateCount element.
9	Expansion or enumeration of certificates for the members of the distribution list failed. The distribution list can have membership over the limit that is set on the server.

If a status code is not returned for a command, the client SHOULD assume success.

2.2.1.13.2.11 To

The **To** element specifies a **recipient** to be resolved and is an **ANR** search field.

Parent elements	Child elements	Data type	Number allowed
<ResolveRecipients> (request only) <Response> (response only)	None	String , limited to 256 characters.	1...1 (required)

The server attempts to match the **To** value specified in a request to common directory service user attributes, and then return the matches. The **To** element(s) that are returned in the response correspond directly to the **To** element(s) that are specified in the request.<10>

2.2.1.13.2.12 Type

The **Type** element indicates the type of **recipient**, either a **contact** entry (2) or a GAL entry (1).

Parent elements	Child elements	Data type	Number allowed
<ResolveRecipients> (request only) <Response> (response only)	None	String , limited to 256 characters.	1...1 (required)

2.2.1.14 Search

The **Search** command is used to find entries in an **address book** or **mailbox**.

The Accept-Language header in a **Search** command request is currently used to define the **locale** of the client so that the search is relevant. If the accept language is not specified, the search is conducted by using the server language.

Searching the Global Address List (GAL)

The **Search** command is used to find **contacts** and **recipients** in the **GAL**, and to retrieve information about them. When a search query matches more than one GAL entry, the **Search** command can return as many entries as requested, up to a total of 100 entries by default.

For each GAL entry that is found, the **Search** command returns all the non-empty properties that are indexed by the online **ANR** in the global catalog server—for example, e-mail alias, display name, first and last names, company name, and so on.

The client can optionally specify the maximum number of entries to retrieve in the **Search** command request by specifying the range. The server **MUST** return entries up to the number that is requested, and **MUST** also indicate the total number of entries that are found.

The text query string that is provided to the **Search** command is used in a prefix-string match. For example, if the device performs a **Search** with a **Query** element value of "Michael A.," the command returns the entries that contain the search string in any text field, such as "Michael Alexander", "Michael Allen". Because the **Search** command matches the **Query** element value against all ANR-indexed GAL text properties, you can also search by e-mail address, company name, and so on.

The ANR system indexes the following properties:

- Display name
- Phone
- Office
- Title
- Company
- Alias
- FirstName
- LastName
- HomePhone
- MobilePhone
- EmailAddress

The list above contains the currently supported properties.

The **Search** command results are sorted by the server according to their ordering in the GAL (that is, by the display name property). Because of how the search results are sorted, the device could have to sort the results to achieve a meaningful list. For example, a search for "123" might return all GAL entries that have mailing addresses or e-mail addresses that begin with 123.

The **Range** option is a zero-based index specifier in the form of "0-n." For more details about the meaning of the **Range** values, see section 2.2.1.14.1.4. The **Search** command does not return more than 1,000 entries.

Searching Outside the GAL <11>

Typical Usage

Essentially, search involves the following three phases:

1. The client issues a request for windowed search results.
2. The client uses subsequent requests to retrieve more results by incrementing the range.
3. Any actions on the search results are carried out by using other protocol commands, such as **ItemOperations**, **Fetch**, **SmartReply**, or **SmartForward**.

The following figure shows the typical usage of the **Search** command to retrieve successive result sets from the server and then perform some action based on those results (for example, retrieve the full message body for an e-mail search result).

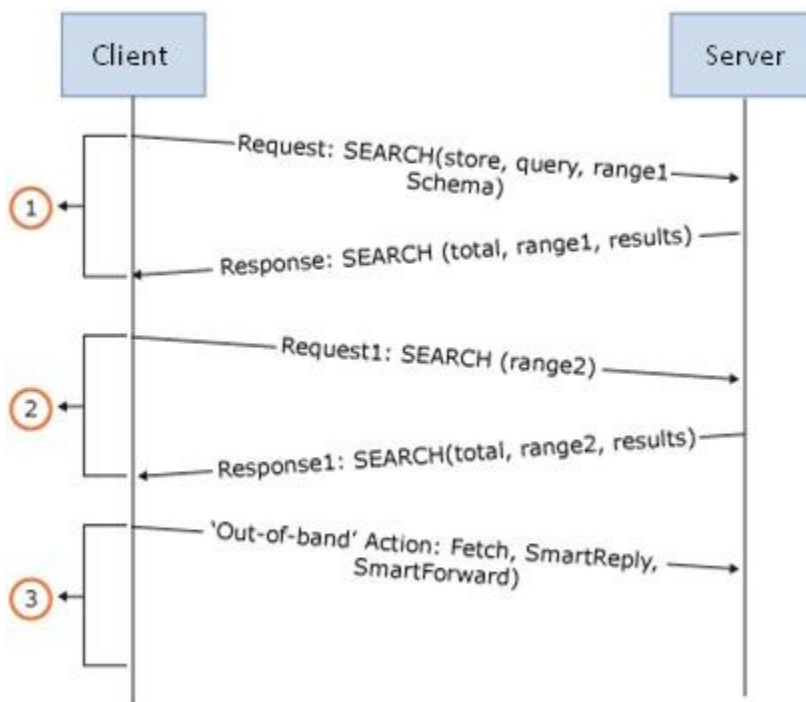


Figure 2: Search command process

2.2.1.14.1 Request

The following code shows the XSD for the **Search** command request.

```
<?xml version="1.0"?>
<xs:schema id="Search" targetNamespace="Search:"
xmlns:calendar="Calendar:" xmlns:contacts2="Contacts2:"
xmlns:contacts="Contacts:" xmlns:email="Email:" xmlns:msdns="Search:"
xmlns="Search:" xmlns:airSync="AirSync:"
```

```

xmlns:airsynibase="AirSyncBase:"
xmlns:documentLibrary="DocumentLibrary:"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="qualified" elementFormDefault="qualified">

  <xs:import namespace="DocumentLibrary:"/>
  <xs:import namespace="AirSync:"/>
  <xs:import namespace="AirSyncBase:"/>
  <xs:import namespace="Email:"/>
  <xs:element name="LongId">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="256"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:complexType name="EmptyTag" />
  <xs:complexType name="queryType" mixed="true">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="And" type="queryType"/>
        <xs:element name="Or" type="queryType"/>
        <xs:element name="FreeText" type="xs:string" />
        <xs:element ref="airSync:Class" />
        <xs:element ref="airSync:CollectionId" />
        <xs:element name="EqualTo" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="documentLibrary:LinkId" minOccurs="1"
maxOccurs="1"/>
              <xs:element minOccurs="1" name="Value">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:maxLength value="1024"/>

```

```

        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GreaterThan">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="email:DateReceived" minOccurs="1"
maxOccurs="1"/>
      <xs:element minOccurs="1" name="Value">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="1024"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="LessThan">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="email:DateReceived" minOccurs="1"
maxOccurs="1"/>
      <xs:element minOccurs="1" name="Value">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="1024"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:choice>
</xs:sequence>
</xs:complexType>
<xs:element name="Search">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="Store">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="Name">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:minLength value="1"/>
                                    <xs:maxLength value="256"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>
                        <xs:element name="Query" minOccurs="0" type="queryType"
/>
                            <xs:element name="Options" minOccurs="0" maxOccurs="1">
                                <!-- Must differentiate between document library and
Mailbox options...!-->
                                <xs:complexType>
                                    <xs:choice maxOccurs="unbounded">
                                        <xs:element ref="airSync:MIMESupport" minOccurs="0"
maxOccurs="1" />
                                        <xs:element ref="airSyncbase:BodyPreference" minOccurs="0"
maxOccurs="unbounded" >
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element ref="airSyncbase:Type" />

```

```

        <xs:element ref="airsynibase:TruncationSize"
/>

        <xs:element ref="airsynibase:AllOrNone" />
    </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="Schema">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="airSync:Class" />
        </xs:choice>
    </xs:complexType>
</xs:element>
<xs:element name="Range">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{1,3}-[0-9]{1,3}" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="UserName">

    <xs:simpleType>

        <xs:restriction base="xs:string">

            <xs:maxLength value="100" />

        </xs:restriction>

    </xs:simpleType>

    </xs:element>

    <xs:element
name="Password">

```



```

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="100" />

</xs:restriction>

</xs:simpleType>

</xs:element>

    <xs:element name="DeepTraversal" type="EmptyTag" />
    <xs:element name="RebuildResults" type="EmptyTag" />
  </xs:choice>
</xs:complexType>
</xs:element>
</xs:all>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.14.1.1 Name

The **Name** element in the **Search** command request specifies which store to search.

Parent elements	Child elements	Data type	Number allowed
<Store>	None	String	1...1 (required)

The following values are also valid:

- **Mailbox**—The client specifies "Mailbox" when it intends to search the **message database**.
- **Document Library**—The client specifies "DocumentLibrary" when it intends to search a Windows SharePoint Services or **UNC** library.

- **GAL** – The client specifies "GAL" when it intends to search the **Global Address List**.

2.2.1.14.1.2 Query

The **Search** command request **XML Query** element specifies the keywords to use for matching the entries in the store that is being searched.

Parent elements	Child elements	Data type	Number allowed
<Query>	<And> <Or> <FreeText> <Class> <CollectionId> <EqualTo> <GreaterThan> <LessThan>	String	1...1 (Required, request only)

The value of the **Query** element is used as a prefix-string matching pattern, and returns entries that match the beginning of the string. For example, searching for "John" would match "John Frum" or "Barry Johnson", but would not match "James Littlejohn".

All nonempty ANR-indexed text properties in the **GAL** are compared with the **Query** element value. Search comparisons are performed by using case-insensitive matching.

For a Windows SharePoint Services document library search, the [MS-ASCMD] protocol supports queries of the following form: *LinkId* == *value*, where *value* specifies the URL of the item or folder and *LinkId* indicates that the value SHOULD be compared to the link ID property.

For **mailbox** search, the query syntax is as follows:

- Folders can be specified in the following ways:
 - Specified ID
 - Specified folder and subfolders
 - All e-mail folders, including Draft, **Inbox** and subfolders, **Outbox**, and Sent Items
- The basic keyword query can be composed of the following:
 - The basic operators **And** and **Or** <12>
 - Date/Time filter specified by using the **GreaterThan** and **LessThan** operators
 - **FreeText** elements that contain keywords

The basic keyword query is executed against all indexed properties.<13>

2.2.1.14.1.3 Options

The **Search** command request **XML Options** element is a container for search options.

Parent elements	Child elements	Data type	Number allowed
<Options>	<airsynbase:BodyPreference> (request only) <airsync:MIMESupport> (request only) <Schema> <Range> <DeepTraversal> <RebuildResults> <UserName> <Password>	Container	0...1 (Optional, Request only)

The **UserName** and **Password** can only be sent in the request after receiving a status 14. The server requires these credentials to access the requested resources. The device **MUST** only send these over a secure or trusted connection, and only in response to a status 14. **UserName** and **Password** are defined as strings consisting of at most 100 characters.

The supported options vary according to the store that is being searched. The following table lists the valid options for each store.

Options	Store
Range UserName Password	GAL
Range Schema DeepTraversal RebuildResults Airynbase:BodyPreference	Mailbox

UserName	
Password	
Range	Document Library
UserName	
Password	

2.2.1.14.1.4 Range

The **Search** command XML **Range** element is used in both the request and response XML documents. In the request XML, the **Range** element specifies the maximum number of matching entries to return. In the response XML, the **Range** element specifies the number of matching entries that are being returned.

Parent elements	Child elements	Data type	Number allowed
<Options> (request) <Store> (response)	None	Zero-based range in the form m-n	0...1 (optional)

The **Range** element value specifies a number of entries, but indicates different things depending on whether the element is in the request or the response XML.

The format of the **Range** element value is in the form of a zero-based index specifier, formed with a zero, a hyphen, and another numeric value: "m-n." The *m* indicates the lowest index of a zero-based array that would hold the items. The *n* indicates the highest index of a zero-based array that would hold the items. For example, a **Range** element value of 0–9 indicates 10 items, and 0–10 indicates 11 items. A **Range** element value of 0–0 indicates 1 item.

If the request does not include a **Range** element, the default range of 0–99 is assumed.

In the request XML, the **Range** element value specifies the maximum number of entries to be returned to the client.

In the response XML, the **Range** element value specifies the actual number of entries that are returned in the response. The **Total** element in the response XML indicates the total number of entries that matched the **Query** element value.

Search results are stored in a **search folder** on the server. This way, when a client comes back with the same query but a new row range, rows are pulled from the result set that is currently stored in the **search folder**. The entire result set does not have to be rebuilt.

2.2.1.14.1.5 Properties

In a **Search** request, the **Properties** element encapsulates the schema in which the client wants to have search results formatted. In a **Search** response, the **Properties** element encapsulates the properties for each search result.

The **Properties** element is accepted but currently ignored by the server.

Parent elements	Child elements	Data type	Number allowed
<Schema> (request only) <Result> (response only)	Named properties <airsynbase:Body> (response only)	Container	0...1 (Optional, request only) 1...1 (Required, response only)

The **Search** command response **Properties** element is a container for properties that apply to an individual entry that matches the **Query** element search string. For example, the **Properties** element contains an element for each nonempty, text-valued **GAL** property that is attached to the matching GAL entry. Only those properties that are attached to the specific GAL entry are returned; therefore different sets of properties can be returned in the response **XML** for different matching GAL entries.

Each element in the **Properties** container is scoped to the appropriate namespace that is specified in the top-level **Search** element.

2.2.1.14.1.6 MIMESupport

The **MIMESupport** element is included in the **Options** element of a client **Search** command request to enable **MIME** support for e-mail items that are sent from the server to the client.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	0...1 (optional)

The following table lists the valid values for the element.

Value	Meaning
0	Never send MIME data.
1	Send MIME data for S/MIME messages only. Send regular body for all other messages.
2	Send MIME data for all messages. This flag could be used by clients to build a more

rich and complete Inbox solution.

The **Search** response can include the **S/MIME BLOB** of a signed/encrypted message.

The **Search** request can include the following in the **Options** element:

- The **MIMESupport** element to tell the server to return MIME for S/MIME-only/All/None messages.
- The **BodyPreference** element with its child element, **Type**, containing a value of 4 to inform the server that the device can read the MIME BLOB.

The response from the server **MUST** include the **Body** element, which is a child of the **Properties** element. The **Body** element is a complex element and **MUST** contain the following child nodes in an S/MIME search response:

- The **Type** element with a value of 4 to inform the device that the data is a MIME BLOB.
- The **EstimatedDataSize** element to specify the rough total size of the data.
- The **Truncated** element to indicate whether the MIME BLOB is truncated.
- The **Data** element that contains the full MIME BLOB.

For more details about the **Body** element or the **BodyPreference** element, see [MS-ASAIRS] sections 2.2.3.3 or 2.2.3.4, respectively.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<Search xmlns="Search:" xmlns:A="AirSyncBase:" xmlns:B="AirSync:"
xmlns:C="Email:">
  <Store>
    <Name>Mailbox</Name>
    <Query>
      <And>
        <B:Class>Email</B:Class>
        <FreeText>text</FreeText>
      </And>
    </Query>
    <Options>
      <RebuildResults/>
      <DeepTraversal/>
      <Range>0-999</Range>
      <A:BodyPreference>
```

```

        <A:Type>4</A:Type>
        <A:TruncationSize>1024</A:TruncationSize>
    </A:BodyPreference>
    <B:MIMESupport>2</B:MIMESupport>
</Options>
</Store>
</Search>

```

Response

```

<Search xmlns:A0="AirSync:" xmlns:A1="POOMCONTACTS:"
xmlns:A2="POOMMAIL:" xmlns:A3="AirNotify:" xmlns:A4="POOMCAL:"
xmlns:A5="Move:" xmlns:A6="GetItemEstimate:"
xmlns:A7="FolderHierarchy:" xmlns:A8="MeetingResponse:"
xmlns:A9="POOMTASKS:" xmlns:A10="ResolveRecipients:"
xmlns:A11="ValidateCert:" xmlns:A12="POOMCONTACTS2:" xmlns:A13="Ping:"
xmlns:A14="Provision:" xmlns:A16="Gal:" xmlns:A17="AirSyncBase:"
xmlns:A18="Settings:" xmlns:A19="DocumentLibrary:"
xmlns:A20="ItemOperations:" xmlns="Search:">
    <Status>1</Status>
    <Response>
        <Store>
            <Status>1</Status>
            <Result>
                <A0:Class>Email</A0:Class>

<LongId>RgAAAAAaty%2f%2b4QxHTJOZnIR0P9qkBwA6pk60fqkEQbWH4Wm%2bnjh7AJKAU
Qo6AAA6pk60fqkEQbWH4Wm%2bnjh7AJKAURrEAAAJ</LongId>
                <A0:CollectionId>6</A0:CollectionId>
                <Properties>
                    <A2:To>"NSyncUser1" &lt;NSyncUser1@contoso.com></A2:To>
                    <A2:From>"NSyncUser1"
&lt;NSyncUser1@contoso.com></A2:From>
                    <A2:Subject>Subject</A2:Subject>
                    <A2:DateReceived>2007-04-
02T19:20:32.000Z</A2:DateReceived>
                    <A2:DisplayTo>NSyncUser1</A2:DisplayTo>

```

```

    <A2:Read>1</A2:Read>
    <A17:Body>
      <A17:Type>4</A17:Type>
      <A17:EstimatedDataSize>2288</A17:EstimatedDataSize>
      <A17:Truncated>1</A17:Truncated>
      <A17:Data>Received: from 157.55.97.120
([157.55.97.120]) by contoso.com ([157.55.97.121]) with Microsoft
Exchange Server HTTP-DAV ; Mon, 2 Apr 2007 19:20:32 +0000 From:
NSyncUser1 <NSyncUser1@contoso.com> To: NSyncUser1
<NSyncUser1@contoso.com> Content-Class: urn:content-classes:message
Date: Mon, 27 Apr 1998 13:05:29 -0700 Subject: Subject Thread-Topic:
Topic Message-ID:
<3AA64EB47EA90441B587E169BE9E387B9280511AC4@contoso.com> Accept-
Language: en-US X-MS-Has-Attach: X-MS-TNEF-Correlator: Content-Type:
text/plain; charset="iso-8859-1" Content-Transfer-Encoding: quoted-
printable MIME-Version: 1.0
Body1234567890123456789012345678901234567890123456789012345678901234567
8901=
23456789012345678901234567890123456789012345678901234567890123456789012
3456=
78901234567890123456789012345678901234567890123456789012345678901234567
8901=
2345678901234567890123456789012345678901234567890123456789</A
17:Data>

    </A17:Body>
    <A2:MessageClass>IPM.Note</A2:MessageClass>
    <A2:InternetCPID>28591</A2:InternetCPID>
    <A2:Flag/>
    <A2:ContentClass>urn:content-
classes:message</A2:ContentClass>
    <A17:NativeBodyType>1</A17:NativeBodyType>
  </Properties>
</Result>
<Range>0-0</Range>
<Total>1</Total>
</Store>

```


</Response>
</Search>

2.2.1.14.1.7 Search

The **Search** element is the top-level element in the **XML** document for the **Search** command. The element identifies the body of the **HTTP** Post as containing a **Search** command.

Parent elements	Child elements	Data type	Number allowed
None	<Store> (request) <Status>, <Response> (response)	Container	1...1 (required)

2.2.1.14.1.8 Store

In the **Search** command request **XML**, the **Store** element is a container for elements that specify the location, string, and options for the search. In the **Search** command response **XML**, the **Store** element contains the **Status**, **Result**, **Range**, and **Total** elements that contain the returned **mailbox** entries.

Parent elements	Child elements	Data type	Number allowed
<Search> (request) <Response> (response)	<Name>, <Query>, <Options> (request) <Status>, <Result>, <Range>, <Total> (response)	Container	1...1 (required)

2.2.1.14.1.9 And

The **And** element is a container that specifies items on which to perform an **AND** operation.

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or>	<And> <Or> <FreeText> <Class>	Container	0...1 (Optional, request only)

	<CollectionId> <EqualTo> <GreaterThan> <LessThan>		
--	--	--	--

The server performs an **AND** operation on the child nodes of the **And** element.<14>

2.2.1.14.1.10 Or

The **Or** element is a container that specifies items on which to perform an **OR** operation.

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or>	<And> <Or> <FreeText> <Class> <CollectionId> <EqualTo> <GreaterThan> <LessThan>	Container	0...1 (Optional, request only)

The server performs an **OR** operation on the child nodes of the **Or** element.<15>

2.2.1.14.1.11 Class

The **Class** element specifies the item **classes** that the client wants returned for a given **collection**.

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or> <Schema>	None	String	0...4

The **Search** request can specify multiple **Class** elements, each of which is observed by the server when it sends the **Search** response to the client.<16>

If one or more **Class** elements are not specified, the server will return all supported content classes.<17>

2.2.1.14.1.12 DeepTraversal

The **DeepTraversal** element indicates that the client wants the server to search all subfolders for the folders that are specified in the query.

Parent elements	Child elements	Data type	Number allowed
<Options>	None	Empty tag	0...1 (optional)

If the **DeepTraversal** element is not present, the subfolders are not searched.

2.2.1.14.1.13 EqualTo

The **EqualTo** element is a container that specifies a property and a value that are compared for equality during a search.

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or>	<LinkId> <Value>	Container	0...1 (optional, Request only)

The comparison is made between the value of the **Value** element and the link ID.<18>

2.2.1.14.1.14 GreaterThan

The **GreaterThan** element is a container that specifies a property and a value that are compared for a "greater than" condition during a search.

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or>	<DateReceived> <Value>	Container	0...1 (optional, Request only)

The **GreaterThan** element is supported only in **mailbox** searches. It is not supported for document library searches. The comparison is made between the value of the **Value** element and the date that a mailbox item was received. <19> The **DateRecieved** element MUST be present before the **Value** element.

Typically, this element is used to filter results by the date on which they were received so that the date received is greater than the specified value.

2.2.1.14.1.15 LessThan

The **LessThan** element is a container that specifies a property and a value that are compared for a "less than" condition during a search.

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or>	<DateReceived> <Value>	Container	0...1 (Optional, request only)

The **LessThan** element is supported only in **mailbox** searches. It is not supported for document library searches. The comparison is made between the value of the **Value** element and the date that a mailbox item was received. <20> The **DateReceived** element MUST be present before the **Value** element.

Typically, this element is used to filter results by the date on which they were received so that the date received is less than the specified value.

2.2.1.14.1.16 Value

The **Value** element specifies the value that is to be used in a comparison.

Parent elements	Child elements	Data type	Number allowed
<EqualTo> <GreaterThan> <LessThan>	None	String (1,024 bytes maximum length)	0...1 (Optional, request only)

The **Value** element is used in the query together with an element that specifies the name of a property. The value that is specified by the **Value** element is compared with the value of the specified property.

2.2.1.14.1.17 FreeText

The **FreeText** element specifies a string for which to search.

Parent elements	Child elements	Data type	Number allowed
<Query>	None	String	0...1 (Optional,

<And>			request only)
<Or>			

The **FreeText** element is used together with the **And** and **Or** elements to build the query.<21>

2.2.1.14.1.18 CollectionId

The **CollectionId** element specifies the folder in which to search.<22>

Parent elements	Child elements	Data type	Number allowed
<Query> <And> <Or>	None	String	0...N (optional, Request only)

Multiple folders can be specified by including multiple **CollectionId** elements.

If the **DeepTraversal** element is present, it applies to all folders under each CollectionId.

2.2.1.14.1.19 RebuildResults

The **RebuildResults** element is used within the **Search** request to force the server to rebuild the **search folder** that corresponds to a given query.

Parent elements	Child elements	Data type	Number allowed
<Options> (request)	None	Empty tag	0...1 (optional)

The search results (that is, the result set) are stored in a **search folder** on the server. This way, when a client comes back with the same query but a new row range, rows are pulled from the result set that is currently stored in the **search folder**. The entire result set does not have to be rebuilt.

The **search folder** remains unchanged until the client does one of the following to update the result set:

- Sends a **Search** request, specifying a new query. In this case, the search folder is automatically rebuilt. The **RebuildResults** node does not have to be included.
- Sends a **Search** request that includes the **RebuildResults** node. In this case, the server is forced to rebuild the search folder.

- Enough time (order of days) has passed since the folder was first accessed (see three sentences down)

If a new item is added, the item does not appear in the result set until the result set is updated. If an item is deleted, the server will filter the deleted item out of the result set.

If a long period of time has elapsed since a given query was issued, the client **SHOULD** send a new **Search** request with the given query and include the **RebuildResults** option to ensure accurate results for that query.

2.2.1.14.1.20 LinkId

The **LinkId** element specifies a **URI** that identifies a resource.

Parent elements	Child elements	Data type	Number allowed
<EqualTo>	None	URI	1...1 (required)

The link ID is a URI that is assigned by the server to certain resources, such as Windows SharePoint Services or **UNC** documents. The client **MUST** store the link ID that is associated with the items that are retrieved by using the **Search** command if it wants to act upon them later.

2.2.1.14.1.21 DateReceived

The **DateReceived** element specifies the date that a **mailbox** item was received.

Parent elements	Child elements	Data type	Number allowed
<GreaterThan> <LessThan>	None	DateTime	1...1 (required)

2.2.1.14.2 Response

2.2.1.14.2.1 LongId

The **LongId** element specifies a unique identifier that is assigned by the server to each result set that is being returned in the **Search** response.

Parent elements	Child elements	Data type	Number allowed
<Result> (response only)	None	String (up to 256 characters)	0...1 (optional)

The value of the **LongId** element can be used in the *LongId* parameter of the **ItemOperations**, **SmartReply**, **SmartForward**, or **MeetingResponse** command requests to reference the result set.

The client **MUST** store the value of **LongId** as an opaque string of up to 256 characters.

2.2.1.14.2.2 Properties

In a **Search** request, the **Properties** element encapsulates the schema in which the client wants to have search results formatted. In a **Search** response, the **Properties** element encapsulates the properties for each search result.

Parent elements	Child elements	Data type	Number allowed
<Schema> (request only)	<Schema> (request only)	Container	0...1 (Optional, request only)
<Result> (response only)	<Result> (response only)		1...1 (Required, response only)

The **Search** command response **Properties** element is a container for properties that apply to an individual entry that matches the **Query** element search string. For example, the **Properties** element contains an element for each nonempty, text-valued **GAL** property that is attached to the matching GAL entry. Only those properties that are attached to the specific GAL entry are returned; therefore different sets of properties can be returned in the response XML for different matching GAL entries.

Each element in the **Properties** container is scoped to the appropriate namespace that is specified in the top-level **Search** element.

2.2.1.14.2.3 Range

The **Search** command XML **Range** element is used in both the request and response XML documents. In the request XML, the **Range** element specifies the range of matching entries to return. In the response XML, the **Range** element specifies the number of matching entries that are being returned.

Parent elements	Child elements	Data type	Number allowed
<Options> (request) <Store> (response)	None	Zero-based range in the form m-n	0...1 (optional)

The **Range** element value specifies a number of entries, but indicates different things depending on whether the element is in the request or the response XML.

The format of the **Range** element value is in the form of a zero-based index specifier, formed with a numeric value, a hyphen, and another numeric value: "m-n." The *m* and *n* indicates the lowest and highest index of a zero-based array that would hold the items. For example, a **Range** element value of 0–9 indicates 10 items, and 0–10 indicates 11 items. A **Range** element value of 0–0 indicates 1 item.

In the request XML, the **Range** element value specifies the range of entries to be returned to the client.

In the response XML, the **Range** element value specifies the actual number of entries that are returned in the response. The **Total** element in the response XML indicates the total number of entries that matched the **Query** element value.

Search results are stored in a **search folder** on the server. This way, when a client comes back with the same query but a new row range, rows are pulled from the result set that is currently stored in the **search folder**. The entire result set does not have to be rebuilt.

2.2.1.14.2.4 Response

The **Response** element is a container for the results that are returned from the server.

Parent elements	Child elements	Data type	Number allowed
<Search> (response only)	<Store> (response only)	Container	1...1 (required)

2.2.1.14.2.5 Result

The **Search** command response XML **Result** element is a container for an individual matching **mailbox** item.

Parent elements	Child elements	Data type	Number allowed
<Store> <Response>	<Properties> <Class> LongId ParentId	Container	1...N (required)

One **Result** element is present for each match that is found. If no matches are found, an empty **Result** element is present in the **Store** container element of the response XML.

Inside the **Result** element, the **Properties** element contains a list of nonempty text properties on the entry.

When the store being searched is the Mailbox:

- There is one **Result** element for each match that is found in the mailbox. If no matches are found, an empty **Result** element is present in the **Store** container element of the response XML.
- Inside the **Result** element, the **Properties** element contains a list of requested properties for the mailbox item.

When the store that is being searched is the Document Library:

- The first result that is returned in the **Search** response is the metadata for the root folder or item to which the **LinkId** is pointing. The client can choose to ignore this entry if it does not require it.
- If the **LinkId** in the request points to a folder, the metadata properties of the folder are returned as the first item, and the contents of the folder are returned as subsequent results. The **Range** element applies to these results with no difference; for example, the index 0 would always be for the root item to which the link is pointing.
- If the **LinkId** in the request points to an item, only one result is returned: the metadata for the item.
- Inside the **Result** element, the **Properties** element contains a list of requested properties for the mailbox item.

2.2.1.14.2.6 Search

The **Search** element is the top-level element in the **XML** document for the **Search** command. The element identifies the body of the **HTTP** Post as containing a **Search** command.

Parent elements	Child elements	Data type	Number allowed
None	<Store> (request) <Status>, <Response> (response)	Container	1...1 (required)

2.2.1.14.2.7 Status

The **Search** command response **XML Status** element indicates whether the server encountered an error while it was processing the search query.

Parent elements	Child elements	Data type	Number allowed
<Store>, <Response> (response only)	None	Integer	1...1 (required)

The following table specifies valid values for the **Status** element as a child of the **Search** node in the search response.

Value	Meaning
1	Success
3	Server error

The following table specifies valid values for the **Status** element as a child of the **Store** element in the search response.

Value	Meaning
1	Success.
2	Protocol violation/XML validation error.
3	Server error.
4	Bad Link
5	Access Denied
6	Not Found
7	Connection Failed (try again)
8	The search query is too complex.
9	Unable to execute this query because Content Indexing is not loaded.
10	Search timed out.
11	Bad CollectionId (the client MUST perform a FolderSync).
12	Server reached the end of the range that is retrievable by synchronization.
13	Access Blocked (policy restriction)
14	Credentials Required to Continue

The **Status** element value indicates only that the **Search** command was processed correctly. It does not indicate whether any matches were found. The **Total** and **Range** response XML elements indicate how many matches were found and returned, respectively.

The response will contain multiple **Status** elements. The **Status** element indicates the processing status of the overall **Search** command when the **Search** element is the immediate parent of the **Status** element. When the immediate parent of the **Status** command is the **Store** element, that **Status** element indicates the processing status for only that store. This structure was chosen to enable possible future expansion of the command to searching multiple locations, **address lists**, and **contacts** folders.

2.2.1.14.2.8 Store

In the **Search** command request **XML**, the **Store** element is a container for elements that specify the location, string, and options for the search. In the **Search** command response **XML**, the **Store** element contains the **Status**, **Result**, **Range**, and **Total** elements that contain the returned **mailbox** entries.

Parent elements	Child elements	Data type	Number allowed
<Search> (request) <Response> (response)	<Name>, <Query>, <Options> (request) <Status>, <Result>, <Range>, <Total> (response)	Container	1...1 (required)

2.2.1.14.2.9 Total

The **Search** command response **XML** element **Total** indicates the total number of **mailbox** entries that matched the search **Query** element value.

Parent elements	Child elements	Data type	Number allowed
Store	None	Integer	1...1 (required)

The value of the **Total** element does not always equal the number of entries that are returned. To determine the number of entries that are returned by the **Search** command, use the **Range** element value.

The **Total** element indicates the number of entries that are available. In cases where all the results are returned in the response **XML**, the value of the **Total** element is one more than the end-index value that is provided in the **Range** element. For example, if the **Search** command returns 15 entries, the value of the **Range** element is 0–14, while the value of the **Total** element is 15.

The **Total** element **SHOULD** be used by clients to determine whether more matching entries were found in the mailbox than have been returned by the **Search** command. For example, a device might perform an initial search and specify a requested **Range** of 0–4 (return 5 entries

maximum). If the **Total** element indicates that there are actually 25 matching items, the device can then enable the user to retrieve the full results.

2.2.1.15 SendMail

The **SendMail** command is used by clients to send **MIME**-formatted e-mail messages to the server.

2.2.1.15.1 Request

This command is issued in the **HTTP POST** command **URI**, and does not contain an XML body. The body will instead contain the **MIME**-formatted message. The *SaveInSent* **HTTP POST** command parameter provides the option of storing a copy of the message in the **Sent Mail folder**. **Attachments** can also be included in the message as a **MIME** part.

The Content-Type **MIME header field** **MUST** be set to "message/rfc822".

For more details about the RFC 822 format, see [RFC822]. [RFC821]-formatted messages are not supported.

Messages **SHOULD NOT** be saved directly to the local Sent Mail folder by the client; instead, messages can use the *SaveInSent* parameter to automatically have the messages saved on the server. It is not possible to reconcile the local Sent Mail folder with the server's Sent Mail folder by using the **Sync** command. Items in the server's Sent Mail folder can be added to the client by using the **Sync** command, but it is not possible to add items that are in the local **Sent Mail folder** to the server.

To instruct the server to save the outgoing e-mail message in the user's Sent Mail folder in the mailbox, include the *SaveInSent* parameter in the **HTTP POST** command **URI** and set the parameter to 'T'. To instruct the server not to save the outgoing e-mail message in the user's Sent Mail folder, omit the *SaveInSent* parameter. The *SaveInSent* parameter is set to 'F' by default. <25>

The **From:** header in the outgoing message is set on the server to the primary e-mail address of the logged-on user. If the message might be overwritten on the server, the client can choose to not send the message to the server. The same is true for the **SmartReply** and **SmartForward** commands.

2.2.1.15.2 Response

The response from the server contains the **HTTP** status code.

2.2.1.16 Settings

The **Settings** command supports get and set operations on global properties. <23>

The **Get** and **Set** operations act on named properties. In the context of the **Get** and **Set** operations, each **named property** can contain a set of property-specific data nodes.

The **Settings** command can contain multiple get and set requests and responses in any order. The implication of this batching mechanism is that commands are executed in the order in which they are received and that the ordering of get and set responses will match the order of those commands in the request.

The following is the generic form of the **Settings** request:

```
<Settings>
    <PropertyName>
        Data nodes
    </PropertyName>
    ...
</Settings>
```

The **PropertyName** is a named property (that is, the actual name of the property). The **Settings** command can be used on the following named properties:

- **OOF**
- Device Password
- Device Information
- User Information

The argument or data nodes are **Get** or **Set**, which can also have their own arguments. It is up to the individual property handlers to parse and interpret them as necessary.

It is possible to have many **PropertyName** nodes. Each property **MUST** be processed in order. There can be cases in which one property call affects another property call or the same property is in the **Settings** request twice. The responses will come back in the same order in which they were requested.

Each response has a global status response, which is mainly for protocol errors, followed by the property responses.

The **Status** node **SHOULD** return Success if **Settings** is returning property responses. If the command was not successful, the processing of the request cannot begin, no property responses are returned, and **Status** **SHOULD** indicate a protocol error.

Any error other than a protocol error is returned in the status codes of the individual property responses. All property responses, regardless of the property, **MUST** contain a status tag to indicate success or failure. This status node **MUST** be the first node in the property response.

The *Return data* is specified by the individual properties.

2.2.1.16.1 Request

The following code shows the XSD for the **Settings** command request.

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<xs:schema
  xmlns:tns="Settings:"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="Settings:"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="EmptyStringType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="NonEmptyStringType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="DeviceInformationStringType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="1024"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="Settings">
    <xs:complexType>
      <xs:all>
        <xs:element name="Oof" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice>
              <xs:element name="Get">
                <xs:complexType>

```

```

        <xs:sequence>
            <xs:element name="BodyType"
type="tns:NonEmptyStringType" minOccurs="1" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="Set">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="OofState"
type="tns:NonEmptyStringType" minOccurs="0" maxOccurs="1" />
            <xs:element name="StartTime"
type="tns:NonEmptyStringType" minOccurs="0" maxOccurs="1" />
            <xs:element name="EndTime"
type="tns:NonEmptyStringType" minOccurs="0" maxOccurs="1" />
            <xs:element name="OofMessage" minOccurs="0"
maxOccurs="3">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="AppliesToInternal"
type="tns:EmptyStringType" minOccurs="0" maxOccurs="1" />
                        <xs:element name="AppliesToExternalKnown"
type="tns:EmptyStringType" minOccurs="0" maxOccurs="1" />
                        <xs:element name="AppliesToExternalUnknown"
type="tns:EmptyStringType" minOccurs="0" maxOccurs="1" />
                        <xs:element name="Enabled"
type="tns:NonEmptyStringType" minOccurs="0" maxOccurs="1" />
                        <xs:element name="ReplyMessage"
type="xs:string" minOccurs="0" maxOccurs="1" />
                        <xs:element name="BodyType"
type="tns:NonEmptyStringType" minOccurs="0" maxOccurs="1" />
                    </xs:all>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>

```

```

        </xs:complexType>
    </xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="DevicePassword" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:choice>
            <xs:element name="Set">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Password" type="xs:string"
minOccurs="1" maxOccurs="1" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
</xs:element>
<xs:element name="DeviceInformation" minOccurs="0"
maxOccurs="1">
    <xs:complexType>
        <xs:choice>
            <xs:element name="Set">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="Model"
type="tns:DeviceInformationStringType" maxOccurs="1" minOccurs="0" />
                        <xs:element name="IMEI"
type="tns:DeviceInformationStringType" maxOccurs="1" minOccurs="0" />
                        <xs:element name="FriendlyName"
type="tns:DeviceInformationStringType" maxOccurs="1" minOccurs="0" />
                        <xs:element name="OS"
type="tns:DeviceInformationStringType" maxOccurs="1" minOccurs="0" />

```



```

        <xs:element name="OSLanguage"
type="tns:DeviceInformationStringType" maxOccurs="1" minOccurs="0" />
        <xs:element name="PhoneNumber"
type="tns:DeviceInformationStringType" maxOccurs="1" minOccurs="0" />
        <xs:element
name="UserAgent" type="tns:DeviceInformationStringType" maxOccurs="1"
minOccurs="0" />
    </xs:all>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="UserInformation" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:choice>
            <xs:element minOccurs="1" maxOccurs="1" name="Get"
type="tns:EmptyStringType"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
</xs:all>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.16.1.1 Settings

The **Settings** element is the top-level element in the **XML** document for the **Settings** command.

Parent elements	Child elements	Data type	Number allowed
None	Named Property nodes <Status/> (response	Container	1...1 (required)

	only)		
--	-------	--	--

The **Settings** element encapsulates one or more named property nodes that contain actions and arguments that apply to those properties.

The **Settings** command can be used on the following named properties:

- **Oof**
- **DevicePassword**
- **DeviceInformation**
- **UserInfo**

It is possible to have many **PropertyName** nodes. Each property **MUST** be processed in order. There can be cases in which one property call affects another property call or the same property is in the **Settings** request twice. The responses will come back in the same order in which they were requested.

2.2.1.16.1.2 Oof

The **Oof** element specifies a **named property** node for retrieving and setting **Out of Office (OOF)** information.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Get> <Set> (request only) <Status> (response only)	Container	0...1

The **Settings** command will support **Get** and **Set** operations for the OOF property. The OOF property enables a user to do the following:

- Specify whether the user is currently out of office.
- Schedule an out of office message to be sent between a particular start date and end date.
- Specify the message that is to be shown to various audiences when the mobile device user is out of office.

OOF Get Request and Response

The **Get** command within the **Oof** element enables the client to retrieve OOF information from the server. The client specifies the **BodyType** to be retrieved and the server will return all OOF information and messages.

There is one **OofMessage** node per audience in an OOF **Get** response. If a sender group is not allowed and is disabled (an unknown external sender can be disabled by the administrator), an

OofMessage node is not reported to the client in a **Get** response. If the sender group is allowed, but is disabled and has no Reply message (specified by the **ReplyMessage** element), an **OofMessage** node is still reported to the client.

If the client does not receive a group, it is presumably because the client does not have permission to enter settings for that group; in such a case, any attempt to set those properties results in an Access Denied status code.

OOF Set Request and Response

The **Set** command enables the client to set the OOF status, time OOF, and OOF messages for one or more of the following groups:

- Internal
- External Known Senders (such as **contacts**)
- External Unknown Senders

2.2.1.16.1.3 Get

The **Get** command enables the client to retrieve information from the server for any **named property** that supports **Get**.

Parent elements	Data type	Number allowed
<Oof> <UserInformation> (request only)	Container	0...1

Child elements in an <Oof> request	Child elements in an <Oof> response	Child elements in a <UserInformation> request or response
<BodyType>	<OofState> <StartTime> <EndTime> <OofMessage>	<EmailAddresses> (response only)

Only the **OOF** and User Information named properties support **Get**.

In an **Oof** request, the client specifies the body type to be retrieved and the server will return all OOF settings and messages for that body type.

2.2.1.16.1.4 Set

The **Set** command enables the client to set information on the server for any **named property** that supports **Set**.

Parent elements	Data type	Number allowed
<Oof> (request only) <DeviceInformation> <DevicePassword>	Container	0...1 (required)

Child elements in an <Oof> request	Child elements in an <Oof> response
<OofState> <StartTime> <EndTime> <OofMessage>	None

Child elements in a <DeviceInformation> request	Child elements in a <DeviceInformation> response
<Model> <IMEI> <FriendlyName> <OS> <OSLanguage> <PhoneNumber>	<Status>

Child elements in a <DevicePassword> request	Child elements in a <DevicePassword> response
<Password>	<Status>

The named properties that support **Set** are **OOF**, Device Information, and Device Password.

2.2.1.16.1.4.1 OOF Property

The **Set** command enables the client to set the following in the OOF property:

- OOF state
- Start time and end time, if the user wants to schedule an OOF message
- OOF message or messages for one or more of the supported audiences

2.2.1.16.1.4.2 Device Information Property

Set enables the client to specify values for any of the Device Information parameters. The following statements apply to the **Set** command request implementation:

- The client **MUST** specify all supported Device Information parameters in the **Set** request.
- The client or server makes no assumptions about ordering. The Device Information parameters can be specified in any order.
- To delete a given Device Information value, the client sends the **Set** command with an empty element for that parameter. Active Sync will set that parameter to **NULL**.

2.2.1.16.1.4.3 Device Password Property

The **Set** command enables the client to set or clear the recovery password of the device.

2.2.1.16.1.5 OofState

The **OofState** element specifies the availability of the **OOF** property.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	None	Integer	0...1 (optional)

The following table lists the valid values for **OofState**.

Value	Meaning
0	The OOF property is disabled.
1	The OOF property is global.
2	The OOF property is time-based.

The values of **OofState** match those of the Availabilities Service enumeration. **OofState** MUST be set to 2 if the **StartTime** and **EndTime** elements are present.

2.2.1.16.1.6 StartTime

The **StartTime** element is used with the **EndTime** element to specify the range of time during which the user is out of office.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	None	DateTime	1...1 (required)

The **StartTime** element can be present within the **Get** element of the **Settings** response for the **OOF** property, or within the **Set** element of the **Settings** request for the **OOF** property.

If **StartTime** is present, the **EndTime** element MUST be present also, and **OofState** MUST be set to 2. Otherwise, the client will receive a protocol error.

2.2.1.16.1.7 EndTime

The **EndTime** element is used with the **StartTime** element to specify the range of time during which the user is out of office.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	None	DateTime	1...1 (required)

The **EndTime** element can be present within the **Get** element of the **Settings** response for the **OOF** property or within the **Set** element of the **Settings** request for the **OOF** property.

If **EndTime** is present, the **StartTime** element MUST be present also, and **OofState** MUST be set to 2. Otherwise, the client will receive a protocol error.

2.2.1.16.1.8 OofMessage

The **OofMessage** element contains a set of elements that specify the **OOF** message for a particular audience.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	<AppliesToInternal> <AppliesToExternalKnown> <AppliesToExternalUnknown> <Enabled> <ReplyMessage> <BodyType>	Container	0...3

The **OOF** property supports the following three audiences for an OOF message:<24>

- **Internal**—A user who is in the same organization as the sending user.
- **Known external**—A user who is outside the sending user's organization, but is represented in the sending user's **contacts**.
- **Unknown external**—A user who is outside the sending user's organization and is not represented in the sending user's contacts.

The presence of one of the following elements, which are mutually exclusive, indicates the audience to which an OOF message pertains:

- **AppliesToInternal**—The OOF message is relevant to an internal audience.
- **AppliesToExternalKnown**—The OOF message is relevant to a known external audience.
- **AppliesToExternalUnknown**—The OOF message is relevant to an unknown external audience.

There is one **OofMessage** node per audience in an OOF **Get** response. If a sender group is not allowed and is disabled (an unknown external sender can be disabled by the administrator), an **OofMessage** node is not reported to the client in a **Get** response. If a sender group is allowed, but is disabled and has no reply message (specified by the **ReplyMessage** element), an **OofMessage** node is still reported to the client.

If the client does not receive a group, it is presumably because the client does not have permission to enter settings for that group; in such a case, any attempt to set those properties SHOULD result in an Access Denied status code.

In an OOF **Set** request, the client MUST NOT include the same *AppliesTo** element in more than one **OofMessage** element.

2.2.1.16.1.9 BodyType

The **BodyType** element is a string that specifies the format of the **OOF** message.

Parent elements	Child elements	Data type	Number allowed
<Get> (request only) <OofMessage>	None	String	1...1 (required) (On Gets) 0...1 (optional) under OofMessage, required if a message is set

The following are the permitted values for the **BodyType** element:

- Text
- HTML

If **BodyType** has the value HTML, all message strings are sent in the HTML format. If **BodyType** has the value Text, the message strings are sent in **plain text**. Because there is no default value, the **BodyType** node **MUST** be present on all Gets and on any OofMessage where the ReplyMessage has been set.

2.2.1.16.1.10 AppliesToInternal

The **AppliesToInternal** element indicates that the **OOF** message applies to internal users. (An internal user is a user who is in the same organization as the sending user.)

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Flag	0...1 (Choice)

When the **AppliesToInternal** element is present, its peer elements (that is, the other elements within the **OofMessage** element) specify the OOF settings with regard to internal users.

The following are the peer elements of the **AppliesToInternal** element:

- **Enabled**—Specifies whether an OOF message is sent to this audience while the sending user is OOF.
- **ReplyMessage**—Specifies the OOF message itself.
- **BodyType**—Specifies the format of the OOF message.

The **AppliesToInternal**, **AppliesToExternalKnown**, and **AppliesToExternalUnknown** elements, each of which indicates the audience to which an OOF message pertains, are mutually exclusive.

2.2.1.16.1.11 AppliesToExternalKnown

The **AppliesToExternalKnown** element indicates that the **OOF** message applies to known external users. (A known external user is a user who is outside the sending user's organization, but is represented in the sending user's **contacts**.)

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Flag	0...1 (Choice)

When the **AppliesToExternalKnown** element is present, its peer elements (that is, the other elements within the **OofMessage** element) specify the OOF settings with regard to known external users.

The following are the peer elements of the **AppliesToExternalKnown** element:

- **Enabled**—Specifies whether an OOF message is sent to this audience while the sending user is OOF.
- **ReplyMessage**—Specifies the OOF reply message. <24>
- **BodyType**—Specifies the format of the OOF message.

The **AppliesToInternal**, **AppliesToExternalKnown**, and **AppliesToExternalUnknown** elements, each of which indicates the audience to which an OOF message pertains, are mutually exclusive.

2.2.1.16.1.12 AppliesToExternalUnknown

The **AppliesToExternalUnknown** element indicates that the **OOF** message applies to unknown external users. (An unknown external user is a user who is outside the sending user's organization and is not represented in the sending user's **contacts**.)

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Flag	0...1 (Optional)

When the **AppliesToExternalKnown** element is present, its peer elements (that is, the other elements within the **OofMessage** element) specify the OOF settings with regard to unknown external users.

The following are the peer elements of the **AppliesToExternalKnown** element:

- **Enabled**—Specifies whether an OOF message is sent to this audience while the sending user is OOF.
- **ReplyMessage**—Specifies the OOF reply message. <8>
- **BodyType**—Specifies the format of the OOF message.

The **AppliesToInternal**, **AppliesToExternalKnown**, and **AppliesToExternalUnknown** elements, each of which indicates the audience to which an OOF message pertains, are mutually exclusive.

2.2.1.16.1.13 Enabled

The **Enabled** element specifies whether an **OOF** message is sent to this audience while the sending user is OOF.

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Integer	0...1 (optional)

The **Enabled** element is used in the OOF **Get** response to retrieve the current value. The **Enabled** element is used in the OOF **Set** request to set the value.

The value of **Enabled** is 1 if an OOF message is sent while the sending user is OOF; otherwise, the value is 0.

2.2.1.16.1.14 ReplyMessage

The **ReplyMessage** element specifies the message to be shown to a particular audience when the user is **OOF**.

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	String	0...1 (optional)

The **ReplyMessage** can be used in an OOF **Get** response to convey the requested OOF message, or in an OOF **Set** request to set the message that the client wants to send to a particular audience. In a **Set**, any **ReplyMessage** **MUST** also specify a **BodyType**.

The OOF property supports the following three audiences for an OOF message:

- **Internal**—A user who is in the same organization as the sending user.
- **Known external**—A user who is outside the sending user's organization, but is represented in the sending user's **contacts**.
- **Unknown external**—A user who is outside the sending user's organization and is not represented in the sending user's contacts.

The presence of one of the following elements, which are mutually exclusive, indicates the audience to which an OOF message pertains:<8>

- **AppliesToInternal**—The OOF message is relevant to an internal audience.
- **AppliesToExternalKnown**—The OOF message is relevant to a known external audience.

- **AppliesToExternalUnknown**—The OOF message is relevant to an unknown external audience.

2.2.1.16.1.15 DeviceInformation

The **DeviceInformation** element is the container node that is used for sending the client device's properties to the server.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Set>	Container	0...1 (required)

It is important to have pertinent information about a user's device for monitoring and troubleshooting. The **DeviceInformation** element is used in the **Settings** command to send the following information about a client device to the server:

- Device model
- International Mobile Equipment Identity (IMEI)
- Device friendly name
- Device operating system
- Telephone number
- Device operating system language
- User Agent

This information is reflected both in the output to administrative tasks (for example, reporting).<25>

The device information is represented as a flat list of settings under the **DeviceInformation** node in the **Settings** command. **DeviceInformation** has only one child element, **Set**, which contains the list of device information items in the request and the status in the response. The **DeviceInformation** property supports only the **Set** operation because this information is write-only from the device.

2.2.1.16.1.16 Model

The **Model** element specifies a name that generally describes the device of the client.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

The descriptive name of the device can be any string that the client chooses, but it **SHOULD** be a general description of the device. For example, the name of the manufacturer, the model

name, or the model number can be used. The server **SHOULD NOT** perform any validation of this string, so the client can submit any string.

2.2.1.16.1.17 IMEI

The **IMEI** element specifies a 15-character code that **MUST** uniquely identify a device.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

The server does not validate the IMEI format.

The device ID parameter that is currently included in the request URL is not precisely defined; licensees are free to populate the field as they want. To enable workable inventory-type report generation, an ID that uniquely identifies a device in the space of all devices is required. The **IMEI** element satisfies this requirement.

2.2.1.16.1.18 FriendlyName

The **FriendlyName** element specifies a name that **MUST** uniquely describe the client device.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

The friendly name of the device **SHOULD** be a string that is meaningful to the user. The server does not validate this value.

The friendly name of the device is typically specified during partnership creation if the user cradles the device to the desktop.

2.2.1.16.1.19 OS

The **OS** element specifies the operating system of the client device.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

Some information about the operating system of the device can be collected from the user agent string that is associated with requests from that client. The mapping from user agent to operating system is not one to one, however, and therefore does not provide sufficient information to troubleshoot and establish an inventory.

The **OS** element is a string value that enables the client to precisely specify the operating system of the device. The server does not perform any validation of this value, but clients **SHOULD** use the following convention:

<Operating System Product Name> <Operating System Major Version> <Operating System Minor Version>

2.2.1.16.1.20 OSLanguage

The **OSLanguage** element specifies the language that is used by the operating system of the client device.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

Knowledge of the user's language facilitates localization if the server is required to send localizable content to the client device. The server does not validate the value of the **OSLanguage** element.

2.2.1.16.1.21 PhoneNumber

The **PhoneNumber** element specifies a unique number that identifies the client device.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

The telephone number facilitates troubleshooting and device management by providing a well-known and unique identifier for the client device. The server does not validate the value of the **PhoneNumber** element.

2.2.1.16.1.22 UserAgent

The **UserAgent** element specifies the user agent.

Parent elements	Child elements	Data type	Number allowed
<Set> (DeviceInformation request only)	None	String	0...1 (optional)

The **UserAgent** element SHOULD contain the information in the User-Agent header. The User-Agent header SHOULD be removed from the **HTTP** request. The server does not validate the value of the **UserAgent** element.

2.2.1.16.1.23 DevicePassword

The **DevicePassword** element is a container node that is used to send the recovery password of the client device to the server.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Set>	Container	0...1 (optional)

Use the **Set** operation on the **DevicePassword** property enable the device to send or store a recovery password on the server. The recovery password is be stored in the user's **mailbox** and can be retrieved by the administrator or the end-user if the user forgets his or her password.

2.2.1.16.1.24 Password

The **Password** element specifies the recovery password of the client device, which is stored by the server.

Parent elements	Child elements	Data type	Number allowed
<Set> (DevicePassword request only)	None	String	1...1 (required)

The value of the **Password** element has a maximum length of 255 characters.

To clear an existing recovery password, the client MUST send a **Set** request with an empty **Password** element.

2.2.1.16.1.25 UserInformation

The **UserInformation** element is a container node that is used to request a list of a user's e-mail addresses from the server.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Get> (UserInformation request only) <Status> (UserInformation response only)	Container	0...1 (optional)

The list of a user's e-mail addresses can be useful, for example, for ensuring that the user is not included when performing a Reply to All operation to an e-mail message.

In a request, the **UserInformation** element contains the **Get** command to indicate that the server is to return all available e-mail addresses for the user.

The **Settings** command supports read-only access to the list of a user's various e-mail addresses via the **Get** command. The client is unable to write this information.

2.2.1.16.2 Response

2.2.1.16.2.1 Settings

The **Settings** element is the top-level element in the **XML** document for the **Settings** command.

Parent elements	Child elements	Data type	Number allowed
None	Named Property nodes <Status/> (response only)	Container	1...1 (required)

The **Settings** element encapsulates one or more named property nodes that contain actions and arguments that apply to those properties.

The **Settings** command can be used on the following named properties:

- **Oof**
- **DevicePassword**
- **DeviceInformation**
- **UserInformation**

It is possible to have many **PropertyName** nodes. Each property **MUST** be processed in order. There can be cases in which one property call affects another property call or the same

property is in the **Settings** request twice. The responses will come back in the same order in which they were requested.

2.2.1.16.2.2 Status

The **Status** element contains a code that indicates the success or failure of the **Settings** command and the success or failure of actions that are associated with a specific property node (**Oof**, **DeviceInformation**, **DevicePassword**, **UserInfo**).

Parent elements (response only)	Child elements	Data type	Number allowed for <Settings> parent	Number allowed for <Oof> parent, <Set> parent, or >UserInfo> parent
<Settings> <Oof> <Set> (only in a DeviceInformation or DevicePassword response)	None	Integer	1...1 (required)	0...1

The following table lists the valid values for the **Status** element in the context of the **Settings** command response. This is the status at the top level.

Value	Meaning
1	Success.
2	Protocol error.
3	Access denied. The user's access to the server is disabled.
4	Service/storage unavailable.
5	Invalid arguments. An unsupported property is specified.
6	Conflicting arguments. There are multiple property nodes (Oof , DeviceInformation , DevicePassword , UserInfo) with the same name.

The following table lists the valid values for **Status** in a **Settings** command **DeviceInformation** response.

Value	Meaning
-------	---------

1	Success.
2	Protocol error. The XML code is formatted incorrectly.
3	Access denied. An unauthorized write to the user's mailbox was tried.
4	Service/storage unavailable. Unable to write settings to the user's mailbox.
5	Invalid arguments. There is an unsupported value within the DeviceInformation node.
6	Conflicting arguments. There are multiple entries with same name in the DeviceInformation node.

The following table lists the values for **Status** in a **Settings** command **DevicePassword** response.

Value	Meaning
1	Success.
2	Protocol error. The XML code is formatted incorrectly.
3	Access denied. An unauthorized write to the user's mailbox was tried.
5	Invalid arguments. The specified password is too long.
7	Denied by policy. The administrator has disabled password recovery in this deployment.

The following table lists the values for **Status** in a **Settings** command **UserInformation** response.

Value	Meaning
1	Success.
2	Protocol error. The XML code is formatted incorrectly.

The status is specified for the **Settings** command response and for each property node (**Oof**, **DeviceInformation**, **DevicePassword**, **UserInformation**) within **Settings**.

Error code values 100 to 255 are reserved for property-specific error codes and vary from property to property. Any status value that is not 1 is a failure.

If the status is not returned for a property node, the client assumes success.

2.2.1.16.2.3 Oof

The **Oof** element specifies a **named property** node for retrieving and setting **OOF** information.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Get> <Set> (request only) <Status> (response only)	Container	0...1

The **Settings** command will support **Get** and **Set** operations for the OOF property. The OOF property enables a user to do the following:

- Specify whether the user is currently out of office.
- Schedule an out of office message to be sent between a particular start date and end date.
- Specify the message that is to be shown to various audiences when the mobile device user is out of office.

OOF Get Request and Response

The **Get** command within the **Oof** element enables the client to retrieve OOF information from the server. The client specifies the **BodyType** to be retrieved and the server will return all OOF information and messages.

There is one **OofMessage** node per audience in an OOF **Get** response. If a sender group is not allowed and is disabled (an unknown external sender can be disabled by the administrator), an **OofMessage** node is not reported to the client in a **Get** response. If the sender group is allowed, but is disabled and has no Reply message (specified by the **ReplyMessage** element), an **OofMessage** node is still reported to the client.

If the client does not receive a group, it is presumably because the client does not have permission to enter settings for that group; in such a case, any attempt to set those properties MUST result in an Access Denied status code.

OOF Set Request and Response

The **Set** command enables the client to set the OOF status, time OOF, and OOF messages for one or more of the following groups:

- Internal
- External Known Senders (such as **contacts**)
- External Unknown Senders

2.2.1.16.2.4 Get

The **Get** command enables the client to retrieve information from the server for any **named property** that supports **Get**.

Parent elements	Data type	Number allowed
<Oof> <UserInformation> (request only)	Container	0...1

Child elements in an <Oof> request	Child elements in an <Oof> response	Child elements in a <UserInformation> request or response
<BodyType>	<OofState> <StartTime> <EndTime> <OofMessage>	None

Only the **OOF** and User Information named properties support **Get**.

In an **Oof** request, the client specifies the body type to be retrieved and the server will return all OOF settings and messages for that body type.

2.2.1.16.2.5 Set

The **Set** command enables the client to set information on the server for any **named property** that supports **Set**.

Parent elements	Data type	Number allowed
<Oof> (request only) <DeviceInformation> <DevicePassword>	Container	0...1 (required)

Child elements in an <Oof> request	Child elements in an <Oof> response
<OofState>	None

<StartTime> <EndTime> <OofMessage>	
--	--

Child elements in a <DeviceInformation> request	Child elements in a <DeviceInformation> response
<Model> <IMEI> <FriendlyName> <OS> <OSLanguage> <PhoneNumber>	<Status>

Child elements in an <DevicePassword> request	Child elements in an <DevicePassword> response
<Password>	<Status>

The named properties that support **Set** are **OOF**, Device Information, and Device Password.

2.2.1.16.2.5.1 OOF Property

The **Set** command enables the client to set the following in the OOF property:

- OOF state
- Start time and end time, if the user wants to schedule an OOF message
- OOF message or messages for one or more of the supported audiences

2.2.1.16.2.5.2 Device Information Property

Set enables the client to specify values for any of the Device Information parameters. The following statements apply to the **Set** command request implementation:

- The client **MUST** specify all supported Device Information parameters in the **Set** request.
- The client or server makes no assumptions about ordering. The Device Information parameters can be specified in any order.

- To delete a given Device Information value, the client **MUST** send the **Set** command with an empty element for that parameter. Active Sync will set that parameter to **NULL**.

2.2.1.16.2.5.3 Device Password Property

The **Set** command enables the client to set or clear the recovery password of the device.

2.2.1.16.2.6 OofState

The **OofState** element specifies the availability of the **OOF** property.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	None	Integer	0...1 (optional)

The following table lists the valid values for **OofState**.

Value	Meaning
0	The OOF property is disabled.
1	The OOF property is global.
2	The OOF property is time-based.

The values of **OofState** match those of the Availabilities Service enumeration. **OofState** **MUST** be set to 2 if the **StartTime** and **EndTime** elements are present.

2.2.1.16.2.7 StartTime

The **StartTime** element is used with the **EndTime** element to specify the range of time during which the user is **OOF**.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	None	DateTime	1...1 (required)

The **StartTime** element can be present within the **Get** element of the **Settings** response for the OOF property, or within the **Set** element of the **Settings** request for the OOF property.

If **StartTime** is present, the **EndTime** element **MUST** also be present, and **OofState** **MUST** be set to 2. Otherwise, the client will receive a protocol error. In addition, .

2.2.1.16.2.8 EndTime

The **EndTime** element is used with the **StartTime** element to specify the range of time during which the user is **OOF**.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	None	DateTime	1...1 (required)

The **EndTime** element can be present within the **Get** element of the **Settings** response for the OOF property, or within the **Set** element of the **Settings** request for the OOF property.

If **EndTime** is present, the **StartTime** element **MUST** also be present, and **OofState** **MUST** be set to 2. Otherwise, the client will receive a protocol error.

2.2.1.16.2.9 OofMessage

The **OofMessage** element contains a set of elements that specify the **OOF** message for a particular audience.

Parent elements	Child elements	Data type	Number allowed
<Get> (Oof response only) <Set> (Oof request only)	<AppliesToInternal> <AppliesToExternalKnown> <AppliesToExternalUnknown> <Enabled> <ReplyMessage> <BodyType>	Container	0...3

The OOF property supports the following three audiences for an OOF message:<7>

- Internal—A user who is in the same organization as the sending user.
- Known external—A user who is outside the sending user's organization, but is represented in the sending user's **contacts**.

- **Unknown external**—A user who is outside the sending user's organization and is not represented in the sending user's contacts.

The presence of one of the following elements, which are mutually exclusive, indicates the audience to which an OOF message pertains:

- **AppliesToInternal**—The OOF message is relevant to an internal audience.
- **AppliesToExternalKnown**—The OOF message is relevant to a known external audience.
- **AppliesToExternalUnknown**—The OOF message is relevant to an unknown external audience.

There is one **OofMessage** node per audience in an OOF **Get** response. If a sender group is not allowed and is disabled (an unknown external sender can be disabled by the administrator), an **OofMessage** node is not reported to the client in a **Get** response. If a sender group is allowed, but is disabled and has no reply message (specified by the **ReplyMessage** element), an **OofMessage** node is still reported to the client.

If the client does not receive a group, it is presumably because the client does not have permission to enter settings for that group; in such a case, any attempt to set those properties **MUST** result in an Access Denied status code.

In an OOF **Set** request, the client **MUST** not include the same *AppliesTo** element in more than one **OofMessage** element.

2.2.1.16.2.10 BodyType

The **BodyType** element is a string that specifies the format of the **OOF** message.

Parent elements	Child elements	Data type	Number allowed
<Get> (request only) <OofMessage>	None	String	1...1 (required)

The following are the permitted values for the **BodyType** element:

- Text
- HTML

If **BodyType** has the value HTML, all message strings are sent in the HTML format. If **BodyType** has the value Text, the message strings are sent in **plain text**. Because there is no default value, the **BodyType** node **MUST** be present.

2.2.1.16.2.11 AppliesToInternal

The **AppliesToInternal** element indicates that the **OOF** message applies to internal users. (An internal user is a user who is in the same organization as the sending user.)

Parent elements	Child elements	Data type	Number allowed
<Get> (request only) <OofMessage>	None	String	1...1 (required)

When the **AppliesToInternal** element is present, its peer elements (that is, the other elements within the **OofMessage** element) specify the OOF settings with regard to internal users.

The following are the peer elements of the **AppliesToInternal** element:

- **Enabled**—Specifies whether an OOF message is sent to this audience while the sending user is OOF.
- **ReplyMessage**—Specifies the OOF message itself.
- **BodyType**—Specifies the format of the OOF message.

The **AppliesToInternal**, **AppliesToExternalKnown**, and **AppliesToExternalUnknown** elements, each of which indicates the audience to which an OOF message pertains, are mutually exclusive.

2.2.1.16.2.12 AppliesToExternalKnown

The **AppliesToExternalKnown** element indicates that the **OOF** message applies to known external users. (A known external user is a user who is outside the sending user's organization, but is represented in the sending user's **contacts**.)

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Flag	0...1 (Choice)

When the **AppliesToExternalKnown** element is present, its peer elements (that is, the other elements within the **OofMessage** element) specify the OOF settings with regard to known external users.

The following are the peer elements of the **AppliesToExternalKnown** element:

- **Enabled**—Specifies whether an OOF message is sent to this audience while the sending user is OOF.
- **ReplyMessage**—Specifies the OOF reply message.<7>
- **BodyType**—Specifies the format of the OOF message.

The **AppliesToInternal**, **AppliesToExternalKnown**, and **AppliesToExternalUnknown** elements, each of which indicates the audience to which an OOF message pertains, are mutually exclusive.

2.2.1.16.2.13 AppliesToExternalUnknown

The **AppliesToExternalUnknown** element indicates that the **OOF** message applies to unknown external users. (An unknown external user is a user who is outside the sending user's organization and is not represented in the sending user's **contacts**.)

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Flag	0...1 (Optional

When the **AppliesToExternalKnown** element is present, its peer elements (that is, the other elements within the **OofMessage** element) specify the OOF settings with regard to unknown external users.

The following are the peer elements of the **AppliesToExternalKnown** element:

- **Enabled**—Specifies whether an OOF message is sent to this audience while the sending user is OOF.
- **ReplyMessage**—Specifies the OOF reply message.<7>
- **BodyType**—Specifies the format of the OOF message.

The **AppliesToInternal**, **AppliesToExternalKnown**, and **AppliesToExternalUnknown** elements, each of which indicates the audience to which an OOF message pertains, are mutually exclusive.

2.2.1.16.2.14 Enabled

The **Enabled** element specifies whether an **OOF** message is sent to this audience while the sending user is OOF.

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	Integer	0...1 (optional)

The **Enabled** element is used in the OOF **Get** response to retrieve the current value. The **Enabled** element is used in the OOF **Set** request to set the value.

The value of **Enabled** is 1 if an OOF message is sent while the sending user is OOF; otherwise, the value is 0.

2.2.1.16.2.15 ReplyMessage

The **ReplyMessage** element specifies the message to be shown to a particular audience when the user is **OOF**.

Parent elements	Child elements	Data type	Number allowed
<OofMessage>	None	String	0...1 (optional)

The **ReplyMessage** can be used in an OOF **Get** response to convey the requested OOF message, or in an OOF **Set** request to set the message that the client wants to send to a particular audience.

The OOF property supports the following three audiences for an OOF message:<7>

- Internal—A user who is in the same organization as the sending user.
- Known external—A user who is outside the sending user's organization, but is represented in the sending user's **contacts**.
- Unknown external—A user who is outside the sending user's organization and is not represented in the sending user's contacts.

The presence of one of the following elements, which are mutually exclusive, indicates the audience to which an OOF message pertains:

- **AppliesToInternal**—The OOF message is relevant to an internal audience.
- **AppliesToExternalKnown**—The OOF message is relevant to a known external audience.
- **AppliesToExternalUnknown**—The OOF message is relevant to an unknown external audience.

2.2.1.16.2.16 DeviceInformation

The **DeviceInformation** element is the container node that is used for sending the client device's properties of the client device to the server.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Set>	Container	0...1 (required)

It is important to have pertinent information about a user's device for monitoring and troubleshooting. The **DeviceInformation** element is used in the **Settings** command to send the following information about a client device to the server:

- Device model
- International Mobile Equipment Identity (IMEI)
- Device friendly name
- Device operating system
- Telephone number
- Device operating system language

This information is reflected both in the Microsoft Office Outlook® Web Access mobile device console and the output to administrative tasks (for example, reporting).

The device information is represented as a flat list of settings under the **DeviceInformation** node in the **Settings** command. **DeviceInformation** has only one child element, **Set**, which contains the list of device information items in the request and the status in the response. The

DeviceInformation property supports only the **Set** operation because this information is write-only from the device.

2.2.1.16.2.17 DevicePassword

The **DevicePassword** element is a container node that is used to send the recovery password of the client device to the server.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Set>	Container	0...1 (optional)

Use the **Set** operation on the **DevicePassword** property to enable the device to send or store a recovery password on the server. The recovery password is stored in the user's **mailbox** and can be retrieved by the administrator or the end-user if the user forgets his or her password.

2.2.1.16.2.18 UserInformation

The **UserInformation** element is a container node that is used to request a list of a user's e-mail addresses from the server.

Parent elements	Child elements	Data type	Number allowed
<Settings>	<Get> <Status> (UserInformation response only)	Container	0...1 (optional)

The list of a user's e-mail addresses can be useful, for example, for ensuring that the user is not included when performing a Reply to All operation to an e-mail message.

In a request, the **UserInformation** element contains the **Get** command to indicate that the server is to return all available e-mail addresses for the user.

The **Settings** command supports read-only access to the list of a user's various e-mail addresses via the **Get** command. The client is unable to write this information.

2.2.1.16.2.19 EmailAddresses

The **EmailAddresses** element contains one or more e-mail addresses for the user.

Parent elements	Child elements	Data type	Number allowed
<Get> (response only)	<SMTPAddress> (response only)	Container	0...1 (optional)

2.2.1.16.2.20 SMTPAddress

The **SMTPAddress** element specifies one of the user's e-mail addresses.

Parent elements	Child elements	Data type	Number allowed
<EmailAddresses>	None	String	1...N (optional)

2.2.1.17 SmartForward

The **SmartForward** command is used by clients to forward messages without retrieving the full, original message from the server.

2.2.1.17.1 Request

The **SmartForward** command is issued in the **HTTP POST** command **URI**; the command does not specify additional information in an **XML** body. The *ItemId* command parameter specifies the server ID of the message item to be forwarded. The *CollectionId* command parameter specifies the server ID of the collection that contains the message to be forwarded. The *SaveInSent* command parameter specifies whether a copy of the forwarded message is saved in the **Sent Mail folder**.

Messages SHOULD NOT be saved directly to the local Sent Mail folder by the client. It is not possible to reconcile the local Sent Mail folder with the server's Sent Mail folder by using the **Sync** command. Items in the server's Sent Mail folder can be added to the client by using the **Sync** command, but it is not possible to add items that are in the local Sent Mail folder to the server.

To instruct the server to save the forwarded e-mail message in the user's Sent Mail folder in the **mailbox**, include the *SaveInSent* parameter in the **HTTP POST** command URI and set the parameter to 'T'. To instruct the server not to save the outgoing e-mail message in the user's Sent Mail folder, omit the *SaveInSent* parameter. The *SaveInSent* parameter is set to 'F' by default. <26>

The forwarded message is included as a **MIME**-formatted message in the body of the **POST** command request; the Content-Type MIME **header field** is set to "message/rfc822". For more details about the RFC 822 format, see [RFC822]. [RFC821]-formatted messages are not supported.

The **SmartForward** command can be applied to a **meeting**. When **SmartForward** is applied to a recurring meeting, the command's *Occurrence* parameter specifies the ID of a particular occurrence in the recurring meeting. If **SmartForward** is applied to a recurring meeting and the *Occurrence* parameter is absent, the server SHOULD forward the entire recurring meeting. If the value of *Occurrence* is invalid, the server SHOULD respond with an error.

When **SmartForward** is applied to an **appointment**, the original message is included by the server as an **attachment** to the outgoing message. When smart-forwarding a normal message or a meeting, **SmartForward**'s behavior is the same as that of the **SmartReply** command.

The **SmartForward** command is similar to the **SendMail** command, but the outgoing message consists of the new message followed by the text of the original message. The full text of the original message is sent. Using the server copy of the original message saves network bandwidth by not downloading the original message and then uploading it again with the forward.

The **SmartForward** command lists the message **recipients**.

By default, because the original message and the forward messages can use different **character sets**, this command will always send the outgoing message by using the UTF8 character set for the body of the message.

2.2.1.17.2 Response

The response from the server contains the **HTTP** status code.

A 500 HTTP status code is returned to the client if the client attempts to forward a message that has been either moved or deleted from the specified folder on the server.

2.2.1.18 SmartReply

The **SmartReply** command is used by clients to reply to messages without retrieving the full, original message from the server.

2.2.1.18.1 Request

The **SmartReply** command is issued in the **HTTP POST** command **URI**; the command does not specify any additional information in an **XML** body. The *ItemId* command parameter specifies the server ID of the message item that is being replied to. The *CollectionId* command parameter specifies the server ID of the **collection** that contains the message item to be replied to. The *SaveInSent* command parameter specifies whether a copy of the reply message is saved in the **Sent Mail folder**.

Messages SHOULD NOT be saved directly to the local Sent Mail folder by the client. It is not possible to reconcile the local Sent Mail folder with the server's Sent Mail folder by using the **Sync** command. Items in the server's Sent Mail folder can be added to the client by using the **Sync** command, but it is not possible to add items that are in the local Sent Mail folder, to the server.

To instruct the server to save the outgoing e-mail message in the user's Sent Mail folder in the **mailbox**, include the *SaveInSent* parameter in the HTTP POST command URI and set the parameter to 'T'. To instruct the server not to save the outgoing e-mail message in the user's Sent Mail folder, omit the *SaveInSent* parameter. The *SaveInSent* parameter is set to 'F' by default.<25>

The reply message is included as a **MIME**-formatted message in the body of the **POST** command request; the Content-Type MIME **header field** is set to "message/rfc822". For more details about the [RFC822] format, see [RFC822]. [RFC821]-formatted messages are not supported.

The **SmartReply** command can be applied to a meeting. When **SmartReply** is applied to a recurring meeting, the command's *Occurrence* parameter specifies the ID of a particular occurrence in the recurring meeting. If **SmartReply** is applied to a recurring meeting and the *Occurrence* parameter is absent, the server **SHOULD** reply to the entire recurring meeting. If the value of *Occurrence* is invalid, the server **SHOULD** respond with an error.

The **SmartReply** command is similar to the **SendMail** command, but the outgoing message consists of the new message followed by the text of the original message. The full text of the original message is sent. Using the server copy of the original message saves network bandwidth by not downloading the original message and then uploading it again with the reply.

The **SmartReply** command lists the message **recipients**, so it is used to implement both Reply and Reply-to-All functionality. It is the responsibility of the client to implement Reply and Reply-to-All functionality.

By default, because the original message and the reply messages can use different **character sets**, this command will always send the outgoing message by using the UTF8 character set for the body of the message.

2.2.1.18.2Response

The response from the server contains the **HTTP** status code.

A 500 HTTP status code is returned to the client if the client attempts to reply to a message that has been either moved or deleted from the folder on the server.

2.2.1.19 Sync

The **Sync** command synchronizes changes in a **collection** between the client and the server.

For more details about the AirSyncBase elements that are used by this command, see [MS-ASAIRS] section 2.2.

Synchronization is a two-step process. For each collection, the client **MUST** issue an initial **Sync** request by sending a synchronization key of 0. This request establishes a synchronization relationship with the server and initializes the synchronization state there. The server responds with an initial value of the synchronization key, which the client can then use to get the initial set of objects from the server. (From this point forward, client requests **SHOULD** always include the synchronization key that was received in the last response from the server.) The client then sends a **Sync** command request to the server with the response synchronization key and includes any changes that were made on the client.

If the client device has not yet synchronized a folder, there **SHOULD** be no client-side changes. The device **SHOULD** synchronize the entire folder, and then have changes, additions, and deletions applied.

The response from the server indicates whether the client's changes were accepted, and includes any changes that were made on the server. The server response also contains a synchronization key that is to be used for the next synchronization session for the folder. The **Outbox folder** cannot be synchronized and the **Sent Mail folder** is a one-way synchronization from the server to the client.

[MS-ASCMD] has been optimized for the case in which there are no changes to any of the collections that are specified in the **Sync** request. In such a case, the client can receive an empty response from the server. After the client receives an empty response, the client can issue an empty **Sync** request. The server then re-executes the previous request, which it cached.

The following diagram shows request and response processing by the client.

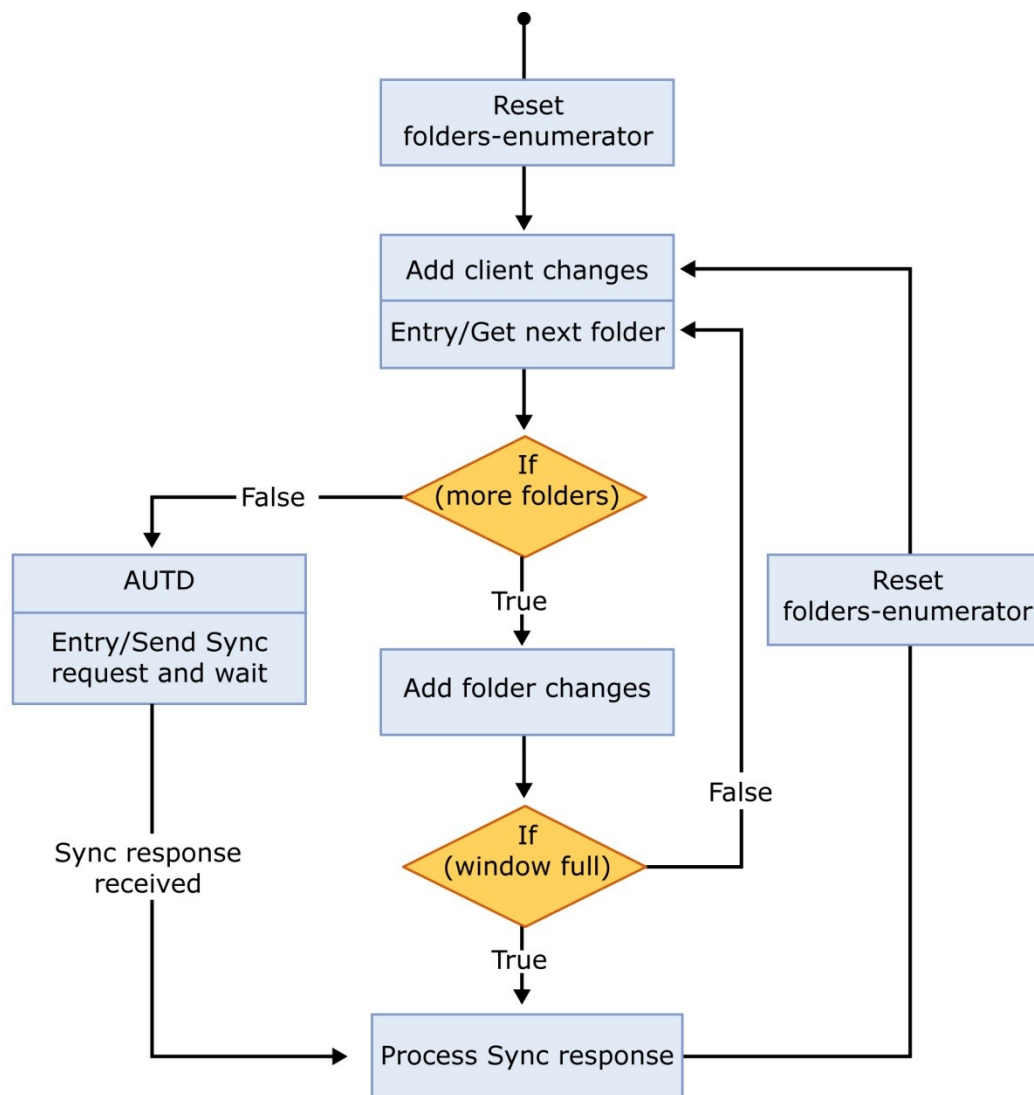


Figure 3: Sync command client processing

2.2.1.19.1 Request

The following **XML** is optional in the **Sync** command request body.

```

<?xml version="1.0" ?>

<xs:schema xmlns:tns="AirSync:" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="AirSync:"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:contacts="Contacts:"
xmlns:contacts2="Contacts2:" xmlns:calendar="Calendar:"
xmlns:email="Email:" xmlns:airsyncbase="AirSyncBase:"
xmlns:tasks="Tasks:">

  <xs:import namespace="Contacts2:"/>
  <xs:import namespace="Contacts:"/>

```



```

<xs:import namespace="Email:"/>
<xs:import namespace="Calendar:"/>
<xs:import namespace="AirSyncBase:"/>
<xs:import namespace="Tasks:"/>
<xs:element name="MIMESupport">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedByte">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="2" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="CollectionId">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="ServerId">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="FilterType">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedByte">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```

```

<xs:element name="SyncKey">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="Class" type="xs:string"/>
<xs:element name="Sync">
  <xs:complexType>
    <xs:sequence minOccurs="1" maxOccurs="1">
      <xs:element name="Collections" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded"
name="Collection">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="SyncKey">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:maxLength value="64"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element minOccurs="1" name="CollectionId">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:maxLength value="64"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element minOccurs="0" name="Supported">
                    <xs:complexType mixed="true">
                      <xs:sequence minOccurs="0">

```

```

        <xs:choice maxOccurs="unbounded">
            <xs:group ref="contacts:GhostingProps"/>
            <xs:group ref="contacts2:GhostingProps"/>
            <xs:group ref="calendar:GhostingProps"/>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="DeletesAsMoves"/>
<xs:element minOccurs="0" name="GetChanges"/>
<xs:element minOccurs="0" name="WindowSize">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element minOccurs="0" name="Options">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="FilterType" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedByte">
                        <xs:minInclusive value="0"/>
                        <xs:maxInclusive value="8"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element ref="airsyncbase:BodyPreference"
minOccurs="0" maxOccurs="unbounded" />
            <xs:element minOccurs="0" name="Conflict"
type="xs:unsignedByte"/>
            <xs:element minOccurs="0" maxOccurs="1"
name="MIMESupport">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedByte">

```

```

        <xs:minInclusive value="0" />
        <xs:maxInclusive value="2" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element minOccurs="0" maxOccurs="1"
name="MIMETruncation">
    <xs:simpleType>
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="0" />
            <xs:maxInclusive value="8" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="Commands">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element minOccurs="0"
maxOccurs="unbounded" name="Change">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="ServerId">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:maxLength value="64"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>
                        <xs:element name="ApplicationData">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:choice maxOccurs="unbounded">

```

	<xs:element ref="email:Flag"/>
	<xs:element ref="email:Read"/>
ref="calendar:OrganizerName"/>	<xs:element
ref="calendar:OrganizerEmail"/>	<xs:element
ref="calendar:Exceptions"/>	<xs:element
ref="calendar:Attendees"/>	<xs:element
ref="calendar:TimeZone"/>	<xs:element
ref="calendar:AllDayEvent"/>	<xs:element
ref="airsyncbase:NativeBodyType"/>	<xs:element
ref="airsyncbase:Body"/>	<xs:element
ref="calendar:BusyStatus"/>	<xs:element
ref="calendar:Categories"/>	<xs:element
ref="calendar:DtStamp"/>	<xs:element
ref="calendar:EndTime"/>	<xs:element
ref="calendar:Location"/>	<xs:element
ref="calendar:MeetingStatus"/>	<xs:element
ref="calendar:Reminder"/>	<xs:element
ref="calendar:Sensitivity"/>	<xs:element
ref="calendar:Subject"/>	<xs:element
ref="calendar:StartTime"/>	<xs:element
ref="calendar:UID"/>	<xs:element

ref="calendar:Recurrence"/>	<xs:element
ref="contacts:Anniversary"/>	<xs:element
ref="contacts:AssistantName"/>	<xs:element
ref="contacts:AssistantPhoneNumber"/>	<xs:element
ref="contacts:AssistnamePhoneNumber"/>	<xs:element
ref="contacts:Birthday"/>	<xs:element
ref="contacts:Business2PhoneNumber"/>	<xs:element
ref="contacts:BusinessAddressCity"/>	<xs:element
ref="contacts:BusinessAddressCountry"/>	<xs:element
ref="contacts:BusinessAddressPostalCode"/>	<xs:element
ref="contacts:BusinessAddressState"/>	<xs:element
ref="contacts:BusinessAddressStreet"/>	<xs:element
ref="contacts:BusinessFaxNumber"/>	<xs:element
ref="contacts:BusinessPhoneNumber"/>	<xs:element
ref="contacts:CarPhoneNumber"/>	<xs:element
ref="contacts:Categories"/>	<xs:element
ref="contacts:Children"/>	<xs:element
ref="contacts:CompanyName"/>	<xs:element
ref="contacts:Department"/>	<xs:element
ref="contacts:Email1Address"/>	<xs:element
ref="contacts:Email2Address"/>	<xs:element

ref="contacts:Email3Address"/>	<xs:element
ref="contacts:FileAs"/>	<xs:element
ref="contacts:FirstName"/>	<xs:element
ref="contacts:MiddleName"/>	<xs:element
ref="contacts:Home2PhoneNumber"/>	<xs:element
ref="contacts:HomeAddressCity"/>	<xs:element
ref="contacts:HomeAddressCountry"/>	<xs:element
ref="contacts:HomeAddressPostalCode"/>	<xs:element
ref="contacts:HomeAddressState"/>	<xs:element
ref="contacts:HomeAddressStreet"/>	<xs:element
ref="contacts:HomeFaxNumber"/>	<xs:element
ref="contacts:HomePhoneNumber"/>	<xs:element
ref="contacts:JobTitle"/>	<xs:element
ref="contacts:LastName"/>	<xs:element
ref="contacts:MobilePhoneNumber"/>	<xs:element
ref="contacts:OfficeLocation"/>	<xs:element
ref="contacts:OtherAddressCity"/>	<xs:element
ref="contacts:OtherAddressCountry"/>	<xs:element
ref="contacts:OtherAddressPostalCode"/>	<xs:element
ref="contacts:OtherAddressState"/>	<xs:element
ref="contacts:OtherAddressStreet"/>	<xs:element

ref="contacts:PagerNumber"/>	<xs:element
ref="contacts:RadioPhoneNumber"/>	<xs:element
ref="contacts:Spouse"/>	<xs:element
ref="contacts:Suffix"/>	<xs:element
ref="contacts:Title"/>	<xs:element
ref="contacts:WebPage"/>	<xs:element
ref="contacts:YomiCompanyName"/>	<xs:element
ref="contacts:YomiFirstName"/>	<xs:element
ref="contacts:Picture"/>	<xs:element
ref="contacts2:CustomerId"/>	<xs:element
ref="contacts2:GovernmentId"/>	<xs:element
ref="contacts2:IMAddress"/>	<xs:element
ref="contacts2:IMAddress2"/>	<xs:element
ref="contacts2:IMAddress3"/>	<xs:element
ref="contacts2:ManagerName"/>	<xs:element
ref="contacts2:CompanyMainPhone"/>	<xs:element
ref="contacts2:AccountName"/>	<xs:element
ref="contacts2:NickName"/>	<xs:element
ref="contacts2:MMS"/>	<xs:element
ref="contacts:YomiLastName"/>	<xs:element
ref="tasks:Complete"/>	<xs:element


```

ref="tasks:Subject"/>
ref="tasks:Categories"/>
ref="tasks:DateCompleted"/>
ref="tasks:DueDate"/>
ref="tasks:UtcDueDate"/>
ref="tasks:Importance"/>
ref="tasks:Recurrence"/>
ref="tasks:ReminderSet"/>
ref="tasks:ReminderTime"/>
ref="tasks:Sensitivity"/>
ref="tasks:StartDate"/>
ref="tasks:UtcStartDate"/>
</xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0"
maxOccurs="unbounded" name="Delete">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ServerId">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="64"/>

```

```

        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
  <xs:element minOccurs="0"
maxOccurs="unbounded" name="Add">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ClientId">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="64"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="ApplicationData">
          <xs:complexType>
            <xs:sequence>
              <xs:choice maxOccurs="unbounded">
                <xs:element
ref="calendar:OrganizerName"/>
                <xs:element
ref="calendar:OrganizerEmail"/>
                <xs:element
ref="calendar:Exceptions"/>
                <xs:element
ref="calendar:Attendees"/>
                <xs:element
ref="calendar:TimeZone"/>
                <xs:element
ref="calendar:AllDayEvent"/>
                <xs:element
ref="airsyncbase:NativeBodyType"/>
                <xs:element
ref="airsyncbase:Body"/>
              </xs:choice>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

ref="calendar:BusyStatus"/>	<xs:element
ref="calendar:Categories"/>	<xs:element
ref="calendar:DtStamp"/>	<xs:element
ref="calendar:EndTime"/>	<xs:element
ref="calendar:Location"/>	<xs:element
ref="calendar:MeetingStatus"/>	<xs:element
ref="calendar:Reminder"/>	<xs:element
ref="calendar:Sensitivity"/>	<xs:element
ref="calendar:Subject"/>	<xs:element
ref="calendar:StartTime"/>	<xs:element
ref="calendar:UID"/>	<xs:element
ref="calendar:Recurrence"/>	<xs:element
ref="contacts:Anniversary"/>	<xs:element
ref="contacts:AssistantName"/>	<xs:element
ref="contacts:AssistantPhoneNumber"/>	<xs:element
ref="contacts:AssistnamePhoneNumber"/>	<xs:element
ref="contacts:Birthday"/>	<xs:element
ref="contacts:Business2PhoneNumber"/>	<xs:element
ref="contacts:BusinessAddressCity"/>	<xs:element
ref="contacts:BusinessAddressCountry"/>	<xs:element
ref="contacts:BusinessAddressPostalCode"/>	<xs:element

ref="contacts:BusinessAddressState"/>	<xs:element
ref="contacts:BusinessAddressStreet"/>	<xs:element
ref="contacts:BusinessFaxNumber"/>	<xs:element
ref="contacts:BusinessPhoneNumber"/>	<xs:element
ref="contacts:CarPhoneNumber"/>	<xs:element
ref="contacts:Categories"/>	<xs:element
ref="contacts:Children"/>	<xs:element
ref="contacts:CompanyName"/>	<xs:element
ref="contacts:Department"/>	<xs:element
ref="contacts:Email1Address"/>	<xs:element
ref="contacts:Email2Address"/>	<xs:element
ref="contacts:Email3Address"/>	<xs:element
ref="contacts:FileAs"/>	<xs:element
ref="contacts:FirstName"/>	<xs:element
ref="contacts:MiddleName"/>	<xs:element
ref="contacts:Home2PhoneNumber"/>	<xs:element
ref="contacts:HomeAddressCity"/>	<xs:element
ref="contacts:HomeAddressCountry"/>	<xs:element
ref="contacts:HomeAddressPostalCode"/>	<xs:element
ref="contacts:HomeAddressState"/>	<xs:element
ref="contacts:HomeAddressStreet"/>	<xs:element

ref="contacts:HomeFaxNumber"/>	<xs:element
ref="contacts:HomePhoneNumber"/>	<xs:element
ref="contacts:JobTitle"/>	<xs:element
ref="contacts:LastName"/>	<xs:element
ref="contacts:MobilePhoneNumber"/>	<xs:element
ref="contacts:OfficeLocation"/>	<xs:element
ref="contacts:OtherAddressCity"/>	<xs:element
ref="contacts:OtherAddressCountry"/>	<xs:element
ref="contacts:OtherAddressPostalCode"/>	<xs:element
ref="contacts:OtherAddressState"/>	<xs:element
ref="contacts:OtherAddressStreet"/>	<xs:element
ref="contacts:PagerNumber"/>	<xs:element
ref="contacts:RadioPhoneNumber"/>	<xs:element
ref="contacts:Spouse"/>	<xs:element
ref="contacts:Suffix"/>	<xs:element
ref="contacts:Title"/>	<xs:element
ref="contacts:WebPage"/>	<xs:element
ref="contacts:YomiCompanyName"/>	<xs:element
ref="contacts:YomiFirstName"/>	<xs:element
ref="contacts:YomiLastName"/>	<xs:element
ref="contacts:Picture"/>	<xs:element

ref="contacts2:CustomerId"/>	<xs:element
ref="contacts2:GovernmentId"/>	<xs:element
ref="contacts2:IMAddress"/>	<xs:element
ref="contacts2:IMAddress2"/>	<xs:element
ref="contacts2:IMAddress3"/>	<xs:element
ref="contacts2:ManagerName"/>	<xs:element
ref="contacts2:CompanyMainPhone"/>	<xs:element
ref="contacts2:AccountName"/>	<xs:element
ref="contacts2:NickName"/>	<xs:element
ref="contacts2:MMS"/>	<xs:element
ref="tasks:Complete"/>	<xs:element
ref="tasks:Subject"/>	<xs:element
ref="tasks:Categories"/>	<xs:element
ref="tasks:DateCompleted"/>	<xs:element
ref="tasks:DueDate"/>	<xs:element
ref="tasks:UtcDueDate"/>	<xs:element
ref="tasks:Importance"/>	<xs:element
ref="tasks:Recurrence"/>	<xs:element
ref="tasks:ReminderSet"/>	<xs:element
ref="tasks:ReminderTime"/>	<xs:element
ref="tasks:Sensitivity"/>	<xs:element

```

ref="tasks:StartDate"/>
ref="tasks:UtcStartDate"/>
</xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0"
maxOccurs="unbounded" name="Fetch">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ServerId">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="64"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Wait" minOccurs="0" maxOccurs="1">
  <xs:simpleType>

```

```

        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="59"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="WindowSize" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="Partial" minOccurs="0" maxOccurs="1" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

2.2.1.19.1.1 Sync

The **Sync** element is the top-level element in the **XML** document. It identifies the body of the **HTTP** Post as containing a **Sync** command.

Parent elements	Child elements	Data type	Number allowed
None	<Limit> (response only) <Partial> (request only) <Wait> <WindowSize> (request only)	Container	0...1 (optional)

The **Sync** element can also include one or more explicit namespace attributes.

The **Limit** element and **Collections** element are mutually exclusive in a **Sync** response. That is, a **Sync** response can include either a **Limit** element or a **Collections** element, but not both. If the server response to a sync request with no body, the client can issue the next sync request

with no body as well to save bandwidth. The server will cache the last request if it issues an empty response, and if the next request is empty the server will use the cached request instead. If there exists any body, then it **MUST** contain a **Sync** element.

2.2.1.19.1.2 Wait

The **Wait** element specifies, in a request, the number of minutes that the server **SHOULD** delay a response and, in a response, the number of minutes that the server can wait for any changes before responding.

Parent elements	Child elements	Data type	Number allowed
<Sync>	None	Integer	0...1 (optional)

Valid values for **Wait** are 1 through 59. When the client requests a wait-interval that is outside the acceptable range the server will send a response that includes a **Status** value of 14 and a **Limit** element.

When **Wait** is used in a **Sync** request, the element indicates to the server that a response **SHOULD** be delayed either until the wait-interval, which is indicated by the contents of the **Wait** element, elapses or until any of the **collections** that are included in the request have changed.

It is at the discretion of the client to send the **Wait** element; the server is only guaranteed to respond immediately when **Wait** is not present. The client typically wants a server response immediately in the following cases:

- The client adds new items by using the **Add** element. In this case, an immediate response is required because the client requires the server-provided item ID to track changes to those new items.
- The client sends up a large change by using the **Change** element. In this case, a delayed response increases the possibility that the client has to resend the change because of a lost connection.

Although the server is only guaranteed to respond immediately when **Wait** is not present, the server **SHOULD** always respond immediately to a **Sync** request that includes an **Add** or a **Change**, unless the addition or change involves only flags.

A hard delete of tasks or calendar items will cause a waited **Sync** to finish. The benefit of this is a better user experience. For example, a user will not get **reminders** for deleted **meetings**. A hard delete is infrequent and rarely results in an extra roundtrip. A flagging change or a move out of (and not into) a folder which is being synced **SHOULD NOT** cause the request to finish early.

2.2.1.19.1.3 Partial

The **Partial** element indicates to the server that the client sent a partial list of **collections**, in which case the server **SHOULD** obtain the rest of the collections from its cache.

Parent elements	Child elements	Data type	Number allowed
<Sync> (request)	None	Empty	0...1 (optional)

The client **MUST NOT** send a **Partial** element without any other elements in the **Sync** request. A **Sync** request is valid with just a **Partial** element and either a **Wait** element, a **WindowSize** element, a **Collections** element, or any combination of the three. A **Sync** request requires, at least, either a **Partial** element or a **Collections** element.

When a request includes a **Partial** element but does not specify some collections, the settings and synchronization key for each of those unspecified collections specified in the previous **Sync** request remain the same as specified in the previous request. Such a request is equivalent to a request that specifies each of these collections with the same settings and synchronization key as in the previous request. This enables the client to modify some aspect of the previous request (one of the collections, the wait time, the global window size, etc) without sending up every unchanged collection.

2.2.1.19.1.4 WindowSize

The **WindowSize** element is sent from the client to the server to specify a maximum number of changed items in a **collection** or a request that **SHOULD** be included in the synchronization response.

Parent elements	Child elements	Data type	Number allowed
<Collection> (request) <Sync>	None	Integer	0...1 (optional)

The server sends the requested number of changes, and if there are more, the server includes a **MoreAvailable** element in the response. The maximum value for the **WindowSize** element is 512.

The **WindowSize** element appears only in requests that are sent to the server from the client. If **WindowSize** is omitted, the server behaves as if a **WindowSize** element with a value of 100 were submitted.

A good value for **WindowSize** is 100. Small values increase the load on the server, increase bandwidth, and decrease battery life because of the additional requests that are required to obtain all changes. Larger values cause larger responses, which are more susceptible to communication errors. A lower **WindowSize** value can be useful if the client can display the initial set of objects while additional ones are still being retrieved from the server.

If the window size is changed during a synchronization transaction, this can result in a **MoreAvailable** element being returned in the response but no items are returned. If this occurs, the client SHOULD synchronize again to continue getting items from the server.

The **WindowSize** element has been repurposed to also impose a global limit on the number of changes that are returned by the server. **WindowSize** can still be specified at the **Collection** level and the server MUST honor both settings.

When **WindowSize** is not sent by the client, the server assumes a default **WindowSize** of 100. This value is used by most clients and this default will save those clients some bytes. The default is also in effect at the **Collection** level. The maximum value for **WindowSize** is 512.

The collections are to be processed by the server in the order received, as follows:

- If the server has filled the **WindowSize** on a particular collection that has more changes, it will return the **MoreAvailable** element for that collection and continue to process the other collections until the global **WindowSize** has been filled.
- When the server has filled the global **WindowSize** and collections that have changes did not fit in the response, the server can return a **MoreAvailable** element.
- If a collection is not present in a **Sync** response, the client can assume that no changes are currently available for that collection.

The actual number of changes that are included in a **Sync** response for any particular collection depends on the **WindowSize** of the collection, the overall number of changes that are already included in the response, and the global **WindowSize**. The server will stop processing after the global **WindowSize** has been filled and simply not process the remaining collections. Any server-side changes that are pending in the unprocessed collections are picked up in the next synchronization.

The following synchronization request specifies that up to 100 changes be sent from the server back to the client. If there are more than 100 changes on the server, the **MoreAvailable** element is included in the response.

Request

```
<Collection>
  <Class>Email</Class>
  <SyncKey>1</SyncKey>
  <CollectionId>1</CollectionId>
  <DeletesAsMoves/>
  <GetChanges/>
  <WindowSize>100</WindowSize>
</Collection>
```

2.2.1.19.1.5 Add

The **Add** command can be used to create a new object in a **collection** on the client or on the server.

When a new item is being sent from the client to the server, the **ClientId** element specifies a temporary ID for the item, which is unique on the client. The **ApplicationData** element specifies the item data. The server then responds with an **Add** element in a **Responses** element, which specifies the client ID and the server ID that was assigned to the new item.

When the client sends a **Sync** command to the server and a new item has been added to the server collection since the last synchronization, the server responds with an **Add** element in a **Commands** element. This **Add** element specifies the server ID and data of the item to be added to the collection on the client.

When you add a calendar item, the **Timezone** property MUST be specified first and the **StartTime** and **EndTime** properties MUST be present in the **ApplicationData** element.

Parent elements	Child elements	Data type	Number allowed
<Commands> <Responses> (response only)	<ServerID/> (response only, see below) <ClientId/> <ApplicationData> <Status/> (response only)	Container	0...N (optional)

One or more **Add** elements can appear as a child of the **Commands** and **Responses** elements for a particular collection.

The **Add** element cannot be used to add any e-mail items from the client to the server.

If the server ID in an **Add** element from the server matches the server ID for an item on the client, the client SHOULD treat the addition as a change to the client item.

The server is not required to send an individual response for every command that is sent by the client. The client only receives responses for successful additions and fetches, and failed changes and deletions. When the client does not receive a response, the client SHOULD assume that the command succeeded unless informed otherwise.

2.2.1.19.1.6 ApplicationData

The **ElementName** element encloses data for a particular object, such as a **contact**, e-mail message, calendar **appointment**, or task item. The **ApplicationData** element can be used to add or change items on the client device or server. The format of this data is determined by the schema for the object.

Parent elements	Child elements	Data type	Number allowed
<Add> <Change>	Data elements from the content classes . For details about the content classes, see [MS-ASCAL], [MS-ASCNTC], [MS-ASDOC], [MS-ASEMAIL], and [MS-ASTASK].	Container	1 (required)

The following **ApplicationData** element is used to add a contact item, identified by the **ServerId** element, to a folder on the client device.

Response

```

<Add>
    <ServerId> 2:6</ServerId>
    <ApplicationData>
<A:Body></A:Body>
        <A:Email1Address>"jdobney@fourthcoffee.com"
        &lt;jdobney@fourthcoffee.com&gt;</A:Email1Address>
        <A:FileAs>Dobney, JoLynn Julie</A:FileAs>
        <A:FirstName>JoLynn</A:FirstName>
        <A:HomePhoneNumber>425 555 1234</A:HomePhoneNumber>
        <A:MiddleName>Julie</A:MiddleName>
        <A:MobilePhoneNumber>425 555 1111</A:MobilePhoneNumber>
        <A:CompanyName>Fourth Coffee</A:CompanyName>
        <A:LastName>Dobney</A:LastName>
        <A:BusinessPhoneNumber>425 555
5555</A:BusinessPhoneNumber>
        <A:JobTitle>Usability Engineer</A:JobTitle>
    </ApplicationData>
</Add>

```

2.2.1.19.1.7 Change

The **Change** element modifies properties of an existing object on the client device or the server. The object to change is identified by its **ServerId** element.

Parent elements	Child elements	Data type	Number allowed
<Commands> <Responses> (response only)	<ServerID/> <ApplicationData> <Status/> (response only)	Container	0...N (optional)

One or more **Change** elements can appear as a child of the **Commands** element for a particular **collection**.

Certain in-schema properties remain untouched in the following three cases:

- If there is only a **Flag** or **Read** change (that is, if only a **Flag** or **Read** node is present), all other properties will remain unchanged and nothing else has to be sent.
- If an **Exceptions** node is not specified, the properties for that **Exceptions** node will remain unchanged. If an **Exception** node within the **Exceptions** node is not present, that particular exception will remain unchanged.
- If **Body**, **Data**, **Picture**, or **RTF** nodes are not present, the corresponding properties will remain unchanged.

In all other cases, if an in-schema property is not specified in a change request, the property is actively deleted from the item on the server. A client **MUST** be aware of this when it is sending **Sync** requests; otherwise, data can be unintentionally removed.

2.2.1.19.1.8 ClientId

The **ClientId** is a unique identifier that is generated by the client to temporarily identify a new object that is being created by using the **Add** command. The client includes the **ClientId** element in the **Add** command request that it sends to the server. The server response contains an **Add** element that contains the original client ID and a new server ID that was assigned for the object, which replaces the client ID as the permanent object identifier.

Parent elements	Child elements	Data type	Number allowed
<Add>	None	String (Typically an integer)	Request: 1 (required)

The **ClientId** element is a unique identifier that consists of up to 40 digits and letters. The client generates this ID. The value only has to be unique for the device during the duration of the **Sync** request that adds the object to the server. The client **SHOULD** store the client IDs

until the synchronization session is completed successfully, to make recovery easier if the synchronization process fails.

An easy way to implement the client ID is to use a counter that is incremented for each new object that is created on the client.

2.2.1.19.1.9 Collection

The **Collection** element wraps commands and options that apply to a particular **collection**.

Parent elements	Child elements	Data type	Number allowed
<Collections>	<SyncKey/> <Supported> (request only) <CollectionId/> <DeletesAsMoves/> (request only) <GetChanges/> (request only) <WindowSize/> (request only) <Options> (request only) <Status/> (response only) <MoreAvailable/> (response only) <Commands> <Responses> (response only)	Container	1..512 (required)

The **Collection** element contains identification information (**Class**, **CollectionID**), synchronization state (**SyncKey**), commands (**GetChanges**, **Commands**), and options (**WindowSize**, **Options**, **DeleteAsMoves**, **MoreAvailable**). Only one collection can be specified in a **Sync** command.

There is a strict ordering of the **XML** elements within a **Collection** node in a **Sync** request. The order is as follows:

1. <Class>

2. <SyncKey>
3. <CollectionId>
4. <Supported>
5. <DeletesAsMoves>
6. <GetChanges>
7. <WindowSize>
8. <Options>
9. <Commands>

The **Collection** element appears in both **Sync** requests and responses. The form is similar, although some child elements are valid in only one context.

A single **Collections** element can contain multiple **Collection** elements. Therefore, each collection does not require its own **Sync** command. That is, a **Sync** request can specify multiple collections to be synchronized.

2.2.1.19.1.10 SyncKey

The **SyncKey** element contains a value that is used by the server to mark the synchronization state of a **collection**.

Parent elements	Child elements	Data type	Number allowed
<Collection>	None	String (Up to 64 characters)	1 (required)

A synchronization key of value 0 initializes the synchronization state on the server and causes a full synchronization of the collection. The server sends a response that includes a new synchronization key value. The client **MUST** store this synchronization key value until the client requires the key value for the next synchronization request for that collection. When the client uses this synchronization key value to do the next synchronization of the collection, the client sends this synchronization key value to the server in a **Sync** request. If the synchronization is successful, the server responds by sending all objects in the collection. The response includes a new synchronization key value that the client **MUST** use on the next synchronization of the collection.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

2.2.1.19.1.11 Supported

The **Supported** element is used to specify which **contact** and calendar properties are supported.

Parent elements	Child elements	Data type	Number allowed
-----------------	----------------	-----------	----------------

<Collection>	Any Contact or Calendar property. [Only container elements (Children , Categories) are valid. Their child elements (Child , Category) are not valid.]	Container	0...1 (optional)
--------------	--	------------------	------------------

The **Supported** element lists all properties that the client can manage. Properties that are not named are not changed when the client sends an update to the server.

The supported properties list is sent on the initial synchronization only; the server remembers the list for subsequent synchronizations.

The initial **Sync** request MUST include a **CollectionId** node, which MUST always precede the **Supported** node. See the **Collection** element (section 2.2.1.19.2.5) for the order of elements within the **Collection** node. This order is strictly enforced.

```
<Collection>
  <Supported>
    <c:FirstName/>
    <c:MiddleName/>
    <c:LastName/>
    <c:HomePhoneNumber/>
    <c:MobilePhoneNumber/>
    <c:BusinessPhoneNumber/>
    <c:Email1Address/>
  </Supported>
</Collection>
```

2.2.1.19.1.12 GetChanges

The **GetChanges** element requests the server to include in its response any pending changes to the **collection** that is specified by the **ServerId** element. If there have been changes since the last synchronization, the server response includes a **Commands** element that contains additions, deletions, and changes.

Parent elements	Child elements	Data type	Number allowed
<Collection>	None	Boolean	0...1 (optional)

(request only)			
----------------	--	--	--

The **GetChanges** element appears only in requests to the server from the client.

The server will not notify the client about changes in the calendar **Reminder**, **DtStamp**, **Email**, and **Name** properties in a **Sync** command response. If the client requests to receive the changes, the server will not return the calendar item if only these properties were changed. If any other property on the calendar item was changed, it is returned in the response.

If a calendar event is changed on both the server and the client, and the client has chosen the option to keep the server object in a conflict (**Conflict**=1), a conflict is only reported by the server if the **GetChanges** element is included in the body as false. If the **GetChanges** element is included in the request body as false, no conflict is reported, the changes on the client are applied to the server, and a response code of 1 is returned. If the **GetChanges** element is not included or is included with a value of true, the server correctly reports a status 7, and then returns the server version of the calendar event in the **Commands** element (section 2.2.1.19.2.8) of the response.

If the client does not want the server changes to be returned, the request **MUST** include the **GetChanges** element with a value of 0 (**FALSE**). A value of 1 (**TRUE**), which is the default, indicates that the client wants the server changes to be returned. The default is assumed when the **GetChanges** element is either empty or not present.

For requests with a **SyncKey** value of 0, the **GetChanges** element is a protocol error.

2.2.1.19.1.13 CollectionId

The **CollectionId** element specifies the server ID of the folder to be synchronized.

Parent elements	Child elements	Data type	Number allowed
<Collection>	None	String (Up to 64 characters)	1 (required)

The server ID of the folder is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command.

2.2.1.19.1.14 Collections

The **Collections** element serves as a container for the **Collection** element.

Parent elements	Child elements	Data type	Number allowed
<Sync>	<Collection>	Container	0...1 (optional)

The **Collections** element appears both in synchronization requests and responses. The structure is identical.

The **Collections** element is optional. If **Collections** is present, it can contain multiple **Collection** elements.

Request/Response

```
<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns="AirSync:">
  <Collections>
    <Collection>
      ...
    </Collection>
  </Collections>
```

2.2.1.19.1.15 Commands

The **Commands** element is a container for commands that apply to a **collection**. Available commands are **Add**, **Delete**, **Change**, and **Fetch**. Client commands are sent in the POST request; server commands are sent in the POST response.

This element is optional. If it is present, it MUST include at least one command. It is a child of the **Collection** element.

Parent elements	Child elements	Data type	Number allowed
<Collection>	<Add> <Delete> <Change> <Fetch>	Container	0...1 (optional)

The **Commands** element can appear in both **Sync** requests and responses.

Request/Response

```
<Collection>
  <Commands>
    <Add>
      ...
    </Add>
    <Delete>
      ...
    </Delete >
```

```

    <Change>
        ...
    </Change >
    <Fetch>
        ...
    </Fetch>
</Commands>
</Collection>

```

2.2.1.19.1.16 SoftDelete

The **SoftDelete** element deletes an object on the client due to being out of filter on the server. The object is identified by its **ServerId** element.

Parent elements	Child elements	Data type	Number allowed
<Commands>	<ServerId/>	Container	0...N (optional)

2.2.1.19.1.17 Fetch

The **Fetch** element is used to request the application data of an item that was truncated in a synchronization response from the server. The complete item is then returned to the client in a server response.

Parent elements	Child elements	Data type	Number allowed
<Commands> (request only) <Responses> (response only)	<ServerID/> <Status/> (response only) <ApplicationData> (response only)	Container	0...N (optional)

The **Fetch** element cannot be used to get truncated calendar, **contact**, or task items from the server.

2.2.1.19.1.18 DeletesAsMoves

The **DeletesAsMoves** element indicates that any deleted items SHOULD be moved to the Deleted Items folder.

Parent elements	Child elements	Data type	Number allowed
<Collection> (request only)	None	Boolean	0...1 (optional)

The **DeletesAsMoves** element appears only in requests to the server from the client. If the **DeleteAsMoves** element is set to false, the deletion is permanent.

If the client wants to permanently delete items, the request **MUST** include the **DeletesAsMoves** element with a value of 0 (**FALSE**). A value of 1 (**TRUE**), which is the default, indicates that any deleted items **SHOULD** be moved to the Deleted Items folder. The default is assumed when the **DeletesAsMoves** element is either empty or not present.

2.2.1.19.1.19 Options

The **Options** element is a container that encloses elements that control certain aspects of how the synchronization is performed.

Parent elements	Child elements	Data type	Number allowed
<Collection> (request only)	<FilterType/> <Conflict/> <MIMETruncation/> <MIMESupport/> <airsyncbase:BodyPreference>	Container	0...1 (optional)

This element is optional, but if it is present, it **MUST** include at least one child element. The **Options** element appears only in requests to the server from the client.

Additional synchronization options enable the client to specify truncation and content settings. These settings are encapsulated within a **BodyPreference** node within the **Options** element as follows:

```
<airsyncbase:BodyPreference>
  <airsyncbase:Type>1</Type>
  <airsyncbase:TruncationSize>512</TruncationSize>
  <airsyncbase:AllOrNone/>
</airsyncbase:BodyPreference>
```

Because synchronization options are specified on a **collection**, the client can specify a unique **BodyPreference** for each collection that it is being synchronized. For more details about the **BodyPreference** element, see [MS-ASAIRS] section 2.2.3.4.

The server preserves the options across requests. Therefore, the options **MUST** only be sent once per collection unless the client sends a **SyncKey** value of 0. Whenever the client specifies new options by including an **Options** node in the request, the server **MUST** replace the original options with the new options.

The following **Options** element specifies that items in the collection that are older than three days **SHOULD NOT** be returned to the client, that items **MUST** be truncated to 512 characters if they are larger, and that, if there are any item conflicts, the server **MUST** replace the client items.

Request

```
<Collection>
  <Options>
    <FilterType>2</FilterType>
    <Conflict>1</Conflict>
  </Options>
</Collection>
```

2.2.1.19.1.20 Conflict

The **Conflict** element specifies how to resolve the conflict that occurs when an object has been changed on both the client and the server. The value specifies which object—the client object or the server object—to keep if there is a conflict.

Parent elements	Child elements	Data type	Number allowed
<Options>	None	Integer	0...1 (optional)

The following table lists valid values for the element.

Value	Meaning
0	Client object replaces server object.
1	Server object replaces client object.

If the **Conflict** element is not present, the server object will replace the client object when a conflict occurs.

A value of 0 means to keep the client object; a value of 1 means to keep the server object. If the value is 1 and there is a conflict, a **Status** value of 7 is returned to inform the client that the object that the client sent to the server was discarded.

The **Conflict** element applies to the entire **collection**; therefore, it is not possible to use the element on an object-by-object basis in a single **Sync** command.

The **Conflict** element is a child of the **Options** element, and therefore the **Conflict** element appears only in requests to the server from the client.

If a **Delete** command conflicts with an **Add** or **Change** command, the **Delete** takes precedence.

2.2.1.19.1.21 FilterType

The **FilterType** element specifies an optional time window for the objects that are sent from the server to the client. It applies to e-mail and calendar **collections**. If **FilterType** is specified, the server sends only objects that are dated within the specified time window.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	0...1 (optional)

The following table lists valid values for the element.

Value	Meaning	E-mail?	Calendar?	Tasks?
0	No filter-synchronize all items	Yes	Yes	Yes
1	1 day back	Yes	No	No
2	3 days back	Yes	No	No
3	1 week back	Yes	No	No
4	2 weeks back	Yes	Yes	No
5	1 month back	Yes	Yes	No
6	3 months back	No	Yes	No
7	6 months back	No	Yes	No
8	Filter by incomplete tasks	No	No	Yes

When the **FilterType** element is specified, the server manages objects on the client to maintain the time window. New objects are added when they are within the time window. The server sends **SoftDelete** commands for objects on the client when they become older than the window.

Calendar items that are in the future or that have recurrence but no end date are sent to the client regardless of the **FilterType** element value.

The **FilterType** element is a child of the **Options** element. Therefore, it appears only in requests to the server from the client.

If the **FilterType** element is omitted, all objects are sent from the server without regard for their age.

Filters cannot be sent on **Contact** folders.

The following **Options** element in a synchronization request on an Inbox specifies that only e-mail messages that date back three days are returned to the client in the server synchronization response.

```
<Options>
  <FilterType>2</FilterType>
</Options>
```

2.2.1.19.1.22 MIMETruncation

The **MIMETruncation** element is included in the **Options** element of a client **Sync** command request to specify to the server whether the **MIME** data of the e-mail item SHOULD be truncated when it is sent from the server to the client.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	0...1 (optional)

The following table lists valid values for the element.

Value	Meaning
0	Truncate all body text.
1	Truncate text over 4,096 characters.
2	Truncate text over 5,120 characters.
3	Truncate text over 7,168 characters.

4	Truncate text over 10,240 characters.
5	Truncate text over 20,480 characters.
6	Truncate text over 51,200 characters.
7	Truncate text over 102,400 characters.
8	Do not truncate; send complete MIME data.

If the size of the MIME data exceeds the value that is specified by the client in the **MIMETruncation** element, the string that is returned in the **MIMEData** element is truncated up to the MIMETruncation value. The value of the **MIMESize** element will then contain the original size, in characters, of the MIME data. Note that for the FETCH case, the complete MIME data of the message is returned to the client regardless of any **MIMETruncation** option.

2.2.1.19.1.23 MIMESupport

The **MIMESupport** element is included in the **Options** element of a client **Sync** command request to enable **MIME** support for e-mail items that are sent from the server to the client.

Parent elements	Child elements	Data type	Number allowed
<Options> (request only)	None	Integer	0...1 (optional)

The following table shows valid values for the element.

Value	Meaning
0	Never send MIME data.
1	Send MIME data for S/MIME messages only. Send regular body for all other messages.
2	Send MIME data for all messages. This flag could be used by clients to build a more rich and complete Inbox solution.

The **Sync** request **MUST** include the following in the **Options** element:

- The **MIMESupport** element to tell the server to return MIME for **S/MIME-only/All/None** messages.

- The **BodyPreference** element with its child element, **Type**, which contains a value of 4 to inform the server that the device can read the MIME **BLOB**.

The response from the server **MUST** include the **Body** element, which is a child of the **ApplicationData** element. The **Body** element is a complex element and **MUST** contain the following child nodes in an S/MIME synchronization response:

- The **Type** element with a value of 4 to inform the device that the data is a MIME BLOB.
- The **EstimatedDataSize** element to specify the rough total size of the data.
- The **Truncated** element to indicate whether the MIME BLOB is truncated.
- The **Data** element that contains the full MIME BLOB.

For more details about the **Body** element or the **BodyPreference** element, see [MS-ASAIRS] sections 2.2.3.3 or 2.2.3.4, respectively.

For more details about the **ApplicationData.Body** element, see [MS-ASEMAIL] section 2.2.3.3.

2.2.1.19.2 Response

The following code shows the XSD for the **Sync** command response.

```
<?xml version="1.0" ?>

<xs:schema xmlns:tns="AirSync:" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="AirSync:"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:contacts="CONTACTS:"
xmlns:contacts2="CONTACTS2:" xmlns:calendar="CAL:" xmlns:email="EMAIL:"
xmlns:airsyncbase="AirSyncBase:" xmlns:tasks="TASKS:">

  <xs:import namespace="CONTACTS:" />
  <xs:import namespace="CONTACTS2:" />
  <xs:import namespace="EMAIL:" />
  <xs:import namespace="CAL:" />
  <xs:import namespace="AirSyncBase:" />
  <xs:import namespace="TASKS:" />
  <xs:element name="Sync" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" name="Status" type="xs:unsignedByte" />
        <xs:element minOccurs="0" name="Collections">
          <xs:complexType>
            <xs:sequence>
```

```

        <xs:element name="Collection" minOccurs="0"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:choice maxOccurs="unbounded">

<xs:element name="SyncKey" type="xs:string" />
                    <xs:element name="CollectionId" type="xs:string" />
                    <xs:element name="Status" type="xs:unsignedByte" />
                    <xs:element name="Commands">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element minOccurs="0" maxOccurs="unbounded"
name="Delete">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ServerId" type="xs:string" />
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="Change">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ServerId" type="xs:string" />
                                            <xs:element name="ApplicationData">
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:choice maxOccurs="unbounded">
                                                            <xs:element ref="calendar:Timezone" />
                                                            <xs:element ref="calendar:DtStamp" />
                                                            <xs:element ref="calendar:StartTime" />
                                                            <xs:element ref="calendar:Subject" />
                                                        </xs:choice>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element ref="calendar:UID" />
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>

```

```

        <xs:element ref="calendar:OrganizerName"
/>

        <xs:element ref="calendar:OrganizerEmail"
/>

        <xs:element ref="calendar:Location" />
        <xs:element ref="calendar:EndTime" />
        <xs:element ref="calendar:Recurrence" />
        <xs:element ref="calendar:Body" />
        <xs:element ref="calendar:Categories" />
        <xs:element ref="calendar:Sensitivity" />
        <xs:element ref="calendar:BusyStatus" />
        <xs:element ref="calendar:AllDayEvent" />
        <xs:element ref="calendar:Reminder" />
        <xs:element ref="calendar:Exceptions" />
        <xs:element ref="calendar:MeetingStatus"
/>

        <xs:element ref="calendar:Rtf" />
        <xs:element ref="calendar:Attendees" />
        <xs:element ref="contacts:Rtf" />
        <xs:element ref="contacts:Anniversary" />
        <xs:element ref="contacts:AssistantName"
/>

        <xs:element
ref="contacts:AssistnamePhoneNumber" />
        <xs:element ref="contacts:Birthday" />
        <xs:element ref="contacts:Body" />
        <xs:element ref="contacts:BodySize" />
        <xs:element ref="contacts:BodyTruncated"
/>

        <xs:element
ref="contacts:Business2PhoneNumber" />
        <xs:element ref="contacts:BusinessCity"
/>

        <xs:element
ref="contacts:BusinessCountry" />

```

```

        <xs:element
ref="contacts:BusinessPostalCode" />
        <xs:element ref="contacts:BusinessState"
/>
        <xs:element ref="contacts:BusinessStreet"
/>
        <xs:element
ref="contacts:BusinessFaxNumber" />
        <xs:element
ref="contacts:BusinessPhoneNumber" />
        <xs:element ref="contacts:CarPhoneNumber"
/>
        <xs:element ref="contacts:Categories" />
        <xs:element ref="contacts:Children" />
        <xs:element ref="contacts:CompanyName" />
        <xs:element ref="contacts:Department" />
        <xs:element ref="contacts:Email1Address"
/>
        <xs:element ref="contacts:Email2Address"
/>
        <xs:element ref="contacts:Email3Address"
/>
        <xs:element ref="contacts:FileAs" />
        <xs:element ref="contacts:FirstName" />
        <xs:element ref="contacts:MiddleName" />
        <xs:element
ref="contacts:Home2PhoneNumber" />
        <xs:element ref="contacts:HomeCity" />
        <xs:element ref="contacts:HomeCountry" />
        <xs:element ref="contacts:HomePostalCode"
/>
        <xs:element ref="contacts:HomeState" />
        <xs:element ref="contacts:HomeStreet" />
        <xs:element ref="contacts:HomeFaxNumber"
/>

```

```

        <xs:element
ref="contacts:HomePhoneNumber" />
        <xs:element ref="contacts:JobTitle" />
        <xs:element ref="contacts:LastName" />
        <xs:element
ref="contacts:MobilePhoneNumber" />
        <xs:element ref="contacts:OfficeLocation"
/>
        <xs:element ref="contacts:OtherCity" />
        <xs:element ref="contacts:OtherCountry"
/>
        <xs:element
ref="contacts:OtherPostalCode" />
        <xs:element ref="contacts:OtherState" />
        <xs:element ref="contacts:OtherStreet" />
        <xs:element ref="contacts:PagerNumber" />
        <xs:element ref="contacts:picture" />
        <xs:element
ref="contacts:RadioPhoneNumber" />
        <xs:element ref="contacts:Spouse" />
        <xs:element ref="contacts:Suffix" />
        <xs:element ref="contacts:Title" />
        <xs:element ref="contacts:WebPage" />
        <xs:element
ref="contacts:YomiCompanyName" />
        <xs:element ref="contacts:YomiFirstName"
/>
        <xs:element ref="contacts:YomiLastName"
/>
        <xs:element ref="contacts2:CustomerId" />
        <xs:element ref="contacts2:GovernmentId"
/>
        <xs:element ref="contacts2:IMAddress" />
        <xs:element ref="contacts2:IMAddress2" />
        <xs:element ref="contacts2:IMAddress3" />

```

```

        <xs:element ref="contacts2:ManagerName"
/>

        <xs:element
ref="contacts2:CompanyMainPhone" />
        <xs:element ref="contacts2:AccountName"
/>

        <xs:element ref="contacts2:NickName" />
        <xs:element ref="contacts2:MMS" />
        <xs:element ref="tasks:Body" />
        <xs:element ref="tasks:BodySize" />
        <xs:element ref="tasks:BodyTruncated" />
        <xs:element ref="tasks:Subject" />
        <xs:element ref="tasks:Categories" />
        <xs:element ref="tasks:Importance" />
        <xs:element ref="tasks:UtcStartDate" />
        <xs:element ref="tasks:StartDate" />
        <xs:element ref="tasks:UtcDueDate" />
        <xs:element ref="tasks:DueDate" />
        <xs:element ref="tasks:Recurrence" />
        <xs:element ref="tasks:Complete" />
        <xs:element ref="tasks:DateCompleted" />
        <xs:element ref="tasks:Sensitivity" />
        <xs:element ref="tasks:ReminderTime" />
        <xs:element ref="tasks:ReminderSet" />
        <xs:element ref="tasks:Rtf" />
    </xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0" maxOccurs="unbounded"
name="Add">

```

```

<xs:complexType>
<xs:sequence>
  <xs:element name="ServerId" type="xs:string" />
  <xs:element name="ApplicationData">
    <xs:complexType>
      <xs:sequence>
        <xs:choice maxOccurs="unbounded">
          <xs:element ref="email:To" />
          <xs:element ref="email:From" />
          <xs:element ref="email:Reply-To" />
          <xs:element ref="email:Subject" />
          <xs:element ref="email:DateReceived"/>
          <xs:element ref="email:DisplayTo" />
          <xs:element ref="email:ThreadTopic" />
          <xs:element ref="email:Importance" />
          <xs:element ref="email:Read" />
          <xs:element ref="email:Attachments" >
            <xs:complexType>
              <xs:sequence>
                <xs:element
ref="airsyncbase:Attachment">
                  <xs:complexType>
                    <xs:all>
<xs:element ref="airsyncbase:DisplayName" />
                    <xs:element
ref="airsyncbase:FileReference" />
                    <xs:element
ref="airsyncbase:Method" />
                    <xs:element
ref="airsyncbase:EstimatedDataSize" />
                    <xs:element
ref="airsyncbase:ContentId" />
                    <xs:element
ref="airsyncbase:ContentLocation" />

```



```

ref="airsynibase:IsInline" />
</xs:all>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element ref="email:BodyTruncated"/>
<xs:element ref="airsynibase:Body" >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="airsynibase:Type"
/>
      <xs:element
ref="airsynibase:EstimatedDataSize" />
      <xs:element
ref="airsynibase:Truncated" />
      <xs:element ref="airsynibase:Data"
/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
  <xs:element ref="email:MessageClass"/>
  <xs:element ref="email:AttRemoved" />
  <xs:element ref="email:MeetingRequest" />
  <xs:element ref="email:MIMETruncated" />
  <xs:element ref="email:MIMEData" />
  <xs:element ref="email:MIMESize" />
  <xs:element ref="email:MessageClass" />
  <xs:element ref="email:InternetCPID" />
  <xs:element ref="email:Flag" />
  <xs:complexType>
    <xs:sequence>

```

```

        <xs:element ref="tasks:Subject" ./>
        <xs:element ref="email:Status" />
        <xs:element ref="email:FlagType" />
        <xs:element ref="tasks:ReminderSet"
/>

        <xs:element
ref="tasks:ReminderTime" />

        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element ref="email:ContentClass" />
<xs:element
ref="airsynbase:NativeBodyType" />
    <xs:element ref="calendar:Timezone" />
    <xs:element ref="calendar:DtStamp" />
    <xs:element ref="calendar:StartTime"/>
    <xs:element ref="calendar:Subject" />
    <xs:element ref="calendar:UID" />
    <xs:element ref="calendar:OrganizerName"
/>

    <xs:element ref="calendar:OrganizerEmail"
/>

    <xs:element ref="calendar:Location" />
    <xs:element ref="calendar:EndTime" />
    <xs:element ref="calendar:Recurrence"/>
    <xs:element ref="calendar:Body" />
    <xs:element ref="calendar:Categories" />
    <xs:element ref="calendar:Sensitivity" />
    <xs:element ref="calendar:BusyStatus" />
    <xs:element ref="calendar:AllDayEvent" />
    <xs:element ref="calendar:Reminder" />
    <xs:element ref="calendar:Exceptions"/>
    <xs:element ref="calendar:MeetingStatus"
/>

```

```

        <xs:element ref="calendar:Rtf" />
        <xs:element ref="calendar:Attendees" />
        <xs:element ref="contacts:Rtf" />
        <xs:element ref="contacts:Anniversary"/>
        <xs:element ref="contacts:AssistantName"
/>

        <xs:element
ref="contacts:AssistnamePhoneNumber" />
        <xs:element ref="contacts:Birthday" />
        <xs:element ref="contacts:Body" />
        <xs:element ref="contacts:BodySize" />
        <xs:element ref="contacts:BodyTruncated"
/>

        <xs:element
ref="contacts:Business2PhoneNumber" />
        <xs:element ref="contacts:BusinessCity"
/>

        <xs:element
ref="contacts:BusinessCountry" />
        <xs:element
ref="contacts:BusinessPostalCode" />
        <xs:element ref="contacts:BusinessState"
/>

        <xs:element ref="contacts:BusinessStreet"
/>

        <xs:element
ref="contacts:BusinessFaxNumber" />
        <xs:element
ref="contacts:BusinessPhoneNumber" />
        <xs:element ref="contacts:CarPhoneNumber"
/>

        <xs:element ref="contacts:Categories"/>
        <xs:element ref="contacts:Children" />
        <xs:element ref="contacts:CompanyName"/>
        <xs:element ref="contacts:Department" />

```

```

        <xs:element ref="contacts:Email1Address"
/>
        <xs:element ref="contacts:Email2Address"
/>
        <xs:element ref="contacts:Email3Address"
/>
        <xs:element ref="contacts:FileAs" />
        <xs:element ref="contacts:FirstName" />
        <xs:element ref="contacts:MiddleName" />
        <xs:element
ref="contacts:Home2PhoneNumber" />
        <xs:element ref="contacts:HomeCity" />
        <xs:element ref="contacts:HomeCountry" />
        <xs:element ref="contacts:HomePostalCode"
/>
        <xs:element ref="contacts:HomeState" />
        <xs:element ref="contacts:HomeStreet"/>
        <xs:element ref="contacts:HomeFaxNumber"
/>
        <xs:element
ref="contacts:HomePhoneNumber" />
        <xs:element ref="contacts:JobTitle" />
        <xs:element ref="contacts:LastName" />
        <xs:element
ref="contacts:MobilePhoneNumber" />
        <xs:element ref="contacts:OfficeLocation"
/>
        <xs:element ref="contacts:OtherCity"/>
        <xs:element ref="contacts:OtherCountry"
/>
        <xs:element
ref="contacts:OtherPostalCode" />
        <xs:element ref="contacts:OtherState"/>
        <xs:element ref="contacts:OtherStreet" />
        <xs:element ref="contacts:PagerNumber"/>

```

```

        <xs:element ref="contacts:picture" />
        <xs:element
ref="contacts:RadioPhoneNumber" />
        <xs:element ref="contacts:Spouse" />
        <xs:element ref="contacts:Suffix" />
        <xs:element ref="contacts:Title" />
        <xs:element ref="contacts:WebPage" />
        <xs:element
ref="contacts:YomiCompanyName" />
        <xs:element ref="contacts:YomiFirstName"
/>
        <xs:element ref="contacts:YomiLastName"
/>
        <xs:element ref="contacts2:CustomerId" />
        <xs:element ref="contacts2:GovernmentId"
/>
        <xs:element ref="contacts2:IMAddress" />
        <xs:element ref="contacts2:IMAddress2" />
        <xs:element ref="contacts2:IMAddress3" />
        <xs:element ref="contacts2:ManagerName"
/>
        <xs:element
ref="contacts2:CompanyMainPhone" />
        <xs:element ref="contacts2:AccountName"
/>
        <xs:element ref="contacts2:NickName" />
        <xs:element ref="contacts2:MMS" />
        <xs:element ref="tasks:Body" />
        <xs:element ref="tasks:BodySize" />
        <xs:element ref="tasks:BodyTruncated" />
        <xs:element ref="tasks:Subject" />
        <xs:element ref="tasks:Categories" />
        <xs:element ref="tasks:Importance" />
        <xs:element ref="tasks:UtcStartDate" />

```

```

        <xs:element ref="tasks:StartDate" />
        <xs:element ref="tasks:UtcDueDate" />
        <xs:element ref="tasks:DueDate" />
        <xs:element ref="tasks:Recurrence" />
        <xs:element ref="tasks:Complete " />
        <xs:element ref="tasks:DateCompleted" />
        <xs:element ref="tasks:Sensitivity" />
        <xs:element ref="tasks:ReminderTime" />
        <xs:element ref="tasks:ReminderSet" />
        <xs:element ref="tasks:Rtf" />
    </xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Responses">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" name="Change">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="ServerId" type="xs:string"
/>
                            <xs:element name="Status"
type="xs:unsignedByte" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

```

```

        <xs:element minOccurs="0" maxOccurs="unbounded"
name="Add">

        <xs:complexType>
        <xs:sequence>
            <xs:element name="ClientId"
type="xs:unsignedByte" />
            <xs:element name="ServerId" type="xs:string"
/>

            <xs:element name="Status"
type="xs:unsignedByte" />
        </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Fetch">
        <xs:complexType>
        <xs:sequence>
            <xs:element name="ServerId"
type="xs:string"></xs:element>
            <xs:element name="Status"
type="xs:unsignedByte"></xs:element>
            <xs:element name="ApplicationData">
                <xs:complexType>
                <xs:sequence>
                    <xs:choice maxOccurs="unbounded">
                        <xs:element ref="email:To" />
                        <xs:element ref="email:From" />
                        <xs:element ref="email:Reply-To" />
                        <xs:element ref="email:Subject" />
                        <xs:element ref="email:DateReceived" />
                        <xs:element ref="email:DisplayTo" />
                        <xs:element ref="email:ThreadTopic" />
                        <xs:element ref="email:Importance" />
                        <xs:element ref="email:Read" />
                        <xs:element ref="email:Attachments" />
                    </xs:choice>
                </xs:sequence>
            </xs:element>
        </xs:complexType>
    </xs:element>

```

```

        <xs:complexType>
            <xs:sequence>
                <xs:element
ref="airsynibase:Attachment">
                    <xs:complexType>
                        <xs:all>
<xs:element ref="airsynibase:DisplayName" />
                            <xs:element
ref="airsynibase:FileReference" />
                                <xs:element
ref="airsynibase:Method" />
                                    <xs:element
ref="airsynibase:EstimatedDataSize" />
                                        <xs:element
ref="airsynibase:ContentId" />
                                            <xs:element
ref="airsynibase:ContentLocation" />
                                                <xs:element
ref="airsynibase:IsInline" />
                                                    </xs:all>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            <xs:element ref="email:BodyTruncated" />
                                <xs:element ref="airsynibase:Body" >
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element ref="airsynibase:Type"
/>
                                                <xs:element
ref="airsynibase:EstimatedDataSize" />
                                                    <xs:element
ref="airsynibase:Truncated" />

```



```

        <xs:element ref="airsynibase:Data"
/>

    </xs:sequence>
</xs:complexType>
</xs:element>

    <xs:element ref="email:MessageClass" />
    <xs:element ref="email:AttRemoved" />
    <xs:element ref="email:MeetingRequest" />
    <xs:element ref="email:InternetCPID" />
    <xs:element ref="email:Flag" />
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="tasks:Subject" ./>
            <xs:element ref="email:Status" />
            <xs:element ref="email:FlagType" />
            <xs:element ref="tasks:ReminderSet"
/>

        <xs:element
ref="tasks:ReminderTime" />

        </xs:sequence>
    </xs:complexType>
</xs:element>

    <xs:element ref="email:ContentClass" />
    <xs:element
ref="airsynibase:NativeBodyType" />
    </xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

```

```

        </xs:element>
        <xs:element name="MoreAvailable"></xs:element>
    </xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:element name="Limit" type="xs:integer" />
</xs:element>
</xs:schema>

```

2.2.1.19.2.1 Add

The **Add** command can be used to create a new object in a **collection** on the client or on the server.

When a new item is being sent from the client to the server, the **ClientId** element specifies a temporary ID for the item, which is unique on the client. The **ApplicationData** element specifies the item data. The server then responds with an **Add** element in a **Responses** element, which specifies the client ID and the server ID that was assigned to the new item.

When the client sends a **Sync** command to the server and a new item has been added to the server collection since the last synchronization, the server responds with an **Add** element in a **Commands** element. This **Add** element specifies the server ID and data of the item to be added to the collection on the client.

Parent elements	Child elements	Data type	Number allowed
<Commands> <Responses> (response only)	<ServerID/> (response only, see below) <ClientId/> <ApplicationData> <Status/> (response only)	Container	0...N (optional)

One or more **Add** elements can appear as a child of the **Commands** and **Responses** elements for a particular collection.

If the server ID in an **Add** element from the server matches the server ID for an item on the client, the client **SHOULD** treat the addition as a change to the client item.

The server is not required to send an individual response for every command that is sent by the client. The client only receives responses for successful additions and fetches, and failed changes and deletions. When the client does not receive a response, the client **SHOULD** assume that the command succeeded unless informed otherwise.

2.2.1.19.2.2 ApplicationData

The **ElementName** element encloses data for a particular object, such as a **contact**, e-mail message, calendar **appointment**, or task item. The **ApplicationData** element can be used to add or change items on the client device or server. The format of this data is determined by the schema for the object.

Parent elements	Child elements	Data type	Number allowed
<Add> <Change>	Data elements from the content classes . For details about the content classes, see [MS-ASCAL], [MS-ASCNTC], [MS-ASDOC], [MS-ASEMAIL], and [MS-ASTASK].	Container	1 (required)

The following **ApplicationData** element is used to add a contact item, identified by the **ServerId** element, to a folder on the client device.

Response

```
<Add>
  <ServerId> 2:6</ServerId>
  <ApplicationData>
    <A:Body></A:Body>
    <A:Email1Address>"jdobney@fourthcoffee.com"
    &lt;jdobney@fourthcoffee.com&gt;</A:Email1Address>
    <A:FileAs>Dobney, JoLynn Julie</A:FileAs>
    <A:FirstName>JoLynn</A:FirstName>
    <A:HomePhoneNumber>425 555 1234</A:HomePhoneNumber>
```

```

        <A:MiddleName>Julie</A:MiddleName>
        <A:MobilePhoneNumber>425 555 1111</A:MobilePhoneNumber>
        <A:CompanyName>Fourth Coffee</A:CompanyName>
        <A:LastName>Dobney</A:LastName>
        <A:BusinessPhoneNumber>425 555
5555</A:BusinessPhoneNumber>
        <A:JobTitle>Usability Engineer</A:JobTitle>
    </ApplicationData>
</Add>

```

2.2.1.19.2.3 Change

The **Change** element modifies properties of an existing object on the client device or the server. The object to change is identified by its **ServerId** element.

Parent elements	Child elements	Data type	Number allowed
<Commands> <Responses> (response only)	<ServerID/> <ApplicationData> <Status/> (response only)	Container	0...N (optional)

One or more **Change** elements can appear as a child of the **Commands** element for a particular **collection**.

Certain in-schema properties remain untouched in the following three cases:

- If there is only a **Flag** or **Read** change (that is, if only a **Flag** or **Read** node is present), all other properties will remain unchanged and nothing else has to be sent.
- If an **Exceptions** node is not specified, the properties for that **Exceptions** node will remain unchanged. If an **Exception** node within the **Exceptions** node is not present, that particular exception will remain unchanged.
- If **Body**, **Data**, **Picture**, or **RTF** nodes are not present, the corresponding properties will remain unchanged.

In all other cases, if an in-schema property is not specified in a change request, the property is actively deleted from the item on the server. A client **MUST** be aware of this when it is sending **Sync** requests; otherwise, data can be unintentionally removed.

2.2.1.19.2.4 ClientId

The **ClientId** is a unique identifier that is generated by the client to temporarily identify a new object that is being created by using the **Add** command. The client includes the **ClientId** element in the **Add** command request that it sends to the server. The server response contains an **Add** element that contains the original client ID and a new server ID that was assigned for the object, which replaces the client ID as the permanent object identifier.

Parent elements	Child elements	Data type	Number allowed
<Add>	None	String (Typically an integer)	Request: 1 (required) Response: 1

The **ClientId** element is a unique identifier that consists of up to 40 digits and letters. The client generates this ID. The value only has to be unique for the device during the duration of the **Sync** request that adds the object to the server. The client **SHOULD** store the client IDs until the synchronization session is completed successfully, to make recovery easier if the synchronization process fails.

An easy way to implement the client ID is to use a counter that is incremented for each new object that is created on the client.

2.2.1.19.2.5 Collection

The **Collection** element wraps commands and options that apply to a particular **collection**.

Parent elements	Child elements	Data type	Number allowed
<Collections>	<SyncKey/> <Supported> (request only) <CollectionId/> <DeletesAsMoves/> (request only) <GetChanges/> (request only) <WindowSize/> (request only) <Options> (request only) <Status/> (response only)	Container	1..512 (required)

	<MoreAvailable/> (response only) <Commands> <Responses> (response only)		
--	---	--	--

The **Collection** element contains identification information (**Class**, **CollectionID**), synchronization state (**SyncKey**), commands (**GetChanges**, **Commands**), and options (**WindowSize**, **Options**, **DeleteAsMoves**, **MoreAvailable**). Only one collection can be specified in a **Sync** command.

There is a strict ordering of the **XML** elements within a **Collection** node in a **Sync** request. The order is as follows:

- <Class>
- <SyncKey>
- <CollectionId>
- <Supported>
- <DeletesAsMoves>
- <GetChanges>
- <WindowSize>
- <Options>
- <Commands>

The **Collection** element appears in both **Sync** requests and responses. The form is similar, although some child elements are valid in only one context.

A single **Collections** element can contain multiple **Collection** elements. Therefore, each **collection** does not require its own **Sync** command. That is, a **Sync** request can specify multiple collections to be synchronized.

2.2.1.19.2.6 CollectionId

The **CollectionId** element specifies the server ID of the folder to be synchronized.

Parent elements	Child elements	Data type	Number allowed
<Collection>	None	String (Up to 64 characters)	1 (required)

The server ID of the folder is obtained from the **ServerId** element of a previous **FolderSync** or **FolderCreate** command.

2.2.1.19.2.7 Collections

The **Collections** element serves as a container for the **Collection** element.

Parent elements	Child elements	Data type	Number allowed
<Sync>	<Collection>	Container	0...1 (optional)

The **Collections** element appears both in synchronization requests and responses. The structure is identical.

The **Collections** element is optional. If **Collections** is present, it can contain multiple **Collection** elements, but MUST contain at least one.

Request/Response

```
<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns="AirSync:">
  <Collections>
    <Collection>
      ...
    </Collection>
  </Collections>
```

2.2.1.19.2.8 Commands

The **Commands** element is a container for commands that apply to a **collection**. Available commands are **Add**, **Delete**, **Change**, and **Fetch**. Client commands are sent in the POST request; server commands are sent in the POST response.

This element is optional. If it is present, it MUST include at least one command. It is a child of the **Collection** element.

Parent elements	Child elements	Data type	Number allowed
<Collection>	<Add> <Delete> <Change> <Fetch>	Container	0...1 (optional)

The **Commands** element can appear in both **Sync** requests and responses.

Request/Response

```
<Collection>
```

```

<Commands>
    <Add>
        ...
    </Add>
    <Delete>
        ...
    </Delete >
    <Change>
        ...
    </Change >
    <Fetch>
        ...
    </Fetch>
</Commands>
</Collection>

```

2.2.1.19.2.9 SoftDelete

The **SoftDelete** element deletes an object. The object is identified by its **ServerId** element.

Parent elements	Child elements	Data type	Number allowed
<Commands>	<ServerId/>	Container	0...N (optional)

2.2.1.19.2.10 Fetch

The **Fetch** element is used to request the application data of an item that was truncated in a synchronization response from the server. The complete item is then returned to the client in a server response.

The **ItemOperations** command (section 2.2.1.8) is the preferred way to fetch items.

Parent elements	Child elements	Data type	Number allowed
<Commands> (request only) <Responses> (response only)	<ServerID/> <Status/> (response only) <ApplicationData> (response only)	Container	0...N (optional)

The **Fetch** element cannot be used to get truncated calendar, **contact**, or task items from the server.

2.2.1.19.2.11 Limit

The **Limit** element specifies either the maximum number of **collections** that can be synchronized or the maximum/minimum value that is allowed for the wait-interval.

Parent elements	Child elements	Data type	Number allowed
<Sync> (response only)	None	Integer	0...1 (optional)

The **Limit** element is returned in a response with a status code of 14 or 15. The value of the **Status** element indicates whether the limit applies to the wait-interval or the number of collections, as follows:

- A status code 14 indicates that **Limit** specifies the minimum or maximum wait-interval that is acceptable. When the value of the **Wait** element is outside the acceptable range, the server responds with the closest acceptable value. The server will send a response that includes a status code 14 and a **Limit** element.
- A status code 15 indicates that **Limit** specifies the maximum number of collections that can be synchronized.

2.2.1.19.2.12 MessageClass

The **MessageClass** element is returned by the server in the **Add** element of a **Sync** command response and specifies the message type.

Parent elements	Child elements	Data type	Number allowed
<Add> (response only)	None	String	0...1 (optional)

The following table lists some values for the element.

Value	Meaning
IPM.Note.SMIME	The message is encrypted and can also be signed.
IPM.Note.SMIME.MultipartSigned	The message is clear signed.
IPM.Note.Receipt.SMIME	The message is a secure read receipt .

The following table lists some values for the element.

Value	Meaning
REPORT.IPM.Note.SMIME.DR	Delivery report for a S/MIME message.
REPORT.IPM.Note.SMIME.IPNRN	Read report for an S/MIME message.
REPORT.IPM.Note.SMIME.IPNNRN	Non-read report for an S/MIME message.
REPORT.IPM.Note.SMIME.MultipartSigned.NDR	Non-delivery report for a signed S/MIME message.
REPORT.IPM.Note.SMIME.MultipartSigned.DR	Delivery report for a signed S/MIME message.
REPORT.IPM.Note.SMIME.MultipartSigned.IPNRN	Read report for a signed S/MIME message.
REPORT.IPM.Note.SMIME.MultipartSigned.IPNNRN	Non-read report for a signed S/MIME message.

2.2.1.19.2.13 MIMEData

The **MIMEData** element is returned by the server in the **ApplicationData** element of a **Sync** command response and contains the raw **MIME** data of an e-mail item.

Parent elements	Child elements	Data type	Number allowed
<ApplicationData> (response only)	None	String	1 (required)

This element **MUST** be returned by the server if the client enabled MIME support, and **MUST NOT** be returned if the client has not enabled MIME support.

2.2.1.19.2.14 MIMESize

The **MIMESize** element specifies the complete size, in characters, of the MIME message that is contained in the **MIMEData** element.

Parent elements	Child elements	Data type	Number allowed
<ApplicationData>	None	Integer	0...1 (optional)

(response only)			
-----------------	--	--	--

The **MIMESize** element is returned by the server in the **ApplicationData** element of a **Sync** command response. This element **MUST** be returned by the server if the client enabled **MIME** support, and **MUST NOT** be returned if the client has not enabled MIME support.

2.2.1.19.2.15 MIMETruncated

The **MIMETruncated** element indicates whether the **MIMEDData** element contains a truncated string.

Parent elements	Child elements	Data type	Number allowed
<ApplicationData> (response only)	None	Integer	0...1 (optional)

The **MIMETruncated** element is returned by the server in the **ApplicationData** element of a **Sync** command response. This element **MUST** be returned by the server if the client enabled **MIME** support, and **MUST NOT** be returned if the client has not enabled MIME support.

A value of zero indicates that the data has not been truncated; a nonzero value indicates otherwise.

2.2.1.19.2.16 MoreAvailable

The **MoreAvailable** element is included in a synchronization response from the server to the client if there are more changes than the number that are requested in the **WindowSize** element.

Parent elements	Child elements	Data type	Number allowed
<Collection> (response only)	None	Empty	0...1 (optional)

The **MoreAvailable** element appears only in responses that are sent from the server to the client. It appears only if the client request contained a **WindowSize** element and there are still changes to be returned to the client.

The **MoreAvailable** element has no body. It is omitted if no additional changes are available. The maximum value for the **WindowSize** element is 512.

If the **WindowSize** element is omitted, the server behaves as if a **WindowSize** element with a value of 100 was submitted. The **MoreAvailable** element is returned by the server if there are more than 512 changes, regardless of whether the **WindowSize** element is included in the request.

2.2.1.19.2.17 Responses

The **Responses** element contains responses to commands that are processed by the server. Each response is wrapped in an element with the same name as the command, such as **Add** and **Delete**. The response contains a status code and other information, depending on the command.

Parent elements	Child elements	Data type	Number allowed
<Collection> (responses)	<Add>, <Fetch> (If the command succeeded.) <Change>, <Delete> (If the command failed.)	Container	0...1 (optional)

The **Responses** element appears only in responses that are sent from the server to the client. It is present only if the server has processed commands from the client. It is omitted otherwise (for example, if the client requested server changes but had no changes to send to the server). If present, it **MUST** include at least one child element.

The following **Responses** element is part of a server response to a synchronization request. It shows items in the server **collection** that have been added, deleted, changed, or fetched.

Response

```
<Collection>
  <Responses>
    <Add>
      ...
    </Add>
    <Change>
      ...
    </Change >
  </Responses>
</Collection>
```

2.2.1.19.2.18 ServerId

The **ServerId** is a unique identifier that is assigned by the server to each object that can be synchronized. The client **MUST** store the server ID for each object and **MUST** be able to locate an object given a server ID. In a synchronization request, commands such as **Change** and **Delete** identify objects by using their server IDs.

Parent elements	Child elements	Data type	Number allowed
<Add> (response only) <Delete> <Change> <Fetch> (response only)	None	String (Up to 64 characters)	1 (required)

The client **MUST** store the server ID as an opaque string of up to 64 characters.

2.2.1.19.2.19 Status

The **Status** element indicates the success or failure of a command. If the command failed, the **Status** element contains a code that indicates the type of failure. The values are summarized in the following table.

Parent elements	Child elements	Data type	Number allowed
<Sync> (response only) <Collection> (response only) <Add> (response only) <Delete> (response only) <Change> (response only) <Fetch> (response only)	None	Integer	1 (required)

The following table lists valid values for the element.

Value	Meaning
1	Success.
2	Protocol version mismatch.

3	Invalid synchronization key.
4	Protocol error.
5	Server error.
6	Error in client/server conversion.
7	Conflict matching the client and server object.
8	Object not found.
9	User account could be out of disk space.
10	An error occurred while setting the notification GUID .
11	The device has not been provisioned for notifications yet.
12	The folder hierarchy has changed.
13	The client sent an empty or partial Sync request, but the server is unable to process it. Please resend the request with the full XML
14	The Sync request was processed successfully but the wait-interval that is specified by the client is outside the range set by the server administrator. If the wait-interval is too great, the response contains a Limit element that specifies the maximum allowed value. If the wait-interval is too low, the response contains a Limit element that specifies the minimum allowed value.
15	The Sync request was processed successfully, but specified more folders to monitor for changes than is allowed by the limit configured by the server administrator. The response includes the Limit element, which specifies the maximum number of folders that can be synchronized.
16	Please retry the same request.

The **Status** element is sent only in responses from the server to the client.

If a status code is not returned for a command, the client SHOULD assume success.

2.2.1.19.2.20 Sync

The **Sync** element is the top-level element in the **XML** document. It identifies the body of the **HTTP** Post as containing a **Sync** command.

Parent elements	Child elements	Data type	Number allowed
None	<Limit> (response only) <Partial> (request only) <Wait> <WindowSize> (request only)	Container	0...1 (optional)

The **Sync** element can also include one or more explicit namespace attributes.

The **Limit** element and **Collections** element are mutually exclusive in a **Sync** response. That is, a **Sync** response can include either a **Limit** element or a **Collections** element, but not both.

If an **XML** body exists, then the **Sync** element is required.

2.2.1.19.2.21 SyncKey

The **SyncKey** element contains a value that is used by the server to represent the synchronization state of a **collection**.

Parent elements	Child elements	Data type	Number allowed
<Collection>	None	String (Up to 64 characters)	1 (required)

A synchronization key of value 0 initializes the synchronization state on the server and causes a full synchronization of the **collection**. The server sends a response that includes a new synchronization key value. The client **MUST** store this synchronization key value until the client requires the key value for the next synchronization request for that collection. When the client uses this synchronization key value to do the next synchronization of the collection, the client sends this synchronization key value to the server in a **Sync** request. If the synchronization is successful, the server responds by sending all objects in the collection. The response includes a new synchronization key value that the client uses on the next synchronization of the collection.

The client **MUST** store the synchronization key as an opaque string of up to 64 characters.

The client always sends a synchronization key value of 0 in an initial **Sync** request and the server sends a new synchronization key value in its response to the client. The client

MUST NOT ignore the synchronization key value that is included in the initial response from the server.

2.2.1.19.2.22 Wait

The **Wait** element specifies, in a request, the number of minutes that the server SHOULD delay a response and, in a response, the number of minutes that the server can wait for any changes before responding.

Parent elements	Child elements	Data type	Number allowed
<Sync>	None	Integer	0...1 (optional)

Valid values for **Wait** are 1 through 59. When the client requests a wait-interval that is outside the acceptable range, the server MUST respond with the wait-interval that is the closest acceptable value. The server MUST send a response that includes a **Status** value of 14 and a **Limit** element.

When **Wait** is used in a **Sync** request, the element indicates to the server that a response SHOULD be delayed either until the wait-interval, which is indicated by the contents of the **Wait** element, elapses or until any of the **collections** that are included in the request have changed.

It is at the discretion of the client to send the **Wait** element; the server is only guaranteed to respond immediately when **Wait** is not present. The client typically wants a server response immediately in the following cases:

- The client adds new items by using the **Add** element. In this case, an immediate response is required because the client requires the server-provided item ID to track changes to those new items.
- The client sends up a large change by using the **Change** element. In this case, a delayed response increases the possibility that the client has to resend the change because of a lost connection.

Although the server is only guaranteed to respond immediately when **Wait** is not present, the server SHOULD always respond immediately to a **Sync** request that includes an **Add** or a **Change**, unless the addition or change involves only flags.

A hard delete of tasks or calendar items will cause a waited **Sync** to finish. The benefit of this is a better user experience. For example, a user will not get **reminders** for deleted **meetings**. A hard delete is infrequent and rarely results in an extra roundtrip. Readflag changes and moves out of (and not into) a folder which is being synchronized will cause the server to batch up the notification.

2.2.1.19.3 Content Class Specific XSDs

The following sections contain the XSDs for content class specific **Sync** command requests and responses.

2.2.1.19.3.1 Sync Command for Calendar Items

2.2.1.19.3.1.1 Sync Command Request for Calendar Items

```
<?xml version="1.0" ?>

<xs:schema xmlns:calendar="Calendar:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:airsyncbase="AirSyncBase:" targetNamespace="Calendar:"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="Calendar:">

  <xs:import namespace="AirSyncBase:"/>

  <xs:element name="TimeZone" type="xs:string" />

  <xs:element name="AllDayEvent" type="xs:unsignedByte" />

  <xs:element name="BusyStatus">
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="5"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="OrganizerName" type="xs:string" />
  <xs:element name="OrganizerEmail" type="xs:string" />
  <xs:element name="DtStamp" type="xs:string" />
  <xs:element name="EndTime" type="xs:string" />
  <xs:element name="Location" type="xs:string" />
  <xs:element name="Reminder" type="xs:unsignedInt" />
  <xs:element name="Sensitivity">
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="3"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="Subject" type="xs:string" />
```

```

<xs:element name="StartTime" type="xs:string" />
<xs:element name="UID">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="300"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="MeetingStatus">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedByte">
      <xs:enumeration value="1"/>
      <xs:enumeration value="0"/>
      <xs:enumeration value="3"/>
      <xs:enumeration value="5"/>
      <xs:enumeration value="7"/>
      <xs:enumeration value="9"/>
      <xs:enumeration value="11"/>
      <xs:enumeration value="13"/>
      <xs:enumeration value="15"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="Attendees">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element name="Attendee" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>
            <xs:element name="Email" type="xs:string" />
            <xs:element name="Name" type="xs:string" />
            <xs:element name="AttendeeStatus" minOccurs="0">
              <xs:simpleType>

```

```

        <xs:restriction base="xs:unsignedByte">
            <xs:enumeration value="0"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
            <xs:enumeration value="4"/>
            <xs:enumeration value="5"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="AttendeeType" minOccurs="0">
    <xs:simpleType>
        <xs:restriction base="xs:unsignedByte">
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
</xs:all>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Categories">
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element maxOccurs="300" name="Category" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="Recurrence">
    <xs:complexType>

```

```

<xs:all minOccurs="0">
  <xs:element minOccurs="1" name="Type">
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="6"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element minOccurs="0" name="Occurrences"
type="xs:unsignedShort" />
  <xs:element minOccurs="0" name="Interval">
    <xs:simpleType>
      <xs:restriction base="xs:unsignedShort">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="999"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element minOccurs="0" name="WeekOfMonth">
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="5"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element minOccurs="0" name="DayOfWeek">
    <xs:simpleType>
      <xs:restriction base="xs:unsignedShort">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="127"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

```

```

        </xs:simpleType>
    </xs:element>
    <xs:element minOccurs="0" name="MonthOfYear">
        <xs:simpleType>
            <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="1"/>
                <xs:maxInclusive value="12"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element minOccurs="0" name="Until" type="xs:string" />
    <xs:element minOccurs="0" name="DayOfMonth">
        <xs:simpleType>
            <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="1"/>
                <xs:maxInclusive value="127"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
</xs:all>
</xs:complexType>
</xs:element>
<xs:element name="Exceptions">
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element name="Exception" maxOccurs="1000">
                <xs:complexType>
                    <xs:all>
                        <xs:element minOccurs="0" name="Deleted"
type="xs:unsignedByte" />
                        <xs:element name="ExceptionStartTime" type="xs:string" />
                        <xs:element minOccurs="0" name="Subject" type="xs:string"
/>

```

```

        <xs:element minOccurs="0" name="StartTime"
type="xs:string" />
        <xs:element minOccurs="0" name="EndTime" type="xs:string"
/>

        <xs:element minOccurs="0" ref="airsyncbase:Body" />
        <xs:element minOccurs="0" name="Location"
type="xs:string" />
        <xs:element minOccurs="0" name="Categories">
            <xs:complexType>
                <xs:sequence>
                    <xs:element maxOccurs="300" name="Category"
type="xs:string" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element minOccurs="0" name="Sensitivity">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedByte">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="3"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element minOccurs="0" name="BusyStatus">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedByte">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="5"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element minOccurs="0" name="AllDayEvent"
type="xs:unsignedByte" />

```

```

        <xs:element minOccurs="0" name="Reminder"
type="xs:unsignedInt" />
        <xs:element minOccurs="0" name="DtStamp" type="xs:string"
/>

        <xs:element minOccurs="0" name="MeetingStatus">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedByte">
                    <xs:enumeration value="1"/>
                    <xs:enumeration value="0"/>
                    <xs:enumeration value="3"/>
                    <xs:enumeration value="5"/>
                    <xs:enumeration value="7"/>
                    <xs:enumeration value="9"/>
                    <xs:enumeration value="11"/>
                    <xs:enumeration value="13"/>
                    <xs:enumeration value="15"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:all>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="EmptyTag"/>
<xs:group name="GhostingProps">
    <xs:sequence>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="TimeZone" type="calendar:EmptyTag" />
            <xs:element name="AllDayEvent" type="calendar:EmptyTag" />
            <xs:element name="BusyStatus" type="calendar:EmptyTag" />
            <xs:element name="OrganizerName" type="calendar:EmptyTag" />
        </xs:choice>
    </xs:sequence>
</xs:group>

```

```

    <xs:element name="OrganizerEmail" type="calendar:EmptyTag" />
    <xs:element name="DtStamp" type="calendar:EmptyTag" />
    <xs:element name="EndTime" type="calendar:EmptyTag" />
    <xs:element name="Location" type="calendar:EmptyTag" />
    <xs:element name="Reminder" type="calendar:EmptyTag" />
    <xs:element name="Sensitivity" type="calendar:EmptyTag" />
    <xs:element name="Subject" type="calendar:EmptyTag" />
    <xs:element name="StartTime" type="calendar:EmptyTag" />
    <xs:element name="UID" type="calendar:EmptyTag" />
    <xs:element name="MeetingStatus" type="calendar:EmptyTag" />
    <xs:element name="Attendees" type="calendar:EmptyTag" />
    <xs:element name="Categories" type="calendar:EmptyTag" />
    <xs:element name="Recurrence" type="calendar:EmptyTag" />
    <xs:element name="Exceptions" type="calendar:EmptyTag" />
  </xs:choice>
</xs:sequence>
</xs:group>
<xs:group name="TopLevelSchemaProps">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="TimeZone" type="calendar:EmptyTag" />
      <xs:element name="StartTime" type="calendar:EmptyTag" />
      <xs:element name="EndTime" type="calendar:EmptyTag" />
      <xs:element name="Subject" type="calendar:EmptyTag" />
      <xs:element name="Location" type="calendar:EmptyTag" />
      <xs:element name="Reminder" type="calendar:EmptyTag" />
      <xs:element name="AllDayEvent" type="calendar:EmptyTag" />
      <xs:element name="BusyStatus" type="calendar:EmptyTag" />
      <xs:element name="Recurrence" type="calendar:EmptyTag" />
      <xs:element name="Sensitivity" type="calendar:EmptyTag" />
      <xs:element name="DtStamp" type="calendar:EmptyTag" />
      <xs:element name="Attendees" type="calendar:EmptyTag" />
      <xs:element name="Categories" type="calendar:EmptyTag" />
    </xs:choice>
  </xs:sequence>
</xs:group>

```



```

    <xs:element name="MeetingStatus" type="calendar:EmptyTag" />
    <xs:element name="OrganizerName" type="calendar:EmptyTag" />
    <xs:element name="OrganizerEmail" type="calendar:EmptyTag" />
    <xs:element name="UID" type="calendar:EmptyTag" />
    <xs:element name="Exceptions" type="calendar:EmptyTag" />
  </xs:choice>
</xs:sequence>
</xs:group>
</xs:schema>

```

2.2.1.19.3.1.2 Sync Command Response for Calendar Items

For the complete **Sync** command response, see section 2.2.1.19.2.

2.2.1.19.3.2 Sync Command for Contacts Folder

2.2.1.19.3.2.1 Sync Command Request for Contacts

```

<?xml version="1.0" ?>
<xs:schema xmlns:contacts="Contacts:"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="Contacts:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Anniversary" type="xs:dateTime" />
  <xs:element name="AssistantName" type="xs:string" />
  <xs:element name="AssistantPhoneNumber" type="xs:string" />
  <xs:element name="AssistnamePhoneNumber" type="xs:string" />
  <xs:element name="Birthday" type="xs:dateTime" />
  <xs:element name="Business2PhoneNumber" type="xs:string" />
  <xs:element name="BusinessAddressCity" type="xs:string" />
  <xs:element name="BusinessPhoneNumber" type="xs:string" />
  <xs:element name="WebPage" type="xs:string" />
  <xs:element name="BusinessAddressCountry" type="xs:string" />
  <xs:element name="Department" type="xs:string" />
  <xs:element name="Email1Address" type="xs:string" />
  <xs:element name="Email2Address" type="xs:string" />
  <xs:element name="Email3Address" type="xs:string" />

```

```

<xs:element name="BusinessFaxNumber" type="xs:string" />
<xs:element name="FileAs" type="xs:string" />
<xs:element name="FirstName" type="xs:string" />
<xs:element name="MiddleName" type="xs:string" />
<xs:element name="HomeAddressCity" type="xs:string" />
<xs:element name="HomeAddressCountry" type="xs:string" />
<xs:element name="HomeFaxNumber" type="xs:string" />
<xs:element name="HomePhoneNumber" type="xs:string" />
<xs:element name="Home2PhoneNumber" type="xs:string" />
<xs:element name="HomeAddressPostalCode" type="xs:string" />
<xs:element name="HomeAddressState" type="xs:string" />
<xs:element name="HomeAddressStreet" type="xs:string" />
<xs:element name="MobilePhoneNumber" type="xs:string" />
<xs:element name="Suffix" type="xs:string" />
<xs:element name="CompanyName" type="xs:string" />
<xs:element name="OtherAddressCity" type="xs:string" />
<xs:element name="OtherAddressCountry" type="xs:string" />
<xs:element name="CarPhoneNumber" type="xs:string" />
<xs:element name="OtherAddressPostalCode" type="xs:string" />
<xs:element name="OtherAddressState" type="xs:string" />
<xs:element name="OtherAddressStreet" type="xs:string" />
<xs:element name="PagerNumber" type="xs:string" />
<xs:element name="Title" type="xs:string" />
<xs:element name="BusinessAddressPostalCode" type="xs:string" />
<xs:element name="LastName" type="xs:string" />
<xs:element name="Spouse" type="xs:string" />
<xs:element name="BusinessAddressState" type="xs:string" />
<xs:element name="BusinessAddressStreet" type="xs:string" />
<xs:element name="JobTitle" type="xs:string" />
<xs:element name="YomiFirstName" type="xs:string" />
<xs:element name="YomiLastName" type="xs:string" />
<xs:element name="YomiCompanyName" type="xs:string" />
<xs:element name="OfficeLocation" type="xs:string" />

```

```

<xs:element name="RadioPhoneNumber" type="xs:string" />
<xs:element name="Picture" type="xs:string" />
<xs:element name="Categories">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element maxOccurs="300" name="Category" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Children">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element maxOccurs="300" name="Child" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="EmptyTag"/>
<xs:group name="GhostingProps">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="Anniversary" type="contacts:EmptyTag"/>
      <xs:element name="Birthday" type="contacts:EmptyTag"/>
      <xs:element name="WebPage" type="contacts:EmptyTag"/>
      <xs:element name="Children" type="contacts:EmptyTag"/>
      <xs:element name="BusinessAddressCountry"
type="contacts:EmptyTag"/>
      <xs:element name="Department" type="contacts:EmptyTag"/>
      <xs:element name="Email1Address" type="contacts:EmptyTag"/>
      <xs:element name="Email2Address" type="contacts:EmptyTag"/>
      <xs:element name="Email3Address" type="contacts:EmptyTag"/>
      <xs:element name="BusinessFaxNumber" type="contacts:EmptyTag"/>
      <xs:element name="FileAs" type="contacts:EmptyTag"/>
      <xs:element name="FirstName" type="contacts:EmptyTag"/>
    </xs:choice>
  </xs:sequence>
</xs:group>

```

```

        <xs:element name="HomeAddressCity" type="contacts:EmptyTag"/>
        <xs:element name="HomeAddressCountry"
type="contacts:EmptyTag"/>
        <xs:element name="HomeFaxNumber" type="contacts:EmptyTag"/>
        <xs:element name="HomePhoneNumber" type="contacts:EmptyTag"/>
        <xs:element name="Home2PhoneNumber" type="contacts:EmptyTag"/>
        <xs:element name="HomeAddressPostalCode"
type="contacts:EmptyTag"/>
        <xs:element name="HomeAddressState" type="contacts:EmptyTag"/>
        <xs:element name="HomeAddressStreet" type="contacts:EmptyTag"/>
        <xs:element name="BusinessAddressCity"
type="contacts:EmptyTag"/>
        <xs:element name="MiddleName" type="contacts:EmptyTag"/>
        <xs:element name="MobilePhoneNumber" type="contacts:EmptyTag"/>
        <xs:element name="Suffix" type="contacts:EmptyTag"/>
        <xs:element name="CompanyName" type="contacts:EmptyTag"/>
        <xs:element name="OtherAddressCity" type="contacts:EmptyTag"/>
        <xs:element name="OtherAddressCountry"
type="contacts:EmptyTag"/>
        <xs:element name="CarPhoneNumber" type="contacts:EmptyTag"/>
        <xs:element name="OtherAddressPostalCode"
type="contacts:EmptyTag"/>
        <xs:element name="OtherAddressState" type="contacts:EmptyTag"/>
        <xs:element name="OtherAddressStreet"
type="contacts:EmptyTag"/>
        <xs:element name="PagerNumber" type="contacts:EmptyTag"/>
        <xs:element name="Title" type="contacts:EmptyTag"/>
        <xs:element name="BusinessAddressPostalCode"
type="contacts:EmptyTag"/>
        <xs:element name="AssistantName" type="contacts:EmptyTag"/>
        <xs:element name="AssistantPhoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="AssistnamePhoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="LastName" type="contacts:EmptyTag"/>

```

```

        <xs:element name="Spouse" type="contacts:EmptyTag"/>
        <xs:element name="BusinessAddressState"
type="contacts:EmptyTag"/>
        <xs:element name="BusinessAddressStreet"
type="contacts:EmptyTag"/>
        <xs:element name="BusinessPhoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="Business2PhoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="JobTitle" type="contacts:EmptyTag"/>
        <xs:element name="YomiFirstName" type="contacts:EmptyTag"/>
        <xs:element name="YomiLastName" type="contacts:EmptyTag"/>
        <xs:element name="YomiCompanyName" type="contacts:EmptyTag"/>
        <xs:element name="OfficeLocation" type="contacts:EmptyTag"/>
        <xs:element name="RadioPhoneNumber" type="contacts:EmptyTag"/>
        <xs:element name="Picture" type="contacts:EmptyTag"/>
        <xs:element name="Categories" type="contacts:EmptyTag"/>
    </xs:choice>
</xs:sequence>
</xs:group>
<xs:group name="TopLevelSchemaProps">
    <xs:sequence>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="Anniversary" type="contacts:EmptyTag"/>
            <xs:element name="Birthday" type="contacts:EmptyTag"/>
            <xs:element name="Webpage" type="contacts:EmptyTag"/>
            <xs:element name="Children" type="contacts:EmptyTag"/>
            <xs:element name="BusinessAddressCountry"
type="contacts:EmptyTag"/>
            <xs:element name="Department" type="contacts:EmptyTag"/>
            <xs:element name="Email1Address" type="contacts:EmptyTag"/>
            <xs:element name="Email2Address" type="contacts:EmptyTag"/>
            <xs:element name="Email3Address" type="contacts:EmptyTag"/>
            <xs:element name="BusinessFaxNumber" type="contacts:EmptyTag"/>

```

```

    <xs:element name="FileAs" type="contacts:EmptyTag"/>
    <xs:element name="FirstName" type="contacts:EmptyTag"/>
    <xs:element name="HomeAddressCity" type="contacts:EmptyTag"/>
    <xs:element name="HomeAddressCountry"
type="contacts:EmptyTag"/>
    <xs:element name="HomeFaxNumber" type="contacts:EmptyTag"/>
    <xs:element name="HomeTelephoneNumber"
type="contacts:EmptyTag"/>
    <xs:element name="Home2TelephoneNumber"
type="contacts:EmptyTag"/>
    <xs:element name="HomeAddressPostalCode"
type="contacts:EmptyTag"/>
    <xs:element name="HomeAddressState" type="contacts:EmptyTag"/>
    <xs:element name="HomeAddressStreet" type="contacts:EmptyTag"/>
    <xs:element name="BusinessAddressCity"
type="contacts:EmptyTag"/>
    <xs:element name="MiddleName" type="contacts:EmptyTag"/>
    <xs:element name="MobileTelephoneNumber"
type="contacts:EmptyTag"/>
    <xs:element name="Suffix" type="contacts:EmptyTag"/>
    <xs:element name="CompanyName" type="contacts:EmptyTag"/>
    <xs:element name="OtherAddressCity" type="contacts:EmptyTag"/>
    <xs:element name="OtherAddressCountry"
type="contacts:EmptyTag"/>
    <xs:element name="CarTelephoneNumber"
type="contacts:EmptyTag"/>
    <xs:element name="OtherAddressPostalCode"
type="contacts:EmptyTag"/>
    <xs:element name="OtherAddressState" type="contacts:EmptyTag"/>
    <xs:element name="OtherAddressStreet"
type="contacts:EmptyTag"/>
    <xs:element name="PagerNumber" type="contacts:EmptyTag"/>
    <xs:element name="Title" type="contacts:EmptyTag"/>
    <xs:element name="BusinessAddressPostalCode"
type="contacts:EmptyTag"/>

```

```

        <xs:element name="AssistantName" type="contacts:EmptyTag"/>
        <xs:element name="AssistantTelephoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="LastName" type="contacts:EmptyTag"/>
        <xs:element name="Spouse" type="contacts:EmptyTag"/>
        <xs:element name="BusinessAddressState"
type="contacts:EmptyTag"/>
        <xs:element name="BusinessAddressStreet"
type="contacts:EmptyTag"/>
        <xs:element name="BusinessTelephoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="Business2TelephoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="JobTitle" type="contacts:EmptyTag"/>
        <xs:element name="YomiFirstName" type="contacts:EmptyTag"/>
        <xs:element name="YomiLastName" type="contacts:EmptyTag"/>
        <xs:element name="YomiCompanyName" type="contacts:EmptyTag"/>
        <xs:element name="OfficeLocation" type="contacts:EmptyTag"/>
        <xs:element name="RadioTelephoneNumber"
type="contacts:EmptyTag"/>
        <xs:element name="Categories" type="contacts:EmptyTag"/>
        <xs:element name="Picture" type="contacts:EmptyTag"/>
    </xs:choice>
</xs:sequence>
</xs:group>
</xs:schema>

```

2.2.1.19.3.2.2 Sync Command Response for Contacts

For the complete **Sync** command response, see section 2.2.1.19.2.

2.2.1.19.3.3 Sync Command for Contacts2 Folder

2.2.1.19.3.3.1 Sync Command Request for Contacts2

```

<?xml version="1.0" ?>
<xs:schema xmlns:contacts2="Contacts2:"
attributeFormDefault="unqualified" elementFormDefault="qualified"

```

```

targetNamespace="Contacts2:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CustomerId" type="xs:string" />
  <xs:element name="GovernmentId" type="xs:string" />
  <xs:element name="IMAddress" type="xs:string" />
  <xs:element name="IMAddress2" type="xs:string" />
  <xs:element name="IMAddress3" type="xs:string" />
  <xs:element name="ManagerName" type="xs:string" />
  <xs:element name="CompanyMainPhone" type="xs:string" />
  <xs:element name="AccountName" type="xs:string" />
  <xs:element name="NickName" type="xs:string" />
  <xs:element name="MMS" type="xs:string" />
  <xs:complexType name="EmptyTag"/>
  <xs:group name="GhostingProps">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="CustomerId" type="contacts2:EmptyTag"/>
        <xs:element name="GovernmentId" type="contacts2:EmptyTag"/>
        <xs:element name="IMAddress" type="contacts2:EmptyTag"/>
        <xs:element name="IMAddress2" type="contacts2:EmptyTag"/>
        <xs:element name="IMAddress3" type="contacts2:EmptyTag"/>
        <xs:element name="ManagerName" type="contacts2:EmptyTag"/>
        <xs:element name="CompanyMainPhone" type="contacts2:EmptyTag"/>
        <xs:element name="AccountName" type="contacts2:EmptyTag"/>
        <xs:element name="NickName" type="contacts2:EmptyTag"/>
        <xs:element name="MMS" type="contacts2:EmptyTag"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>
  <xs:group name="TopLevelSchemaProps">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="CustomerId" type="contacts2:EmptyTag"/>

```



```

    <xs:element name="GovernmentId" type="contacts2:EmptyTag"/>
    <xs:element name="IMAddress" type="contacts2:EmptyTag"/>
    <xs:element name="IMAddress2" type="contacts2:EmptyTag"/>
    <xs:element name="IMAddress3" type="contacts2:EmptyTag"/>
    <xs:element name="ManagerName" type="contacts2:EmptyTag"/>
    <xs:element name="CompanyMainPhone" type="contacts2:EmptyTag"/>
    <xs:element name="AccountName" type="contacts2:EmptyTag"/>
    <xs:element name="NickName" type="contacts2:EmptyTag"/>
    <xs:element name="MMS" type="contacts2:EmptyTag"/>
  </xs:choice>
</xs:sequence>
</xs:group>
</xs:schema>

```

2.2.1.19.3.3.2 Sync Command Response for Contacts2

For the complete **Sync** command response, see section 2.2.1.19.2.

2.2.1.19.3.4 Sync Command for E-Mail Folder

2.2.1.19.3.4.1 Sync Command Request for E-Mail

```

<?xml version="1.0" ?>
<xs:schema xmlns:email="Email:" attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="Email:" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tasks="Tasks:">
  <xs:import namespace="Tasks:"/>
  <xs:element name="Read" type="xs:unsignedByte" />
  <xs:element name="DateReceived" type="email:EmptyTag"/>
  <xs:element name="Flag">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Status" maxOccurs="1"
type="xs:unsignedByte" />
        <xs:element name="FlagType" maxOccurs="1" type="xs:string" />

```

```

        <xs:element name="CompleteTime" maxOccurs="1"
type="xs:string" />
        <xs:element ref="tasks:StartDate" maxOccurs="1"/>
        <xs:element ref="tasks:UtcStartDate" maxOccurs="1"/>
        <xs:element ref="tasks:DueDate" maxOccurs="1"/>
        <xs:element ref="tasks:UtcDueDate" maxOccurs="1"/>
        <xs:element ref="tasks:DateCompleted"
maxOccurs="1"/>
        <xs:element ref="tasks:ReminderSet"
maxOccurs="1"/>
        <xs:element ref="tasks:ReminderTime"
maxOccurs="1"/>
        <xs:element ref="tasks:Subject" maxOccurs="1"/>
        <xs:element ref="tasks:OrdinalDate" maxOccurs="1"/>
        <xs:element ref="tasks:SubOrdinalDate" maxOccurs="1"/>
    </xs:choice>
</xs:complexType>
</xs:element>
<xs:complexType name="EmptyTag"/>
<xs:group name="TopLevelSchemaProps">
    <xs:sequence>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="To" type="email:EmptyTag"/>
            <xs:element name="CC" type="email:EmptyTag"/>
            <xs:element name="From" type="email:EmptyTag"/>
            <xs:element name="ReplyTo" type="email:EmptyTag"/>
            <xs:element name="DateReceived" type="email:EmptyTag"/>
            <xs:element name="Subject" type="email:EmptyTag"/>
            <xs:element name="DisplayTo" type="email:EmptyTag"/>
            <xs:element name="Importance" type="email:EmptyTag"/>
            <xs:element name="Read" type="email:EmptyTag"/>
            <xs:element name="MessageClass" type="email:EmptyTag"/>
            <xs:element name="MeetingRequest" type="email:EmptyTag"/>
            <xs:element name="ThreadTopic" type="email:EmptyTag"/>

```

```

        <xs:element name="InternetCPID" type="email:EmptyTag"/>
    </xs:choice>
</xs:sequence>
</xs:group>
</xs:schema>

```

2.2.1.19.3.4.2 Sync Command Response for E-Mail

For the complete Sync command response, see section 2.2.1.19.2.

2.2.1.19.3.5 Sync Command for Tasks Folder

2.2.1.19.3.5.1 Sync Command Request for Tasks

```

<?xml version="1.0" ?>
<xs:schema xmlns:email="Email:" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="Email:" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tasks="Tasks:">
    <xs:import namespace="Tasks:"/>
    <xs:element name="Read" type="xs:unsignedByte" />
    <xs:element name="DateReceived" type="email:EmptyTag"/>
    <xs:element name="Flag">
        <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="Status" maxOccurs="1"
type="xs:unsignedByte" />
                <xs:element name="FlagType" maxOccurs="1" type="xs:string" />
                <xs:element name="CompleteTime" maxOccurs="1"
type="xs:string" />
                <xs:element ref="tasks:StartDate" maxOccurs="1"/>
                <xs:element ref="tasks:UtcStartDate" maxOccurs="1"/>
                <xs:element ref="tasks:DueDate" maxOccurs="1"/>
                <xs:element ref="tasks:UtcDueDate" maxOccurs="1"/>
                <xs:element ref="tasks:DateCompleted"
maxOccurs="1"/>
                <xs:element ref="tasks:ReminderSet"
maxOccurs="1"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>

```

```

        <xs:element ref="tasks:ReminderTime"
maxOccurs="1"/>
        <xs:element ref="tasks:Subject" maxOccurs="1"/>
        <xs:element ref="tasks:OrdinalDate" maxOccurs="1"/>
        <xs:element ref="tasks:SubOrdinalDate" maxOccurs="1"/>
    </xs:choice>
</xs:complexType>
</xs:element>
<xs:complexType name="EmptyTag"/>
<xs:group name="TopLevelSchemaProps">
    <xs:sequence>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="To" type="email:EmptyTag"/>
            <xs:element name="CC" type="email:EmptyTag"/>
            <xs:element name="From" type="email:EmptyTag"/>
            <xs:element name="ReplyTo" type="email:EmptyTag"/>
            <xs:element name="DateReceived" type="email:EmptyTag"/>
            <xs:element name="Subject" type="email:EmptyTag"/>
            <xs:element name="DisplayTo" type="email:EmptyTag"/>
            <xs:element name="Importance" type="email:EmptyTag"/>
            <xs:element name="Read" type="email:EmptyTag"/>
            <xs:element name="MessageClass" type="email:EmptyTag"/>
            <xs:element name="MeetingRequest" type="email:EmptyTag"/>
            <xs:element name="ThreadTopic" type="email:EmptyTag"/>
            <xs:element name="InternetCPID" type="email:EmptyTag"/>
        </xs:choice>
    </xs:sequence>
</xs:group>
</xs:schema>

```

2.2.1.19.3.5.2 Sync Command Response for Tasks

For the complete **Sync** command response, see section 2.2.1.19.2.

2.2.1.20 ValidateCert

The **ValidateCert** command is used by the client to validate a certificate that has been received via an **S/MIME** mail.

To validate a certificate, the server **MUST** verify that the certificate has not expired and has not been revoked. The server **MUST** walk up the certificate chain, verifying that each intermediate CA certificate has not expired and has not been revoked and that the root certificate is a trusted certificate authority. Certificate validation is particularly important for verifying signatures (for example, on S/MIME signed mail).

2.2.1.20.1 Request

The following code shows the XSD for the **ValidateCert** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="ValidateCert:" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="ValidateCert:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="ValidateCert">
  <xs:complexType>
    <xs:all minOccurs="0" maxOccurs="1">
      <xs:element name="CertificateChain"
minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:choice maxOccurs="unbounded">
            <xs:element
name="Certificate" minOccurs="1" maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction
base="xs:base64Binary">
                  <xs:minLength value="4"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:all>
  </xs:complexType>
</xs:element>
```

```

        <xs:element name="Certificates" minOccurs="1"
maxOccurs="1">
            <xs:complexType>
                <xs:choice maxOccurs="unbounded">
                    <xs:element
name="Certificate" minOccurs="1" maxOccurs="unbounded">
                        <xs:simpleType>
                            <xs:restriction
base="xs:base64Binary">
                                <xs:minLength value="4"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="CheckCrl" minOccurs="0"
maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:integer">
                        <xs:minInclusive value="0"/>
                        <xs:maxInclusive value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:all>
    </xs:complexType></xs:element>

```

2.2.1.20.1.1 ValidateCert

The **ValidateCert** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **ValidateCert** command.

Parent elements	Child elements	Data type	Number allowed
None	<Certificates>, <CertificateChain>, <CheckCRL>	Container	1 (required)

	(request only) <Status> (response only)		
--	--	--	--

2.2.1.20.1.2 Certificate

The **Certificate** element contains the Base64-encoded x509 certificate **BLOB**.

Parent elements	Child elements	Data type	Number allowed
<Certificates>, <CertificateChain>	<Status> (response only)	String (Base64-encoded)	1...N

2.2.1.20.1.3 CertificateChain

The **CertificateChain** element contains the list of certificates to be validated.

Parent elements	Child elements	Data type	Number allowed
<ValidateCert> (request only)	<Certificate>	Container	1 (required)

2.2.1.20.1.4 Certificates

The **Certificates** element contains the list of certificates to be validated.

Parent elements	Child elements	Data type	Number allowed
<ValidateCert> (request only)	<Certificate>	Container	1 (required)

2.2.1.20.1.5 CheckCRL

The **CheckCRL** element specifies whether the server SHOULD ignore an unverifiable revocation status.

Parent elements	Child elements	Data type	Number allowed
<ValidateCert>	None	Integer	0...1 (optional)

The revocation status of a certificate cannot be verified when the certificate revocation lists (CRLs) cannot be retrieved.

When **CheckCRL** is set to 1 (**TRUE**), the server MUST NOT ignore an unverifiable revocation status. The default value is 0.

2.2.1.20.2 Response

The following code shows the XSD for the **ValidateCert** command response.

```
<?xml version="1.0" ?>
<xs:schema xmlns:tns="ValidateCert:" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="ValidateCert:"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="ValidateCert">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Status"
type="xs:unsignedByte" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="Certificate">
  <xs:element minOccurs="1" maxOccurs="1" name="Status"
type="xs:unsignedByte" />
</xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.1.20.2.1 ValidateCert

The **ValidateCert** element is the top-level element in the **XML** document. It identifies the body of the **HTTP Post** as containing a **ValidateCert** command.

Parent elements	Child elements	Data type	Number allowed
None	<Certificates>, <CertificateChain>, <CheckCRL> (request only) <Status> (response only)	Container	1 (required)

2.2.1.20.2.2 Certificate

The **Certificate** element contains the Base64-encoded x509 certificate **BLOB**.

Parent elements	Child elements	Data type	Number allowed
<Certificates>, <CertificateChain>	<Status> (response only)	String (Base64-encoded)	1...N

2.2.1.20.2.3 Status

The **Status** element indicates whether one or more certificates were successfully validated.

Parent elements	Child elements	Data type	Number allowed
<ValidateCert>, <Certificate> (response only)	None	Integer	1...N (required)

The following table shows valid values for the element.

Value	Meaning
1	Successful validation.
2	Protocol error.
3	The signature in the digital ID can't be validated.
4	The digital ID was issued by an untrusted source.
5	The certificate chain that contains the digital ID was not created properly.
6	The digital ID is not valid for signing e-mail messages.
7	The digital ID used to sign the message has expired or is not yet valid.
8	The time periods during which the digital IDs in the certificate chain are not consistent.
9	A certificate is being used for a purpose other than what it was specified for.
10	Information associated with the digital ID is missing or incorrect.
11	A certificate that can only be used as an end-entity is being used as a certification authority (CA), or a CA that can only be used as an end-entity is being used as a

	certificate.
12	The digital ID doesn't match the recipient's e-mail address.
13	The digital ID used to sign this message has been revoked. This can indicate that the issuer of the digital ID no longer trusts the sender, the digital ID was reported stolen, or the digital ID was compromised.
14	The validity of the digital ID can't be determined because the server that provides this information can't be contacted.
15	A digital ID in the chain has been revoked by the authority that issued it.
16	The digital ID can't be validated because its revocation status can't be determined.
17	An unknown server error has occurred.

2.2.2 Status Codes

This section specifies the codes that the server returns in the **Status** element of each command response. For details about processing the return codes, see section 3.1.5.5.

Each status code has a **Scope** assigned to it. The following table lists the **Scope** values.

Scope Value	Description
Global	The status pertains to the overall client request.
Item	The status pertains to a particular item within the overall client request.
Policy	The status pertains to a particular policy within the Provision command.

2.2.2.1 FolderCreate Status Codes

The following table lists the status codes for the **FolderCreate** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.

2	A folder that has this name already exists.	The parent folder already contains a folder that has this name.	Item	Prompt user to supply a unique name.
5	The specified parent folder was not found.	The parent folder does not exist on the server,, possibly because it has been deleted or renamed.	Item	Issue a FolderSync command for the new hierarchy and prompt the user for a new parent folder.
6	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the FolderSync command. If continued attempts to synchronization fail, consider returning to synchronization key zero (0).
7	Access denied.	The user does not have permission to access the mailbox .	Global	Refer user to mail administrator.
8	The request timed out.	The server took too long to respond to the request.	Global	Retry.
9	Synchronization key mismatch or invalid synchronization key.	The client sent a malformed or mismatched synchronization key, or the synchronization state is corrupted on the server.	Global	Delete folders added since last synchronization and return to synchronization key to zero (0).

10	Malformed request.	The client sent a FolderCreate command that contains a semantic error.	Global	Fix bug in client code. Double-check the request for accuracy.
12	Code unknown.	Unusual back-end issue.	Global	No solution.

2.2.2.2 FolderDelete Status Codes

The following table lists the status codes for the **FolderDelete** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.
3	The specified folder is a special folder ; for example, Inbox , Outbox , Contacts , and so on.	The client specified a special folder in a FolderDelete command request. Special folders cannot be deleted.	Item	None.
4	The specified folder does not exist.	The client specified a nonexistent folder in a FolderDelete command request.	Item	Issue a FolderSync command for the new hierarchy.
6	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient	Global	Retry the FolderDelete command. If continued attempts to synchronization fail, consider

		condition.		returning to synchronization key zero (0).
7	Access denied.	The user does not have permission to access the mailbox .	Global	Refer user to mail administrator.
8	The request timed out.	The server took too long to respond to the request.	Global	Retry.
9	Synchronization key mismatch or invalid synchronization key.	The client sent a malformed or mismatched synchronization key, or the synchronization state is corrupted on the server.	Global	Issue a FolderSync command request with a synchronization key of zero (0).
10	Malformed request.	The client sent a FolderCreate command request that contains a semantic or syntactic error.	Global	Fix bug in client code. Double-check the request for accuracy.
11	Code Unknown.	Unusual back-end issue.	Global	No solution.

2.2.2.3 FolderSync Status Codes

The following table lists the status codes for the **FolderSync** command.

Code	Meaning	Cause	Scope	Resolution
1	Success	Server successfully completed command.	Global	None.

6	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the FolderSync command. If continued attempts to synchronization fail, consider returning to synchronization key zero (0).
7	Access denied.	The user does not have permission to access the mailbox .	Global	Refer user to mail administrator.
8	The request timed out.	The server took too long to respond to the request.	Global	Retry.
9	Synchronization key mismatch or invalid synchronization key.	The client sent a malformed or mismatched synchronization key, or the synchronization state is corrupted on the server.	Global	Delete items added since last synchronization and return to synchronization key zero (0).
10	Malformed request.	The client sent a FolderSync command request that contains a semantic or syntactic error.	Global	Fix bug in client code. Double-check the request for accuracy.
11	An unknown error occurred.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient	Global	Retry the FolderSync command request. If continued attempts to

		condition.		synchronization fail, consider returning to synchronization key zero (0).
12	Code Unknown.	Unusual back-end issue.	Global	No solution.

2.2.2.4 FolderUpdate Status Codes

The following table lists the status codes for the **FolderUpdate** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.
3	The specified folder is a special folder ; for example, Inbox , Outbox , Contacts , and so on.	Client specified a special folder in a FolderUpdate command request. Special folders cannot be deleted.	Item	None.
4	The specified folder does not exist.	Client specified a nonexistent folder in a FolderUpdate command request.	Item	Issue a FolderSync command for the new hierarchy.
6	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the FolderUpdate command request. If continued attempts to synchronization

				fail, consider returning to synchronization key 0.
7	Access denied.	The user does not have permission to access the mailbox .	Global	Refer user to mail administrator.
8	The request timed out.	The server took too long to respond to the request.	Global	Retry.
9	Synchronization key mismatch or invalid synchronization key.	The client sent a malformed or mismatched synchronization key, or the synchronization state is corrupted on the server.	Global	Issue a FolderSync command request with a synchronization key of 0.
10	Malformed request.	The client sent a FolderCreate command request that contains a semantic error.	Global	Fix bug in client code. Double-check the request for accuracy.
11	Code unknown.	Unusual back-end issue.	Global	No solution.

2.2.2.5 GetItemEstimate Status Codes

The following table lists the status codes for the **GetItemEstimate** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.

2	A collection was invalid or one of the specified collection IDs was invalid.	One or more of the specified folders does not exist or an incorrect folder was requested.	Item	Issue a FolderSync to get the new hierarchy. Then retry with a valid collection or collection ID.
3	The synchronization state has not been primed.	The client has issued a GetItemEstimate command without first issuing a Sync command request with synchronization key zero (0).	Item	Issue a Sync command with synchronization key of zero (0) before issuing GetItemEstimate again.
4	Invalid synchronization key	Malformed or mismatched synchronization key. —or— The synchronization state is corrupted on the server.	Global	Delete any items added since the last successful synchronization and return to synchronization key zero (0). The ItemID value is the new item ID sent back by Sync during synchronization zero (0). These ItemID values are different than the current item IDs. Client code SHOULD check for duplicate items.

2.2.2.6 MeetingResponse Status Codes

The following table lists the status codes for the **MeetingResponse** command.

Code	Meaning	Cause	Scope	Resolution
------	---------	-------	-------	------------

1	Success.	Server successfully completed command.	Global	None.
2	Invalid meeting request.	The client has sent a malformed or invalid item.	Item	Stop sending the item. This is not a transient condition.
3	An error occurred on the server mailbox .	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the MeetingResponse . If continued attempts fail, synchronize the folder again, and then attempt the MeetingResponse command again. If it still continues to fail, make no changes.
4	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the MeetingResponse . If continued attempts fail, synchronize the folder again, and then attempt the MeetingResponse command again. If it still continues to fail, make no changes.

2.2.2.7 MoveItems Status Codes

The following table lists the status codes for the **MoveItems** command.

Code	Meaning	Cause	Scope	Resolution
1	Invalid collectionId	The source folder collectionId	Item	Issue a FolderSync

	for source.	is not recognized by the server, possibly because the source folder has been deleted.		command to get the new hierarchy. Then, use a valid collectionID .
2	Invalid collectionId for destination.	The destination folder collectionId is not recognized by the server, possibly because the source folder has been deleted.	Item	Issue a FolderSync to get the new hierarchy. Then, use a valid collectionID .
3	Success.	Server successfully completed command.	Global	None.
4	Source and destination collectionIds are the same.	The client supplied a destination folder that is the same as the source.	Item	Only send requests where the collectionIds for the source and destination differ.
5	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the MoveItems . If continued attempts fail, give up on the item.
6	An item with that name already exists in the destination.	Client tried to move an item into a folder that contains another item with the same name.	Item	Inform the user of the collision and prompt the user to rename the item before trying to move it.

7	Source or destination item was locked.	Transient server condition.	Item	Retry.
---	--	-----------------------------	------	--------

2.2.2.8 Ping Status Codes

The following table lists the status codes for the **Ping** command.

Code	Meaning	Cause	Scope	Resolution
1	The heartbeat interval expired before any changes occurred in the folders being monitored.		Global	Reissue the Ping command request.
2	Changes occurred in at least one of the monitored folders. The response specifies the changed folders.		Global	Issue a Sync request for each folder that was specified in the Ping command response.
3	The Ping command request omitted required parameters.	The Ping command request omitted one or both of the folder list or interval.	Global	Reissue the Ping command request with the entire XML body.
4	Syntax error in Ping command request.	Frequently caused by poorly formatted WBXML .	Global	Fix bug in client code.
5	The specified heartbeat interval is	The client sent a Ping command request with a	Global	Reissue the Ping command by

	outside the allowed range. For intervals that were too short, the response contains the shortest allowed interval. For intervals that were too long, the response contains the longest allowed interval.	heartbeat interval that was either too long or too short.		using a heartbeat interval inside the allowed range. Setting the interval to the value returned in the Ping response will most closely accommodate the original value specified.
6	The Ping command request specified more than the allowed number of folders to monitor. The response indicates the allowed number in the MaxFolders element.	The client sent a Ping command request that specified more folders than the server is configured to monitor.	Global	Direct the user to select fewer folders to monitor. Resend the Ping command request with the new, shorter list.
7	Folder hierarchy sync required.	The folder hierarchy is out of date; a folder hierarchy sync is required	Global	Issue a FolderSync command to get the new hierarchy and prompt the user, if it is necessary,

				for new folders to monitor. Reissue the Ping command.
8	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry.

2.2.2.9 Provision Status Codes

The following table lists the status codes for the **Provision** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	The requested policy data is included in the response.	Policy	Apply the policy.
2	Protocol error.	Syntax error in the Provision command request.	Global	Fix bug in client code.
2	Policy not defined.	There is no policy of the requested type that is defined on the server.	Policy	Stop sending policy information. No policy is implemented.
3	The policy type is unknown.	The client sent a policy that the server does not recognize.	Policy	Issue a request by using MS-EAS-Provisioning-WBXML.
3	An error	Server	Global	Retry.

	occurred on the server.	misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.		
4	The policy data is corrupted.	The policy data on the server is corrupted.	Policy	Direct the user to contact the server administrator.
5	Policy key mismatch.	The client is trying to acknowledge an out-of-date or invalid policy.	Policy	Issue a new Provision request to obtain a valid policy key.

2.2.2.10 ResolveRecipients Status Codes

The following table lists the status codes for the **ResolveRecipients** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.
2	The recipient provided was ambiguous. The response lists all possible matches. No certificate nodes were returned.	The recipient string supplied by the client matched more than one, but not more than MaxAmbiguousRecipients , recipients.	Item	Prompt the user to select the intended recipient from the list returned.
3	The recipient	The recipient string supplied by the client matched more than	Item	Prompt the user to select

	provided was ambiguous. The response lists some possible matches. No certificate nodes were returned.	MaxAmbiguousRecipients recipients.		the intended recipient from the list returned or to get more recipients.
4	No matching entries were found for the recipient. No certificates were returned.	The recipient does not exist or the supplied recipient string was incorrect.	Item	Inform the user of the error and direct the user to check the spelling.
5	Protocol error.	Syntactic or semantic error in the ResolveRecipients request.	Global	Fix bug in client code.
6	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry.
7	No certificate found.	No certificate was found but one was requested	Item	Prompt the user.
8	Global limit hit	The global limit of certificates for the request was hit	Item	Retry with fewer recipients if possible, otherwise prompt the user.
9	Certificate	There was an error enumerating the	Item	Prompt the

	enumeration failure.	certificates.		user.
--	----------------------	---------------	--	-------

2.2.2.11 Search Status Codes

The following table lists the status codes for the **Search** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.
2	The request was invalid.	One or more of the search parameters was invalid.	Item	Inform the user which parameter was invalid. Describe the valid options and prompt the user to select a valid value.
3	An error occurred on the server.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry.
4	Bad link.	A bad link was supplied	Global	Prompt user to reformat link.
5	Access denied.	Access was denied to the resource	Global	Prompt the user.
6	Not found.	Resource was not found	Global	Prompt the user.
7	Connection failed.	Failed to connect to the resource	Global	Prompt the user. Sometimes these are transient, so retry. If it continues to fail, point user

				to administrator.
8	Too complex.	The query was too complex.	Global	Reduce the complexity of the query. Prompt user if necessary.
9	Index not loaded.	The server index is not loaded	Global	Point user to the administrator.
10	Timed out.	The search timed out	Global	The search timed out. Retry with or without rebuilding results. If it continues, contact the Administrator.
11	Folder sync required.	The folder hierarchy is out of date.	Global	Issue a FolderSync and try again.
12	End of retrievable range warning.	The requested range has gone past the end of the range of retrievable results.	Local	Prompt the user that there are no more results that can be fetched, and the user might consider restricting their search query.
13	Access blocked.	Access is blocked to the specified resource	Global	Prompt the user.
14	Credentials required.	To complete this request, basic credentials are required.	Global	If over a trusted connection,

				supply the basic credentials from the user (prompt if necessary). Otherwise fail as if the access denied status code was provided.
--	--	--	--	--

2.2.2.12 Sync Status Codes

The following table lists the status codes for the **Sync** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.
2	Protocol version mismatch.	The client's version string specified version was different than previous Sync requests.	Global	Specify the correct version in the version string.
3	Invalid synchronization key.	Malformed or mismatched synchronization key. —or— Synchronization state corrupted on server.	Global	Delete any items that were added since the last successful synchronization and return to synchronization key 0. The ItemID value is the new item ID sent back by Sync during synchronization zero (0). These ItemID values are different

				than the current item IDs. Client code SHOULD check for duplicate items.
4	Protocol error.	There was a semantic error in the recovery synchronization.	Item or Global	<p>This is caused by a bug in the client. Fix the client. In recovery synchronization, resend all changes from previous synchronization.</p> <p>Wait until all changes are acknowledged before committing the new synchronization key.</p> <p>Return to synchronization key zero (0).</p>
5	Server error.	Server misconfiguration, temporary system issue, or bad item. This is frequently a transient condition.	Global	Retry the synchronization. If continued attempts to synchronization fail, consider returning to synchronization key 0.
6	Error in client/server conversion.	The client has sent a malformed or invalid item.	Item	Stop sending the item. This is not a transient condition.
7	Conflict matching the client and server object.	The client has changed an item for which the conflict policy indicates that the server's changes take precedence.	Item	If it is necessary, inform the user that the change they made to the item has been overwritten by a server change.
8	Object not	The client issued a	Item	Issue a synchronization

	found.	Fetch or Change command that has an ItemID value that is no longer valid on the server (for example, the item was deleted).		request and prompt the user if necessary.
9	The Sync command cannot be completed.	User account could be out of disk space.	Item	Free space in the user's mailbox and try the Sync command again.
10	The NotifyGUID element caused an error.	An error occurred while setting the NotifyGUID element.	Item	This error SHOULD be ignored, as the NotifyGUID element is no longer used.
11	The Sync command cannot be completed.	The device has not been provisioned for notifications yet.	Item	Use the Provision command to request policy settings from the server then retry the Sync command.
12	The folder hierarchy has changed.	Mailbox folders are not synchronized.	Item	Perform a FolderSync command and then retry the Sync command.
13	The Sync command request is not complete.	An empty or partial Sync command request is received and the cached set of notify-able collections is missing.	Item	Resend a full Sync command request.

14	Invalid Wait value.	The Wait element value is too large or too small.	Item	Update the Wait element value according to the Limit element and then resend the Sync command request.
15	Invalid Sync command request.	Too many collections are included in the Sync request.	Item	Notify the user and synchronize fewer folders within one request.
16	Retry	Something on the server caused a retrieable error.	Global	Resend the request.

2.2.2.13 ValidateCert Status Codes

The following table lists the status codes for the **ValidateCert** command.

Code	Meaning	Cause	Scope	Resolution
1	Success.	Server successfully completed command.	Global	None.
2	Protocol error.	Supplied protocol parameters are out of range or invalid.	Global	Fix client code.
3	The signature in the digital ID cannot be validated.	The signature in the certificate is invalid.	Item	Verify that the certificate has a valid signature.
4	The digital ID was issued by an untrusted source.	The certificate source is not trusted by the server.	Item	Contact the administrator to add the certificate to the trusted sources list if it is

				required.
5	The certificate chain that contains the digital ID was not created correctly.	Invalid, incorrectly formatted certificate.	Item	Verify that the certificate chain is formatted correctly.
6	The digital ID is not valid for signing e-mail messages.	The supplied certificate is not meant to be used for signing e-mail.	Item	Prompt the user.
7	The digital ID used to sign the message has expired or is not yet valid.	The certificate has expired.	Item	Obtain a new certificate.
8	The time periods during which the digital IDs in the certificate chain are valid are not consistent.	One or more certificates in the chain could be out of date.	Item	Get the most recent certificate chain for the certificate.
9	A digital ID in the certificate chain is used incorrectly.	The supplied certificate is not valid for what it is being used for.	Item	Obtain a new certificate.
10	Information associated with the digital ID is missing or incorrect.	The certificate format is incorrect.	Item	Obtain a new certificate.
11	A digital ID in the certificate chain is used	A certificate in the chain is being used for a purpose other than what it was	Item	Obtain the correct certificate

	incorrectly.	intended for.		chain.
12	The digital ID does not match the recipient's e-mail address.	Incorrect certificate was supplied, could be malicious.	Item	Obtain the correct certificate for the user.
13	The digital ID used to sign this message has been revoked. This can indicate that the issuer of the digital ID no longer trusts the sender, the digital ID was reported stolen, or the digital ID was compromised.	The certificate has been revoked by the certification authority that issued it.	Item	Obtain a new certificate.
14	The validity of the digital ID cannot be determined because the server that provides this information cannot be contacted.	The certificate revocation server is offline.	Item	Retry request after some time.
15	A digital ID in the chain has been revoked by the authority that issued it.	A certificate in the chain has been revoked.	Item	Obtain a new certificate.
16	The digital ID cannot be	The signature in the certificate is	Item	Verify that the certificate has a

	validated because its revocation status cannot be determined.	invalid.		valid signature.
17	An unknown server error has occurred.	The certificate source is not trusted by the server.	Item	Contact the administrator to add the certificate to the trusted sources list if it is necessary.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model, as long as their external behavior is consistent with that specified in this document.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.3.1 Initial Synchronization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The client creates request messages consisting of an **HTTP** header, as specified in [MS-ASHTTP], and the **XML** command to be performed on the server, as specified in section 2.2.

The request message is sent to the server by the client and a response message is received back from the server.

3.1.5.1 Downloading Policy Settings

This section describes how the client device can download policy settings from the server by using the **Provision** command.

The first command the client issues to the server **MUST** be a **Provision** command. This initial request for policy settings contains the **PolicyType** element, which specifies the format in which the policy settings are provided. The server then responds with the **PolicyType**, **PolicyKey**, and **Data** elements. The policy key is used by the server to mark the state of policy settings on the client device. The policy settings, in the format specified in the **PolicyType** element, are contained in the **Data** element.

The client device then applies the policy settings that were received from the server and sends an acknowledgement back to the server in another **Provision** command request. The acknowledgement from the client device contains **PolicyType**, **PolicyKey**, and **Status** elements. The **Status** element indicates whether the policy settings were successfully applied by the client. The response from the server contains **PolicyType**, **PolicyKey**, and **Status** elements. The **Status** element indicates whether the server successfully recorded the client's acknowledgement.

The following figure shows the process for downloading policy settings.

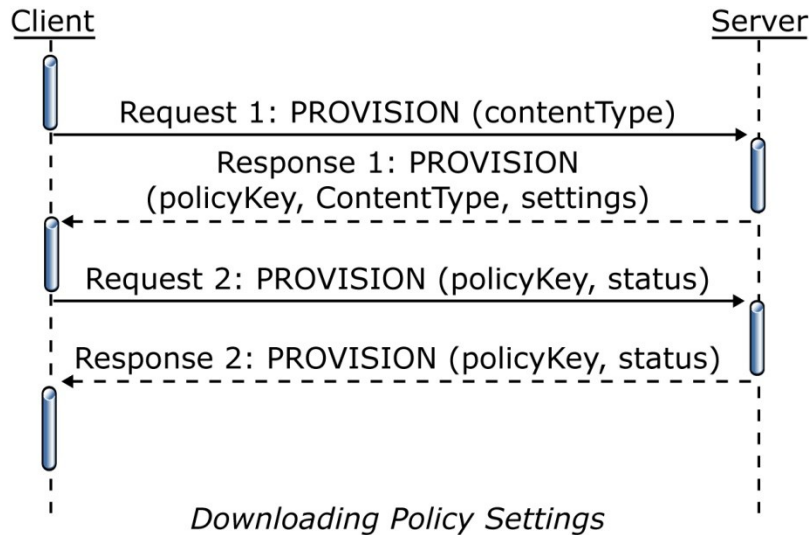


Figure 4: Downloading policy settings

The following table lists the command sequence for downloading policy settings.

Order	Client action	Server action
-------	---------------	---------------

1	The client sends a Provision command request with the type of policy settings to be downloaded.	The server response contains the policy type, policy key, policy settings, and status code.
2	The client acknowledges that it received and applied the policy settings by sending another Provision command request with the policy type, policy key, and status code.	The server response contains the policy type, policy key, and status code to indicate that the server recorded the client's acknowledgement.

3.1.5.2 Synchronizing Inbox, Calendar, Contacts, and Tasks Folders

The client initializes synchronization with the server by sending a Sync command request to the server. The client can synchronize the Inbox, Calendar, or **Contacts** folder, or any folder within the **mailbox**.

Note that the Inbox, Calendar, Contacts, and Tasks folders **SHOULD** be synchronized by using the server ID of the **collection**, which requires that the client first perform a **FolderSync** command to discover the folder hierarchy, as specified in section 3.1.5.3.

In order to start synchronization, an initial synchronization key **MUST** be obtained from the server. The client obtains the key by sending the server an initial synchronization request with a synchronization key of zero (0) and specifying the appropriate CollectionId(s). The server responds with a new synchronization key, which is generated by the server for each transaction. The client **MUST** store the synchronization key and reissue the key in the next synchronization request. Therefore, the initial synchronization of a collection consists of one request to obtain the synchronization key and a second request to do the full synchronization.

The following figure shows the initial synchronization process.

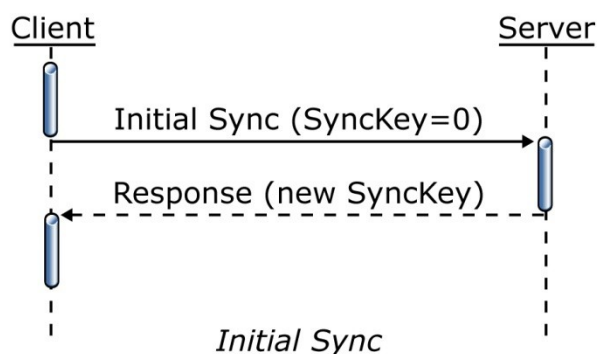


Figure 5: Initial synchronization process

The client then performs a full synchronization of a folder by sending a second **Sync** command to the server, specifying the **GetChanges** element and the synchronization key that is returned by the initial synchronization.

The server responds by adding all the items in the collection to the client and returning a new synchronization key, which can be used in successive synchronizations. The client SHOULD delete its copy of all objects in the collection that are being synchronized before the client performs a full synchronization. The client can use the **GetItemEstimate** command to obtain an estimate of the number of items that have to be synchronized before completely synchronizing a collection, which is useful when the client user interface (UI) displays a progress bar while getting items from the server. In some cases, the client could have to submit a **WindowSize** element that specifies the number of items to be synchronized at a time.

If more items remain to be synchronized, the **MoreAvailable** element is returned in the **Sync** command response. The client then continues to call the **Sync** command until no more items are available. For more details about the **WindowSize** element of the **Sync** command, see section 2.2.1.19.1.4. The following figure shows the folder synchronization process.

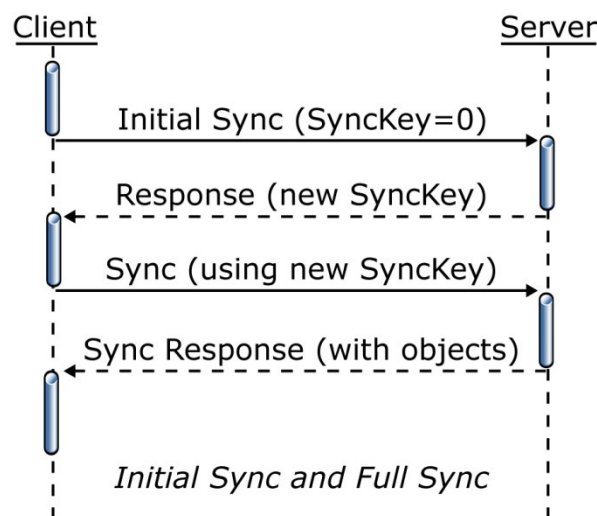


Figure 6: Folder synchronization process

After a full synchronization has been performed on a collection, successive synchronizations are used to obtain additions, deletions, or changes to the initial collection state. The client can use the **Sync** command request to add, delete, or change items on the server, and the server can use the **Sync** command response to add, delete, or change items on the client.

The following table lists the command sequence for folder synchronization.

The asterisk (*) in the Order column means that a step can be repeated multiple times. [n] means that a step is optional.

Order	Client Action	Server Action
1	The client sends the Sync command for the Email , Calendar , Contacts , and/or Tasks collection with a synchronization key of zero (0). This establishes a partnership with the server, initializing server data for the device.	The server response contains the synchronization key for the collection, to be used in successive synchronizations.
2*	The client sends the Sync command with a synchronization key of zero (0) for other collections to be synchronized.	The server responds with new synchronization keys for each collection.
[3]	The client sends the GetItemEstimate command for all collections to be synchronized. This step can be skipped if it is not required by the client UI.	The server response indicates how many items will be added for each collection.
4*	<p>The client sends the Sync command with the GetChanges element for a collection. The command SHOULD include the WindowSize element, the recommended value for which is 100.</p> <p>This step is repeated for each collection to be synchronized or all collections can be combined into one request.</p>	The server response contains Add elements for items in the collection. If the response contains the MoreAvailable element, this step is repeated.

The client SHOULD use the **WindowSize** element to break the server **Add** elements into sets of multiple items. The recommended window size is 100. For more details about the **WindowSize** element used by the **Sync** command, see section 2.2.1.19.1.4.

3.1.5.3 Synchronizing a Folder Hierarchy

This section describes how to use the **FolderSync** command to discover the folder hierarchy of the user's **mailbox**, perform an initial synchronization to get a synchronization key for a **collection**, and perform a full synchronization to get the items in that collection from the server.

The **FolderSync** command is used to synchronize the folder hierarchy of a user's mailbox folder tree. An initial folder synchronization request with a synchronization key of zero (0) is sent to the server, which responds with an initial synchronization key and all the folders in the user's mailbox. The folders are identified by server ID, which can then be used in a **Sync** command to synchronize the items of those folders.

Additional folder synchronizations can be performed, by using the synchronization key from the initial folder **Sync** command response, to get folder additions, deletions, or updates from the server. At any point, the client can repeat the initial folder **Sync** command. Existing collection IDs do not change when the client resynchronizes.

Before a folder hierarchy can be synchronized, an initial synchronization key **MUST** be obtained from the server. The client does this by sending the server an initial synchronization request with a synchronization key of zero (0). The response will contain the server's folder hierarchy.

When the folder hierarchy of the user's mailbox has been discovered and an initial synchronization has been performed on a folder to obtain a synchronization key, the contents of that folder can then be synchronized. To synchronize the content, send the **Sync** command to the server, specifying the **GetChanges** element and the synchronization key for the collection. The server responds by adding all the items in the collection to the client and returning a new synchronization key, which can be used in successive synchronizations. In some cases, the client **SHOULD** use the **GetItemEstimate** command to obtain an estimate of the number of items that **MUST** be synchronized before completely synchronizing a collection, which is useful when the client UI displays a progress bar while it retrieves items from the server. In some cases, the client could have to submit a **WindowSize** element that specifies the number of items to be synchronized at a time. If more items remain to be synchronized, the **MoreAvailable** element is returned in the **Sync** command response. The client then continues to call the **Sync** command until no more items are available. The following figure shows the process for synchronizing multiple folders.

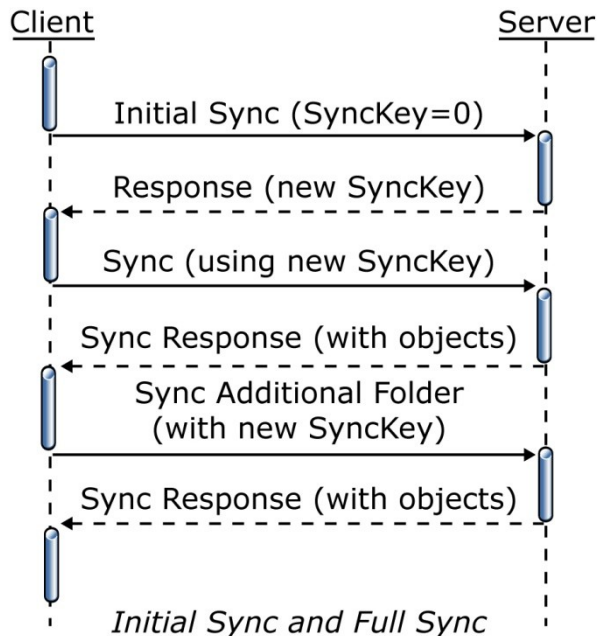


Figure 7: Multiple folder synchronization process

After a full synchronization has been performed on a collection, successive synchronizations are used to obtain additions, deletions, or changes to the initial collection state. The client can use a **Sync** command request to add, delete, or change items on the server, and the server can use the **Sync** command response to add, delete, or change items on the client.

The following table lists the command sequence for folder hierarchy synchronization.

The asterisk (*) in the Order column means that a step can be repeated multiple times. [*n*] means that a step is optional.

Order	Client action	Server action
1	The client sends the FolderSync command with the SyncKey element set to zero (0) to get the folder hierarchy and the server IDs of all the folders.	The server response contains the folder hierarchy. Client stores names and server IDs of all folders that can be synchronized.
2	The client sends the Sync command for one collection, with the SyncKey element set to zero (0) and the server ID of the collection to be synchronized. This establishes a partnership with the server, initializing server data for the device.	The server response contains the synchronization key for the collection, to be used in successive synchronizations.

3*	The client sends a Sync command with the SyncKey element set to zero (0) for other collections to be synchronized.	The server responds with new SyncKey values for each collection.
[4]	The client sends the GetItemEstimate command for all collections to be synchronized. This step can be skipped if it is not required by the client UI.	The server response indicates how many items will be added for each collection.
5*	The client sends the Sync command for a collection, specifying the GetChanges element and the server ID of the collection to be synchronized. The command SHOULD include the WindowSize element, the recommended value for which is 100. Repeat this step for each collection to be synchronized.	The server response contains Add commands for items in the collection. If the response contains the MoreAvailable element, this step is repeated.

3.1.5.4 Receiving and Accepting Meeting Requests

This section describes how to retrieve items from the **Inbox folder** by using the **Sync** command, to respond to a **meeting** request item by using the **MeetingResponse** command, and to synchronize the Calendar folder by using the **Sync** command so that the new calendar object is added to the client's calendar.

A meeting request is returned by the server in response to a synchronization of the Inbox folder. A meeting request is an e-mail message that has an embedded calendar item. The message contains a **MessageClass** element that has a value of **IPM.Schedule.Meeting.Request**, and its **ApplicationData** element contains a **MeetingRequest** element. When the client displays the meeting request message, the client SHOULD offer the options of accepting, declining, or tentatively accepting the meeting. If one of these actions is selected, the client sends a **MeetingResponse** command to the server.

If the response to the meeting is accepted or is tentatively accepted, the server will add or update the corresponding calendar item and return its server ID in the **CalendarId** element of the response. If the response to the meeting is declined, the response will not contain a **CalendarId** element because the server will delete the corresponding calendar item. If the client had created a tentative meeting calendar item, the client SHOULD update that item with the returned server ID (if accepted or tentative). The client MUST also change the busy status on the client calendar item from tentative to busy if the meeting request was accepted. Note that, if the client synchronizes the Calendar folder after responding to a meeting request, the calendar item in question will be in conflict if the client also sends the changed item change

for it back to the server. This conflict is resolved according to the conflict resolution rules that are specified by the client in the **Sync** command request.

If the meeting request was accepted, the Calendar folder **MUST** be synchronized for the client to obtain the new calendar item. The new calendar item for the accepted meeting is added here and **MUST** be added to the client's calendar.

The following figure shows how meeting requests are received and accepted.

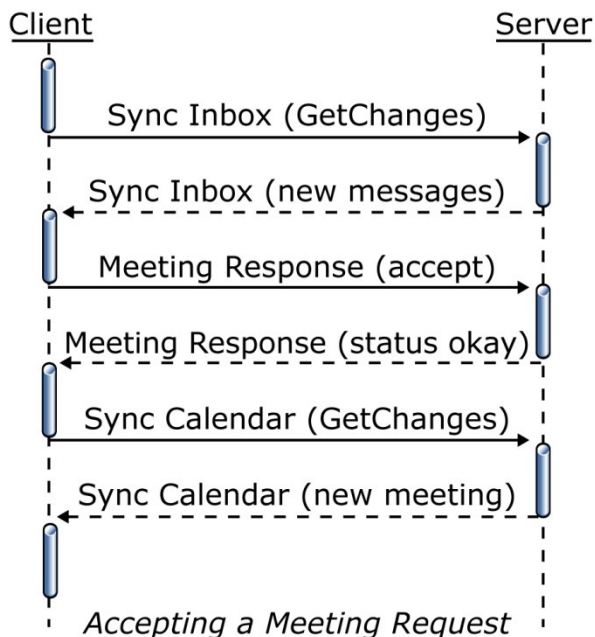


Figure 8: Meeting request acceptance process

The following table lists the command sequence for receiving and accepting meeting requests.

Order	Client action	Server action
1	The client sends the Sync command for the Inbox collection with the value of the SyncKey element set to zero (0).	The server response contains the SyncKey for the collection, to be used in successive synchronizations.
2	The client sends a Sync command, specifying the GetChanges element and the SyncKey for the Inbox folder . The command SHOULD include the WindowSize element, the recommended value for	The server response contains Add elements for items in the Inbox collection, including a meeting request item. If the response contains the MoreAvailable element, this step is repeated.

	which is 100.	
3	The user chooses to accept, decline, or tentatively accept a meeting request that is displayed in the client UI.	
4	The client sends a MeetingResponse command to the server, which specifies that the meeting was accepted, declined, or tentatively accepted, and provides the server IDs of the meeting request message and its parent folder.	The server sends a response that contains the MeetingResponse command request status.
5	If the meeting was accepted, the client sends a Sync command for the Calendar collection, specifying the GetChanges element.	The server responds with any changes to the Calendar folder caused by the last synchronization and the new calendar item for the accepted meeting.

3.1.5.5 Handling Status Errors

Client software **MUST** handle errors that occur during synchronization sessions. Errors fall into two categories: **HTTP** errors and synchronization errors. HTTP errors are standard error codes, such as 401 Logon failed, and they are returned from the server in response to an HTTP POST. Synchronization errors result from a problem during the synchronization process. Synchronization errors are indicated by codes that are returned in the **Status** element of a command response. For more details about the status codes, see section 2.2.2.

The client **MUST** implement error handling and a user interface (UI). Some errors are handled by a recovery procedure. Other errors require that an error message be displayed, along with a prompt for the user to respond.

In addition to synchronization errors that the server sends, incomplete communication between server and client can result in the failure of a synchronization session. The server has an error recovery feature that enables a client to respond to errors by repeating the most recent synchronization session. The client **MUST** handle synchronization failures by retrying the synchronization, either immediately or later. The server tracks synchronization requests to be able to respond appropriately in both of the following cases:

- The client failed in communicating a full request to the server for synchronization.

- In this case, the client sends a request but the server does not receive the request. The server does not act on the request, and no server-side changes occur. Therefore, no response is sent to the client. The client **MUST** resend a synchronization request if there is no server response.
- The server failed in communicating a response to the client for updates.
 - In this case, the server response is not received by the client. The data on the server changed. The client **MUST** resend the request. The server recognizes the duplicate request. Because the server changes have already occurred, the server resends the response to the client to keep the server and client synchronized.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 *Downloading the Current Server Security Policy by Using the Provision Command*

For examples on downloading the current server security policy by using the **Provision** command, see [MS-ASPROV] section 4.

4.2 *Discovering Account Settings by Using the AutoDiscover Command*

The **Autodiscover** command enables clients to discover core account configuration information by using the user's **SMTP** address as the primary input by means of the following process:

1. The end-user enters his or her e-mail address and **domain** credentials, for example, kim@contoso.com.
2. The client uses the domain information in the user's e-mail address, that is, contoso.com, and tries to locate the Autodiscover service by sending an **Autodiscover** command request to the following predefined URLs:
 https://<smtp-address-domain>/autodiscover/autodiscover.xml
 https://autodiscover.<smtp-address-domain>/autodiscover/autodiscover.xml
3. In this example, these URLs map to
 https://contoso/autodiscover/autodiscover.xml and
 https://autodiscover.contoso/autodiscover/autodiscover.xml.

4. If **Domain Name Systems (DNS)** contains a host record that maps one of these URLs to a server where the Autodiscover service is hosted, then the Autodiscover service responds with the settings that are required for the device to begin synchronizing. This response includes values for the Server type, the URL, and the **Name** element.
5. If redirection to another Autodiscover service is required, then the **Redirect** element is present and contains a URL to the Autodiscover server to query for the desired information. The device then re-create a partnership with the new server, and send an **Autodiscover** command request to that server.
6. If the response included the settings that are required for the device to begin synchronization, then the device applies the settings to initiate synchronization.
7. If the Autodiscover command request in step 3 fails, then the device performs a **DNS SRV** lookup for `_autodiscover._tcp.<smtp-address-domain>.com`, which in this example maps to `_autodiscover._tcp.contoso.com`. If the DNS lookup is successful, then `"mail.<smtp-address-domain>.com"` is returned, which maps to `"mail.contoso.com"`. The device then applies the settings to initiate synchronization. For more details about performing the DNS SRV lookup, see [DNS-SRV].

The following sections show success and error response messages.

Account autodiscovery uses an e-mail address to look up information that is required to configure software. Given an e-mail name (such as `EduardDell@Woodgrovebank.com`), a list of possible Autodiscover servers is generated. The client contacts the name `autodiscover.domainname` to provide the information. If that information is not found, the client tries to send the request to the domain name. If the information still is not retrieved, the client can use a manual configuration. For example, the client tries these servers:

- `autodiscover.woodgrovebank.com`
- `woodgrovebank.com`

Each server is sent an **HTTP** Post command. The post data is an **XML** request for a certain type of information. E-mail account configuration is the first use. The **XML** contains information that helps execute the request. For mail, the information includes the e-mail address, the protocols that the client software supports, the Web browser that is installed, the type of proxy that is being used, and the types of authentication that can be used.

The post is sent for `servername/Autodiscover/Autodiscover.xml`. The `servername` is defined according to the process described earlier in this topic.

4.2.1 Request XML Body Outline

```
<Autodiscover
xmlns="http://schemas.microsoft.com/exchange/autodiscover/mobilesync/re
quests/schema/2006">

  <Request>
```

```

    <EmailAddress>eduarddell@woodgrovebank.com</EmailAddress>
    <AcceptableResponseSchema>
      http://schemas.microsoft.com/exchange/autodiscover/mobilesync/
      responseschema/2006
    </AcceptableResponseSchema>
  </Request>
</Autodiscover>

```

4.2.2 Response XML Body Outline - Case Error

```

<Autodiscover
  xmlns:A="http://schemas.microsoft.com/exchange/autodiscover/mobilesync/
  responseschema/2006">
  <A:Response>
    <A:Culture>en:en</A:Culture>
    <A:User>

<A:EmailAddress>eduarddell@woodgrovebank.com</A:EmailAddress>
    </A:User>
    <A:Action>
      <A:Error>
        <Status>1</Status>
        <Message>The directory service could not be
reached</Message>
        <DebugData>MailUser</DebugData>
      </A:Error>
    </A:Action>
  </A:Response>
</Autodiscover>

```

4.2.3 Response XML Body Outline - Case Redirect

In the following redirect example, assume that the **Autodiscover** command request was sent to autodiscover.woodgrovebank.com. The redirect node indicates to the client to try autodiscover.loandepartment.woodgrovebank.com.

```

<Autodiscover
xmlns:A="http://schemas.microsoft.com/exchange/autodiscover/mobilesync/
responseschema/2006">
  <A:Response>
    <A:Culture>en:en</A:Culture>
    <A:User>
      <A:DisplayName>Eduard Dell</A:DisplayName>

<A:EmailAddress>eduarddell@woodgrovebank.com</A:EmailAddress>
    </A:User>
    <A:Action>
      <A:Redirect>eduarddell@loandep.woodgrovebank.com
</A:Redirect>
    </A:Action>
  </A:Response>
</Autodiscover>

```

4.2.4 Response XML Body Outline - Case Server Settings

In the following success response, the Autodiscover service has provided server URL information for two services: MobileSync and CertEnroll. The client can use the MobileSync URL to configure the settings for the [MS-ASCMD]. The client can also optionally use the CertEnroll information to obtain a client certificate for **SSL** negotiation. <27>

```

<Autodiscover
xmlns:A="http://schemas.microsoft.com/exchange/autodiscover/mobilesync/
responseschema/2006">
  <A:Response>
    <A:Culture>en:en</A:Culture>
    <A:User>
      <A:DisplayName>Eduard Dell</A:DisplayName>

<A:EmailAddress>eduarddell@woodgrovebank.com</A:EmailAddress>
    </A:User>
    <A:Action>
      <A:Settings>
        <A:Server>

```

```

        <A:Type>MobileSync</A:Type>
        <A:Url>
            https://loandep.woodgrovebank.com/Microsoft-
Server-ActiveSync
        </A:Url>
        <A:Name>
            https://loandep.woodgrovebank.com/Microsoft-Server-
ActiveSync
        </A:Name>
        </A:Server>
        <A:Server>
            <A:Type>CertEnroll</A:Type>

<A:Url>https://cert.woodgrovebank.com/CertEnroll</A:Url>
        <A:Name />
        <A:ServerData>CertEnrollTemplate</A:ServerData>
        </A:Server>
    </A:Settings>
</A:Action>
</A:Response>
</A:Autodiscover>

```

4.2.5 Response XML Body Outline - Case Framework Error

If the provider cannot be found, or if the **AcceptableResponseSchema** cannot be matched, then the following XML fragment is returned to the client.

The error code 600 means an invalid request was sent to the service, and 601 means that a provider could not be found to handle the **AcceptableResponseSchema** that was specified.

```

<Autodiscover
xmlns:A="http://schemas.microsoft.com/exchange/autodiscover/mobilesync/
responseschema/2006">
    <A:Response>
        <A:Error Time="16:56:32.6164027" Id="1054084152">
            <A:ErrorCode>600</ErrorCode>
            <A:Message>Invalid Request</Message>

```

```

        <A:DebugData />
    </A:Error>
</A:Response>
</Autodiscover>

```

4.2.6 Response XML – Case Framework Default

For unauthenticated requests, the server can serve a static page with contents, such as the following.<28>

```

<Autodiscover >
  <Account>
    <AccountType>default e-mail</AccountType>
    <Action>settings</Action>
    <Image>http://www.abcd.com/def.jpg</Image>
    <ServiceHome>http://www.microsoft.com</ServiceHome>
    <RedirectUrl>...</RedirectUrl>

    <Protocol>
      <Type>POP</Type>
      <Server>popserverFQDN</Server>
      <Port>110</Port>
    </Protocol>

    <Protocol>
      <Type>SMTP</Type>
      <Server>smtpserverFQDN</Server>
      <Port>25</Port>
    </Protocol>

    <Protocol>
      <Type>IMAP</Type>
      <Server>imapserver1FQDN</Server>
    </Protocol>

    <Protocol>

```



```

    <Type>IMAP</Type>
    <Server>imapserver2FQDN</Server>
    <Port>143</Port>
  </Protocol>
</Account>
</Autodiscover>

```

4.3 *Synchronizing Data by Using the Sync Command*

This section provides sample messages related to the **Sync** command.

4.3.1 Synchronizing Folders

The following example shows a request that is sent to the server to synchronize an e-mail folder. The request asks that deleted items be moved to the Deleted Items folder. The request also asks for changes on the server to be specified in the response. The server response contains the new synchronization key and the items to be added, deleted, and changed on the client.

Request

```

<Collections>
  <Collection>
    <Class>Email</Class>
    <SyncKey>6</SyncKey>
    <CollectionId>1</CollectionId>
    <DeletesAsMoves/>
    <GetChanges/>
    <Options> ... </Options>
  </Collection>
</Collections>

```

Response

```

<Collections>
  <Collection>
    <Class>Email</Class>
    <SyncKey>7</SyncKey>
    <CollectionId>1</CollectionId>
    <Status>1</Status>
  </Collection>
</Collections>

```

```

    <Commands>
      <Add>...</Add>
      <Delete>...</Delete>
      <Change>...</Change>
      <Fetch>...</Fetch>
    </Commands>
  </Collection>
</Collections>

```

4.3.2 Fetching an E-Mail by Using the ServerID

The following example shows a request that is sent to the server to fetch an item from the server by its server ID.

Request

```

<Commands>
  <Fetch >
    <ServerId>1:14</ServerId>
  </Fetch >
</Commands>

```

A response from the server contains the server ID, status, and application data of the requested item.

Response

```

<Responses>
  <Fetch>
    <ServerId>1:14</ServerId>
    <Status>1</Status>
    <ApplicationData>...</ApplicationData>
  </Fetch>
</Responses>

```

4.3.3 Uploading New ApplicationData to the Server

This example shows a **Sync** command request that is sent to the server to add a **contact**. The response from the server shows that the synchronization was successful and that the new item from the client, identified by the **ClientId** element, was added to the **collection** on the server. The server also assigns a permanent ID for the newly added item in the **ServerId** element.

After the client receives a successful response, the client uses this server ID for any future **Change** or **Delete** commands for this item.

Request

```
<Commands>
  <Add>
    <ClientId>123</ClientId>
    <ApplicationData>
      <A:EmaillAddress>schai@fourthcoffee.com</A:EmaillAddress>
      <A:FirstName>Sean</A:FirstName>
      <A:MiddleName>W</A:MiddleName>
      <A:LastName>Chai</A:LastName>
      <A:Title>Sr Marketing Manager</A:Title>
    </ApplicationData>
  </Add>
</Commands>
```

Response

```
<Responses>
  <Add>
    <ClientId>123</ClientId>
    <ServerId>4:1</ServerId>
    <Status>1</Status>
  </Add>
</Responses>
```

4.3.4 Updating ApplicationData on the Server

The following example shows a **Sync** command request from the client. The request modifies a **contact**, which is identified by the server ID, on the server. The response from the server shows that the change that is specified by the **Sync** request of the client succeeded and supplies the synchronization key and **collection** ID of the changed item.

Request

```
<Commands>
  <Change>
    <ServerId>3:1</ServerId>
    <ApplicationData>
```

```

        <A:Email1Address>jsmith@fourthcoffee.com</A:Email1Address>
        <A:FirstName>Jeff</A:FirstName>
    </ApplicationData>
</Change>
</Commands>

```

Response

```

<Collections>
    <Collection>
        <Class>Contacts</Class>
        <SyncKey>4</SyncKey>
        <CollectionId>1</CollectionId>
        <Status>1</Status>
    </Collection>
</Collections>

```

4.3.5 Downloading Current Information from the Server

The following example shows a request that is sent to the server to synchronize an e-mail folder. The request asks that deleted items be moved to the Deleted Items folder. The request also asks for changes on the server to be specified in the response. The server response contains the new synchronization key and the items to be added, deleted, and changed on the client.

Request

```

<Collections>
<Collection>
    <Class>Email</Class>
    <SyncKey>6</SyncKey>
    <CollectionId>1</CollectionId>
    <DeletesAsMoves/>
    <GetChanges/>
    <Options> ... </Options>
</Collection>
</Collections>

```

Response

```

<Collections>

```

```

<Collection>
  <Class>Email</Class>
  <SyncKey>7</SyncKey>
  <CollectionId>1</CollectionId>
  <Status>1</Status>
  <Commands>
    <Add>...</Add>
    <Delete>...</Delete>
    <Change>...</Change>
    <Fetch>...</Fetch>
  </Commands>
</Collection>
</Collections>

```

4.3.6 Identifying Acceptance of Partial Collections

The following example shows a **Sync** request that includes the **Partial** element.

```

<Sync xmlns="AirSync:">
  <Collections>
    <Collection>
      <SyncKey>1723058747</SyncKey>
      <CollectionId>10</CollectionId>
    </Collection>
  </Collections>
  <Wait>8</Wait>
  <Partial/>
</Sync>

```

4.3.7 Identifying Acceptance of MIME Content

This example uses the **MIMETruncated**, **MIMETruncation**, and **MIMEDData** elements to identify acceptance of MIME content on the client.

4.3.7.1 Sync Request With Support for MIME Content

```

<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns="AirSync:">

```

```

<Collections>
  <Collection>
    <Class>Email</Class>
    <SyncKey>2</SyncKey>
    <CollectionId>1</CollectionId>
    <DeletesAsMoves/>
    <GetChanges/>
    <Options>
      <MIMETruncation>1</MIMETruncation>
      <MIMESupport>1</MIMESupport>
    </Options>
  </Collection>
</Collections>
</Sync>

```

4.3.7.2 Sync Response With MIME Content

```

<Add>
  <ServerId>1:3</ServerId>
  <ApplicationData>
    <A:To>"James Smith" <jsmith@contoso.com></A:To>
    <A:From>"Jyothi Pai" <jpai@contoso.com></A:From>
    <A:Subject>RE: Presentation</A:Subject>
    <A:DateReceived>2004-11-12T00:45:06.000Z</A:DateReceived>
    <A:DisplayTo>James Smith</A:DisplayTo>
    <A:Importance>1</A:Importance>
    <A:Read>0</A:Read>
    <A:MIMETruncated>0</A:MIMETruncated>
    <A:MIMEData>"Received: from server1.contoso.com
    ([192.168.0.20]) by server2.contoso.com with Microsoft
    SMTPSVC(5.0.2195.6624); ... </A:MIMEData>
    <A:Importance>1</A:Importance>
    <A:Read>0</A:Read>
    <A:MessageClass>IPM.Note.SMIME.MultipartSigned</A:MessageClass>
  </ApplicationData>

```

</Add>

4.3.7.3 Sync Request With BodyPreferences and MIME Support

```
<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns="AirSync:" xmlns:A="AirSyncBase:">
  <Collections>
    <Collection>
      <Class>Email</Class>
      <SyncKey>2</SyncKey>
      <CollectionId>17</CollectionId>
      <DeletesAsMoves/>
      <GetChanges/>
      <Options>
        <A:BodyPreference>
          <A:Type>4</A:Type>
          <A:TruncationSize>512</A:TruncationSize>
        </A:BodyPreference>
        <MIMESupport>1</MIMESupport>
      </Options>
    </Collection>
  </Collections>
</Sync>
```

4.3.7.4 Sync Response with MIME Support

```
<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns="AirSync:" xmlns:A="POOMMAIL:" xmlns:B="AirSyncBase:">
  <Collections>
    <Collection>
      <Class>Email</Class>
      <SyncKey>3</SyncKey>
      <CollectionId>17</CollectionId>
      <Status>1</Status>
      <Commands>
        <Change>
```

```

<ServerId>17:11</ServerId>
<ApplicationData>
  <A:To>"Mike Phipps" <mailto:mike@contoso.com></A:To>
  <A:From>"Arlene Huff" <mailto:arlene@contoso.com></A:From>
  <A:Subject>opaque s + e </A:Subject>
  <A:DateReceived>2007-02-01T06:42:46.015Z</A:DateReceived>
  <A:DisplayTo>Mike Phipps</A:DisplayTo>
  <A:ThreadTopic>opaque s + e</A:ThreadTopic>
  <A:Importance>1</A:Importance>
  <A:Read>1</A:Read>
  <B:Attachments>
    <B:Attachment>
      <B:DisplayName>smime.p7m</B:DisplayName>
      <B:FileReference>17%3a11%3a0</B:FileReference>
      <B:Method>1</B:Method>
      <B:EstimatedDataSize>9340</B:EstimatedDataSize>
    </B:Attachment>
  </B:Attachments>
  <B:Body>
    <B:Type>4</B:Type>
    <B:EstimatedDataSize>13814</B:EstimatedDataSize>
    <B:Truncated>1</B:Truncated>
    <B:Data>Received: from contoso.com ([157.55.97.121])
    By contoso.com ([157.55.97.121]) with mapi;
    Wed, 31 Jan 2007 22:42:45 -0800
    From: Arlene Huff <mailto:arlene@contoso.com>;
    To: Mike <mailto:mike@contoso.com>;
    Content-Class: urn:content-classes:message
    Date: Wed, 31 Jan 2007 22:42:41 -0800
    Subject: opaque s + e
    Thread-Topic: opaque s + e
    Thread-Index: AcdFzCv5tyCXieBuTd2I5APpEvS+iQ==
    Message-ID:

```



```

        <B:Body>
        <A:MessageClass>IPM.Note.SMIME</A:MessageClass>
        <A:InternetCPID>20127</A:InternetCPID>
        <A:Flag/>
        <A:ContentClass>urn:content-
classes:message</A:ContentClass>
        <B:NativeBodyType>1</B:NativeBodyType>
        </ApplicationData>
    </Change>
</Commands>
</Collection>
</Collections>
</Sync>

```

4.3.8 Identifying That More Content is Ready for Download

The following example is a response message indicating that more content is available for download from the server. The content exceeded the **WindowSize** value.

```

<Collection>
    <Class>Email</Class>
    <SyncKey>2</SyncKey>
    <CollectionId>1</CollectionId>
    <Status>1</Status>
    <Commands>
        ...
    </Commands>
    <MoreAvailable/>
</Collection>

```

4.3.9 Synchronizing the Calendar Folder

The following example shows the initial synchronization of the Calendar folder with a synchronization key of 0.

Request

```

<Collection>

```

```

    <Class>Calendar</Class>
    <SyncKey>0</SyncKey>
    <DeletesAsMoves/>
    <GetChanges/>
</Collection>

```

The following example shows the synchronization of the Calendar with a synchronization key that was obtained from a previous synchronization.

Request

```

<Collection>
    <Class>Calendar</Class>
    <SyncKey>9</SyncKey>
    <DeletesAsMoves/>
    <GetChanges/>
</Collection>

```

4.4 *Pinging the Server for Updates by Using the Ping Command*

This section provides sample messages related to **Ping**.

4.4.1 Ping Command Request

The following is an example of the **Ping** element in a **Ping** command request.

```

<?xml version="1.0" encoding="utf-8"?>
<Ping xmlns="Ping:">
    <Folders>
        ...
    </Folders>
</Ping>

```

4.4.2 Ping Command Response

4.4.2.1 Typical Response

The following example shows a typical response to a **Ping** command request, when the heartbeat interval that was specified by the client has expired and there were no changes in any of the specified folders.

```

<Ping xmlns="Ping:">
    <Status>1</Status>

```

```
</Ping>
```

4.4.2.2 Changes Found

The following response message shows that changes have occurred in two folders that were being monitored. The client then synchronizes the specified folders. Do not reissue the next **Ping** command until the folders have been synchronized.

```
<Ping xmlns="Ping:">
  <Status>2</Status>
  <Folders>
    <Folder>1234</Folder>
    <Folder>5678</Folder>
  </Folders>
</Ping>
```

4.4.2.3 HeartbeatInterval Error

The following example shows a response to a **Ping** command request that specified a heartbeat interval outside the acceptable range. The returned heartbeat interval is either the minimum or maximum allowed value. The client compares the requested interval with the returned interval and determine whether the requested heartbeat interval was either too great or too small.

```
<Ping xmlns="Ping:">
  <Status>5</Status>
  <HeartbeatInterval>60</HeartbeatInterval>
</Ping>
```

4.4.2.4 Folder Error

The following example shows a response to a **Ping** command request where the number of folders that was specified was greater than the maximum number of folders that are allowed to be monitored. The maximum number of folders that are allowed to be monitored is returned in the **MaxFolders** element.

```
<Ping xmlns="Ping:">
  <Status>6</Status>
  <MaxFolders>200</MaxFolders>
</Ping>
```

4.5 Fetching E-Mail and Attachments by Using the ItemOperations Command

The **ItemOperations** command enables the client to retrieve **PIM** items and **attachments** (in addition to document library items and search results) outside the **Sync** command context.

These examples focus on retrieval of items and attachments, following a simple request and response model. The following figure shows the request and response model used in fetching e-mail and attachments.

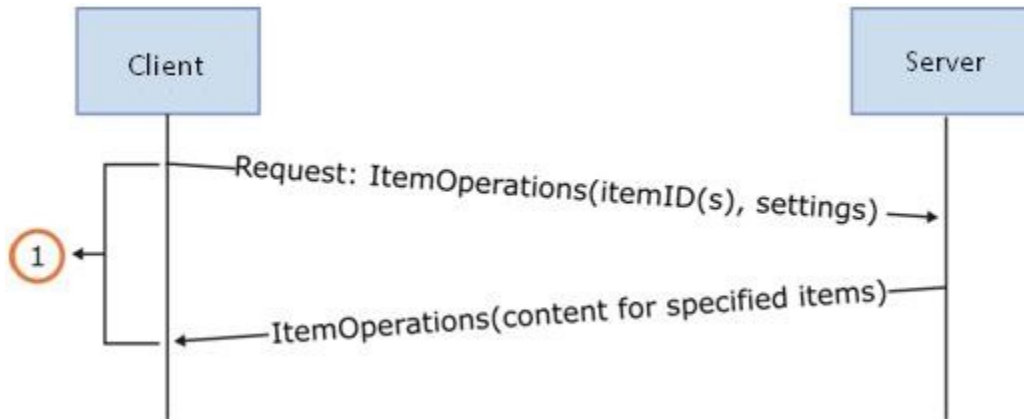


Figure 9: Fetching E-mail

4.5.1 Fetching an E-Mail Item

The following example shows the client retrieving an e-mail message by using the **ItemOperations** command.

Request

```
POST /Microsoft-Server-
ActiveSync?Cmd=ItemOperations&User=deviceuser&DeviceId=device1&
DeviceType=PocketPC HTTP/1.1
Content-Type: application/vnd.ms-sync.wbxml
MS-ASProtocolVersion: 12.1

<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns:airsync="AirSync:"
xmlns:airsyncbase="AirSyncBase:" xmlns="ItemOperations:">
  <Fetch>
    <Store>Mailbox</Store>
```

```

    <airsync:CollectionId>7</airsync:CollectionId>
    <airsync:ServerId>7:1</airsync:ServerId>
    <Options>
      <airsynibase:BodyPreference>
        <airsynibase:Type>1</airsynibase:Type>
        <airsynibase:TruncationSize>5120</airsynibase:TruncationSize>
        <airsynibase:AllOrNone>0</airsynibase:AllOrNone>
      </airsynibase:BodyPreference>
    </Options>
  </Fetch>
</ItemOperations>

```

Response

```

HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 409
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 12.1
Date: Tue, 08 May 2007 17:29:52 GMT

```

```

<?xml version="1.0" encoding="utf-8"?><ItemOperations
xmlns:airsync="AirSync:" xmlns:email="POOMMAIL:"
xmlns="ItemOperations:">
  <Status>1</Status>
  <Response>
    <Fetch>
      <Status>1</Status>
      <airsync:CollectionId>7</airsync:CollectionId>
      <airsync:ServerId>7:1</airsync:ServerId>
      <airsync:Class>Email</airsync:Class>
      <Properties>
        <email:To>"deviceuser" &lt;someone1@example.com&gt;</email:To>
        <email:Cc>"deviceuser3" &lt;someone3@example.com&gt;</email:Cc>

```

```

    <email:From>"deviceuser2" &lt;someone2@example.com&gt;
  </email:From>
  <email:Subject>Subject</email:Subject>
  <email:DateReceived>2007-05-08T17:29:07.890Z
</email:DateReceived>
  <email:DisplayTo>DeviceUserDisplayName</email:DisplayTo>
  <email:ThreadTopic>Subject</email:ThreadTopic>
  <email:Importance>1</email:Importance>
  <email:Read>0</email:Read>
  <airsyncbase:Body>
    <airsyncbase:Type>1</airsyncbase:Type>
    <airsyncbase:EstimatedDataSize>20
  </airsyncbase:EstimatedDataSize>
    <airsyncbase:Data>Body as plain text</airsyncbase:Data>
  </airsyncbase:Body>
  <email:MessageClass>IPM.Note</email:MessageClass>
  <email:InternetCPID>28591</email:InternetCPID>
  <email:Flag />
  <email:ContentClass>urn:content-classes:message
</email:ContentClass>
  <airsyncbase:NativeBodyType>1</airsyncbase:NativeBodyType>
</Properties>
</Fetch>
</Response>
</ItemOperations>

```

4.5.2 Fetching a MIME E-Mail Item

The following example shows the client retrieving a **MIME** e-mail message by using the **MIMESupport** option.

4.5.2.1 Request

```

<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns="ItemOperations:" xmlns:A="AirSync:"
xmlns:B="AirSyncBase:">

```

```

<Fetch>
  <Store>Mailbox</Store>
  <A:CollectionId>17</A:CollectionId>
  <A:ServerId>17:11</A:ServerId>
  <Options>
    <MIMESupport xmlns="AirSync:">1</MIMESupport>
    <B:BodyPreference>
      <B:Type>4</B:Type>
    </B:BodyPreference>
  </Options>
</Fetch>
</ItemOperations>

```

4.5.2.2 Response

```

<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns="ItemOperations:" xmlns:A="AirSync:"
xmlns:B="POOMMAIL:" xmlns:C="AirSyncBase:">
  <Status>1</Status>
  <Response>
    <Fetch>
      <Status>1</Status>
      <A:CollectionId>17</A:CollectionId>
      <A:ServerId>17:11</A:ServerId>
      <A:Class>Email</A:Class>
      <Properties>
        <B:To>"Mike Phipps" &lt;mike@contoso.com&gt;</B:To>
        <B:From>"Arlene Huff" &lt;arlene@contoso.com&gt;</B:From>
        <B:Subject>opaque s + e</B:Subject>
        <B:DateReceived>2007-02-01T06:42:46.015Z</B:DateReceived>
        <B:DisplayTo>Mike Phipps</B:DisplayTo>
        <B:ThreadTopic>opaque s + e</B:ThreadTopic>
        <B:Importance>1</B:Importance>
        <B:Read>1</B:Read>
        <C:Attachments>

```

```

    <C:Attachment>
      <C:DisplayName>smime.p7m</C:DisplayName>

    <C:FileReference>RgAAAAA4u8%2fWvU8lQ7GtLlC7V9V3BwCdyWYIRkOHRp2ozB%2f0DX
    QsAHgM%2bwAFAAA6pk60fqkEQbWH4Wm%2bnjh7AHgNBA%2bgAAAJ%3a0</C:FileReferen
    ce>

    <C:Method>1</C:Method>
    <C:EstimatedDataSize>9340</C:EstimatedDataSize>
  </C:Attachment>
</C:Attachments>
<C:Body>
  <C:Type>4</C:Type>
  <C:EstimatedDataSize>13813</C:EstimatedDataSize>
  <C:Data>Received: from contoso.com ([157.55.97.121])
    by contoso.com ([157.55.97.121]) with mapi;
    Wed, 31 Jan 2007 22:42:45 -0800
  From: Arlene Huff <arlene@contoso.com>;
  To: Mike Phipps <mike@contoso.com>;
  Content-Class: urn:content-classes:message
  Date: Wed, 31 Jan 2007 22:42:41 -0800
  Subject: opaque s + e
  Thread-Topic: opaque s + e
  Thread-Index: AcdFzCv5tyCXieBuTd2I5APpEvS+iQ==
  Message-ID:
    <3AA64EB47EA90441B587E169BE9E387B780D00C326@contoso.com>;
  Content-Language: en-US
  X-MS-Exchange-Organization-AuthAs: Internal
  X-MS-Exchange-Organization-AuthMechanism: 04
  X-MS-Exchange-Organization-AuthSource:
    contoso.com
  X-MS-Has-Attach: yes
  X-MS-Exchange-Organization-SCL: -1
  X-MS-TNEF-Correlator:
  acceptlanguage: en-US

```



```

Content-Type: application/x-pkcs7-mime; smime-type=enveloped-data;
name="smime.p7m"
Content-Disposition: attachment; filename="smime.p7m"
Content-Transfer-Encoding: base64
MIME-Version: 1.0

MIAGCSqGSIB3DQEHA6CAMIACAQAxggJEMIIbHgIBADCBhjB4MRMwEQYKCZImiZPyLQBGRY
DY29t
MRkwFwYKCZImiZPyLQBGRYJbWljcm9zb2Z0MRYwFAYKCZImiZPyLQBGRYGZXh0ZXN0MR0
wGwYK
CZImiZPyLQBGRYNamluZ2h1YWMwMURPTTEPMA0GA1UEAxMGVGVzdENBAgonJIo2AAAAA
HMA0G
(Large section of sample data removed)
    </C:Data>
  </C:Body>
  <B:MessageClass>IPM.Note.SMIME</B:MessageClass>
  <B:InternetCPID>20127</B:InternetCPID>
  <B:Flag/>
  <B:ContentClass>urn:content-
classes:message</B:ContentClass>
  <C:NativeBodyType>1</C:NativeBodyType>
</Properties>
</Fetch>
</Response>
</ItemOperations>

```

4.5.3 Fetching an E-Mail Item with a LongId

The following example shows the client retrieving an e-mail message by using **LongId**. First, use the **Search** command to get the **LongId** of the message, and then use the **Fetch** command with the **LongId** to retrieve the message.

The client sends the **Search** command request message, and it is searching for e-mails containing the text “Sales Totals”.

```

POST /Microsoft-Server-
ActiveSync?Cmd=Search&User=deviceuser&DeviceId=device1&DeviceType=Smart
Phone HTTP/1.1

```

```

Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1
<?xml version="1.0" encoding="utf-8"?>
<Search xmlns="Search:" xmlns:airsync="AirSync:"
xmlns:email="POOMMAIL:">
  <Store>
    <Name>Mailbox</Name>
    <Query>
      <And>
        <airsync:Class>Email</airsync:Class>
        <airsync:CollectionId>7</airsync:CollectionId>
        <FreeText>Sales Totals</FreeText>
      </And>
    </Query>
    <Options>
      <RebuildResults />
      <Range>0-4</Range>
    </Options>
  </Store>
</Search>

```

The server sends the **Search** command response message includes e-mail data for e-mail that contains the string “Sales Totals”. Included with the results is the **LongId** element.

```

HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 423
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 12.1
Date: Tue, 08 May 2007 17:42:07 GMT

```

```

<?xml version="1.0" encoding="utf-8"?><Search xmlns:airsync="AirSync:"
xmlns:email="POOMMAIL:" xmlns:airsyncbase="AirSyncBase:"
xmlns="Search:">

```

```

<Status>1</Status>
<Response>
  <Store>
    <Status>1</Status>
    <Result>
      <airsync:Class>Email</airsync:Class>
      <LongId>RgAAAACYWCHnyBZ%2fTq8buJFmR1EPBwBzyWfENpcEQ7zUNyaWwM4BAAAA8FxEAABzyWfENpcEQ7zUNyaWwM4BAAAA8HACAAAJ</LongId>
      <airsync:CollectionId>7</airsync:CollectionId>
      <Properties>
        <email:To>"deviceuser" &lt;someone1@example.com&gt;
      </email:To>
        <email:From>"deviceuser2" &lt;someone2@example.com&gt;
      </email:From>
        <email:Subject>Sales Totals for April</email:Subject>
        <email:DateReceived>2007-05-08T17:29:07.890Z
      </email:DateReceived>
        <email:DisplayTo>DeviceUserDisplayName</email:DisplayTo>
        <email:Read>1</email:Read>
        <airsyncbase:Body>
          <airsyncbase:Type>1</airsyncbase:Type>
          <airsyncbase:EstimatedDataSize>6
        </airsyncbase:EstimatedDataSize>
          <airsyncbase:Truncated>1</airsyncbase:Truncated>
        </airsyncbase:Body>
        <email:MessageClass>IPM.Note</email:MessageClass>
        <email:InternetCPID>28591</email:InternetCPID>
        <email:Flag />
        <email:ContentClass>urn:content-classes:message
      </email:ContentClass>
        <airsyncbase:NativeBodyType>1</airsyncbase:NativeBodyType>
      </Properties>
    </Result>
  </Store>
</Response>

```

```

    <Range>0-0</Range>
    <Total>1</Total>
  </Store>
</Response>
</Search>

```

The **Fetch** command request is now sent by the client, and includes the **LongId** retrieved by the **Search** command.

```

POST /Microsoft-Server-
ActiveSync?Cmd=ItemOperations&User=deviceuser&DeviceId=device1&
DeviceType=PocketPC HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1

```

```

<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns:airsync="AirSync:"
xmlns:airsyncbase="AirSyncBase:" xmlns="ItemOperations:">
  <Fetch>
    <Store>Mailbox</Store>
    <airsync:LongId>RgAAAACYWCHnyBZ%2fTq8bujFmR1EPBwBzyWfENpc
EQ7zUNYaWwM4BAAAA8FxEAABzyWfENpcEQ7zUNYaWwM4BAAAA8HA
CAAAJ</airsync:CollectionId>
    <Options>
      <airsyncbase:BodyPreference>
        <airsyncbase:Type>1</airsyncbase:Type>
      </airsyncbase:BodyPreference>
    </Options>
  </Fetch>
</ItemOperations>

```

The server sends the **Fetch** command response, which contains the complete e-mail for the specified message.

```

HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 409

```

Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 8.1
Date: Tue, 08 May 2007 17:29:52 GMT

```
<?xml version="1.0" encoding="utf-8"?><ItemOperations
xmlns:airsync="AirSync:" xmlns:email="POOMMAIL:"
xmlns="ItemOperations:">
  <Status>1</Status>
  <Response>
    <Fetch>
      <Status>1</Status>
      <airsync:CollectionId>7</airsync:CollectionId>
      <airsync:ServerId>7:1</airsync:ServerId>
      <airsync:Class>Email</airsync:Class>
      <Properties>
        <email:To>"deviceuser" &lt;someone1@example.com&gt;</email:To>
        <email:From>"deviceuser2" &lt;someone2@example.com&gt;
        </email:From>
        <email:Subject>Sales Totals for April</email:Subject>
        <email:DateReceived>2007-05-08T17:29:07.890Z
        </email:DateReceived>
        <email:DisplayTo>DeviceUserDisplayName</email:DisplayTo>
        <email:ThreadTopic>Subject</email:ThreadTopic>
        <email:Importance>1</email:Importance>
        <email:Read>1</email:Read>
        <airsyncbase:Body>
          <airsyncbase:Type>1</airsyncbase:Type>
          <airsyncbase:EstimatedDataSize>20
          </airsyncbase:EstimatedDataSize>
          <airsyncbase:Data>Income generated by the sales department
          in April can be attributed to the following...
          </airsyncbase:Data>
```

```

    </airsyncbase:Body>
    <email:MessageClass>IPM.Note</email:MessageClass>
    <email:InternetCPID>28591</email:InternetCPID>
    <email:Flag />
    <email:ContentClass>urn:content-classes:message
    </email:ContentClass>
    <airsyncbase:NativeBodyType>1</airsyncbase:NativeBodyType>
  </Properties>
</Fetch>
</Response>
</ItemOperations>

```

4.5.4 Fetching an Attachment

In the following example, the **Sync** command is used to synchronize a new message with an **attachment** to the client. Then, the **ItemOperations** command is used to retrieve the attachment.

In the **XML** scenario code, HTML strings are escaped by using **<** and **>**. However, as these values are passed over the wire, they are passed in their original HTML format, as **<** and **>**.

4.5.4.1 Sync Command Request

```

POST /Microsoft-Server-
ActiveSync?Cmd=Sync&User=deviceuser&DeviceId=device1&DeviceType=
PocketPC HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1
Content-Length: 106

```

```

<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns:airsyncbase="AirSyncBase:" xmlns="AirSync:">
  <Collections>
    <Collection>
      <Class>Email</Class>
      <SyncKey>1</SyncKey>
      <CollectionId>7</CollectionId>
    
```

```

        <DeletesAsMoves />
        <GetChanges />
    </Collection>
</Collections>
</Sync>

```

4.5.4.2 Sync Command Response

HTTP/1.1 200 OK

Cache-Control: private

Content-Length: 347

Content-Type: application/vnd.ms-sync

Server: Microsoft-IIS/6.0

MS-Server-ActiveSync: 8.1

Date: Tue, 08 May 2007 17:57:32 GMT

```

<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns:email="POOMMAIL:"
xmlns:airsyncbase="AirSyncBase:" xmlns="AirSync:">
    <Collections>
        <Collection>
            <Class>Email</Class>
            <SyncKey>2</SyncKey>
            <CollectionId>7</CollectionId>
            <Status>1</Status>
            <Commands>
                <Add>
                    <ServerId>7:1</ServerId>
                    <ApplicationData>
                        <email:To>"deviceuser" &lt;someone@example.com&gt;
                        </email:To>
                        <email:From>"deviceuser2" &lt;someone2@example.com&gt;
                        </email:From>
                        <email:Subject>Email with Attachment</email:Subject>
                    
```

```

<email:DateReceived>2007-05-08T17:57:22.890Z
</email:DateReceived>
<email:DisplayTo>deviceuser</email:DisplayTo>
<email:ThreadTopic>Email with Attachment
</email:ThreadTopic>
<email:Importance>1</email:Importance>
<email:Read>0</email:Read>
<airsynibase:Attachments>
  <airsynibase:Attachment>
    <airsynibase:DisplayName>ActiveSyncClient_
AcceptingMeetingRequest.JPG</airsynibase:DisplayName>
    <airsynibase:FileReference>7%3a1%3a0
    </airsynibase:FileReference>
    <airsynibase:Method>1</airsynibase:Method>
    <airsynibase:EstimatedDataSize>18790
    </airsynibase:EstimatedDataSize>
  </airsynibase:Attachment>
</airsynibase:Attachments>
<airsynibase:Body>
  <airsynibase:Type>2</airsynibase:Type>
  <airsynibase:EstimatedDataSize>58
  </airsynibase:EstimatedDataSize>
  <airsynibase:Truncated>1</airsynibase:Truncated>
  <airsynibase:Data>&lt;html&gt;&lt;hea</airsynibase:Data>
</airsynibase:Body>
<email:MessageClass>IPM.Note</email:MessageClass>
<email:InternetCPID>28591</email:InternetCPID>
<email:Flag />
<email:ContentClass>urn:content-classes:message
</email:ContentClass>
<airsynibase:NativeBodyType>1</airsynibase:NativeBodyType>
</ApplicationData>
</Add>

```



```
        </Commands>
    </Collection>
</Collections>
</Sync>
```

4.5.4.3 ItemOperation Command Request

```
POST /Microsoft-Server-
ActiveSync?Cmd=ItemOperations&User=deviceuser&DeviceId=device1&
DeviceType=PocketPC HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1

<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns:airsyncbase="AirSyncBase:"
xmlns="ItemOperations:">
    <Fetch>
        <Store>Mailbox</Store>
        <airsyncbase:FileReference>7%3a1%3a0</airsyncbase:FileReference>
    </Fetch>
</ItemOperations>
```

4.5.4.4 ItemOperation Command Response

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 1151
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 8.1
Date: Tue, 08 May 2007 17:28:33 GMT

<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns:airsyncbase="AirSyncBase:"
xmlns="ItemOperations:">
    <Status>1</Status>
```

314 of 336

4.6 Retrieving and Changing OOF Settings by Using the Settings Command

This section provides sample messages related to retrieving and changing OOF settings.

4.6.1 Retrieving OOF Settings

The client requests the user's OOF settings by using the **Get** command and specifying the type in which the client wants to have the OOF message formatted.

```
POST /Microsoft-Server-ActiveSync?Cmd=Settings&User=deviceuser&DeviceId=device1&DeviceType=PocketPC HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1
```

```
<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns="Settings:">
  <Oof>
    <Get>
      <BodyType>HTML</BodyType>
    </Get>
  </Oof>
</Settings>
```

The client requested the messages to be returned in HTML, so all OOF messages are formatted as such.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 203
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 8.1
Date: Tue, 08 May 2007 17:46:07 GMT
```

```
<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns="Settings:">
```

```

<Status>1</Status>
  <Oof>
    <Status>1</Status>
    <Get>
      <OofState>2</OofState>
      <StartTime>2007-05-08T10:45:51.250Z</StartTime>
      <EndTime>2007-05-11T10:45:51.250Z</EndTime>
      <OofMessage>
        <AppliesToInternal />
        <Enabled>1</Enabled>
        <ReplyMessage>Internal OOF Message</ReplyMessage>
        <BodyType>HTML</BodyType>
      </OofMessage>
      <OofMessage>
        <AppliesToExternalKnown />
        <Enabled>1</Enabled>
        <ReplyMessage>External OOF Message</ReplyMessage>
        <BodyType>HTML</BodyType>
      </OofMessage>
      <OofMessage>
        <AppliesToExternalUnknown /><Enabled>0</Enabled>
        <ReplyMessage>External OOF Message</ReplyMessage>
        <BodyType>HTML</BodyType>
      </OofMessage>
    </Get>
  </Oof>
</Settings>

```

4.6.2 Turning On the OOF Message

The client wants to turn on the **OOF** message. The client has to update the OOF status by using the **Set** command.

```

POST /Microsoft-Server-
ActiveSync?Cmd=Settings&User=deviceuser&DeviceId=device1&DeviceType=Poc
ketPC HTTP/1.1

```

Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1

```
<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns="Settings:">
  <Oof>
    <Set>
      <OofState>2</OofState>
      <OofMessage>
        <AppliesToInternal/>
        <Enabled>1</Enabled>
        <ReplyMessage> <html><head><meta
http-equiv="Content-Type" content="text/html;
charset=utf-8"><style>@font-face
{font-family:Verdana}p.MsoNormal, li.MsoNormal,
div.MsoNormal {margin:0in; margin-bottom:.0001pt;
font-size:10.0pt; font-family:Verdana} a:link,
span.MsoHyperlink {color:blue; text-
decoration:underline}a:visited,
span.MsoHyperlinkFollowed {color:purple;
text-decoration:underline} span.EmailStyle17
{font-family:Arial; color:windowtext} @page Section1
{margin:1.0in 1.25in 1.0in 1.25in} div.Section1 {}
</style> </head> <body lang="EN-US"
link="blue" vlink="purple"> <div class="Section1">
<p class="MsoNormal"><font size="2"
face="Arial"><span style="font-size:10.0pt;
font-family:Arial">I'll be out of the office
today.</span></font></p> </div>
</body></html></ReplyMessage>
      <BodyType>HTML</BodyType>
    </OofMessage>
  </OofMessage>
</Settings>
```

```

        <AppliesToExternalKnown/>
        <Enabled>0</Enabled>
    </OofMessage>
    <OofMessage>
        <AppliesToExternalUnknown/>
        <Enabled>0</Enabled>
    </OofMessage>
</Set>
</Oof>
</Settings>

```

The server responds with status, to indicate that OOF was successfully enabled.

```

HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 20
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 8.1
Date: Tue, 08 May 2007 17:45:09 GMT

```

```

<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns="Settings:">
    <Status>1</Status>
    <Oof>
        <Status>1</Status>
    </Oof>
</Settings>

```

4.6.3 Turning Off the OOF Message

The client wants to turn off the **OOF** message. The client has to update the OOF status by using the **Set** command.

```

POST /Microsoft-Server-
ActiveSync?Cmd=Settings&User=deviceuser&DeviceId=device1&DeviceType=Poc
ketPC HTTP/1.1
Content-Type: application/vnd.ms-sync

```

MS-ASProtocolVersion: 12.1

```
<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns="Settings:">
  <Oof>
    <Set>
      <OofState>0</OofState>
    </Set>
  </Oof>
</Settings>
```

The server responds with status, to indicate that OOF was successfully disabled.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 20
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 8.1
Date: Tue, 08 May 2007 17:45:09 GMT
```

```
<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns="Settings:">
  <Status>1</Status>
  <Oof>
    <Status>1</Status>
  </Oof>
</Settings>
```

4.7 Retrieving User Information by Using the Settings Command

The following example shows a user-information request and response.

Request

```
<Settings>
  <UserInformation>
    <Get/>
```

```
    </UserInformation>
</Settings>
```

Response

```
<Settings>
  <Status>1</Status>
  <UserInformation>
    <Status>1</Status>
    <Get>
      <EmailAddresses>
        <SMTPAddress>nameA@microsoft.com</SMTPAddress>
        <SMTPAddress>firstB.lastB@microsoft.com</SMTPAddress>
      </EmailAddresses>
    </Get>
  </UserInformation>
</Settings>
```

4.8 *Setting Device Information by Using the Settings Command*

The following example shows a device-information request and response.

Request

```
<Settings xmlns="Settings:">
  <DeviceInformation>
    <Set>
      <Model>...</Model>
      <IMEI>...</IMEI>
      <FriendlyName>...</FriendlyName>
      <OS>...</OS>
      <OSLanguage>...</OSLanguage>
      <PhoneNumber>...</PhoneNumber>
    </Set>
  </DeviceInformation>
</Settings>
```

Response

```
<Settings xmlns="Settings:">
```



```

    <Status>1</Status>
  <DeviceInformation>
    <Set>
      <Status>...</Status>
    </Set>
  </DeviceInformation>
</Settings>

```

4.9 *Setting a Device Password by Using the Settings Command*

The following example shows a device-password request and response.

Request

```

<Settings>
  <DevicePassword>
    <Set>
      <Password>bar</Password>
    </Set>
  </DevicePassword>
</Settings>

```

Response

```

<Settings>
  <Status>1</Status>
  <DevicePassword>
    <Set>
      <Status>...</Status>
    </Set>
  </DevicePassword>
</Settings>

```

4.10 *Accessing Documents on File Shares and URIs by Using the Search and ItemOperations Commands*

This section shows how to use the following process to retrieve an item from a Windows® SharePoint® Services or UNC site :

1. Issue a **Search** command, specifying the link to the folder. The server will return folder/item metadata, specifying the ID, file name, size, and so on for the item. For instructions on completing this task, see section 4.10.1.
2. Issue the **ItemOperations** command, specifying the ID from the item metadata. For instructions on completing this task, see section 4.10.2.

In issuing request 2, the following are considerations for the client pertaining to the size of the file to be retrieved:

- Does the client want to have the item content returned inline in the **WBXML**, or as separate body parts in the **HTTP** response? Using WBXML might be easier to implement, but might consume more memory on the device, depending on how the response parser on the device is implemented.
- What is the maximum number of bytes of item content that the client wants to have returned in one response? (Successive requests can be used to obtain the remaining content.)

The following figure shows the request and response pattern that is used to find and retrieve an item located on a Windows SharePoint Services or **UNC** site.

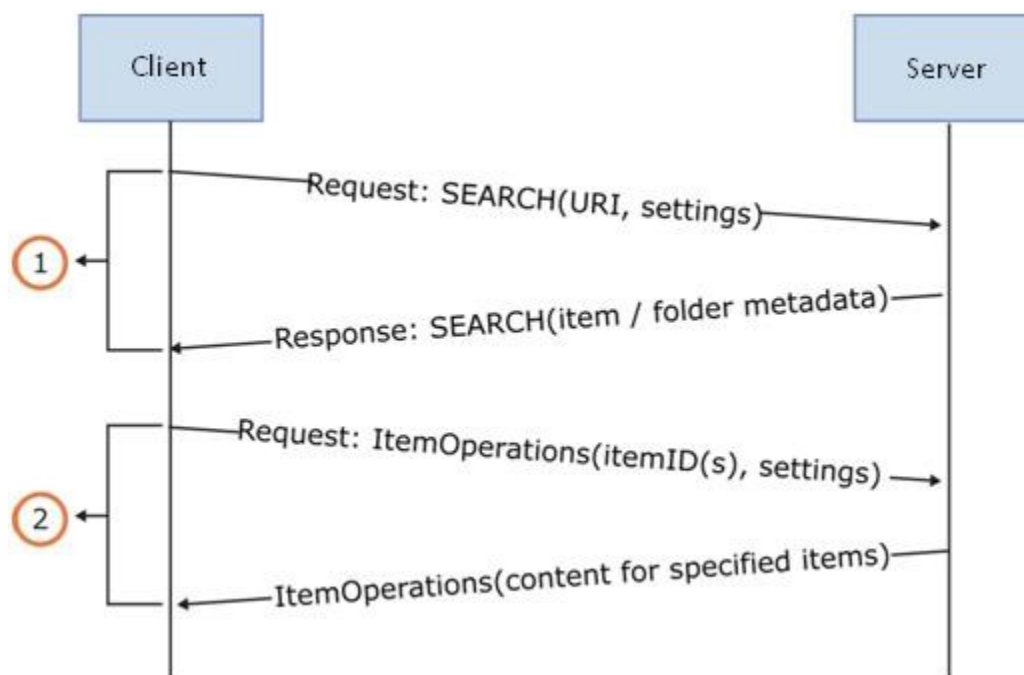


Figure 10: Finding and retrieving an item from a file share or UNC site

4.10.1 Issuing a Search for Item Metadata

As illustrated in the figure, the client first issues a search request to the server to retrieve metadata about the item (if the **URI** points to an item) or the items (if the **URI** points to a folder). The client then does the following:

- Indicates that the client is searching a document library store by using the **Name** element.
- Specifies the URI as the **Value** in an **EqualTo** query.
- Specifies the range of results that the client wants to have returned in the response.

In this case, the client is attempting to retrieve metadata for the files in a **UNC** share.

```
POST /Microsoft-Server-ActiveSync?Cmd=Search&User=deviceuser&DeviceId=device1&DeviceType=SmartPhone HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1
```

```
<?xml version="1.0" encoding="utf-8"?>
<Search xmlns:documentlibrary="DocumentLibrary:"
xmlns="Search:">
  <Store>
    <Name>DocumentLibrary</Name>
    <Query>
      <EqualTo>
        <documentlibrary:LinkId/>
        <Value>\\somehost\directory</Value>
      </EqualTo>
    </Query>
    <Options>
      <Range>0-999</Range>
    </Options>
  </Store>
</Search>
```

The response from the server contains the metadata for the folder and items. The very first node in the response is the top-level node, followed by its children (if any).

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 529
Content-Type: application/vnd.ms-sync
```

Server: Microsoft-IIS/6.0

MS-Server-ActiveSync: 8.1

Date: Tue, 08 May 2007 17:28:25 GMT

<?xml version="1.0" encoding="utf-8"?>

<Search xmlns:documentlibrary="DocumentLibrary:" xmlns="Search:">

<Status>1</Status>

<Response>

<Store>

<Status>1</Status>

<Result>

<Properties>

<documentlibrary:LinkId>\\somehost\directory

</documentlibrary:LinkId>

<documentlibrary:DisplayName>directory

</documentlibrary:DisplayName>

<documentlibrary:IsFolder>1

</documentlibrary:IsFolder>

<documentlibrary:CreationDate>2007-05-08T17:28:15.375Z

</documentlibrary:CreationDate>

<documentlibrary:LastModifiedDate>2007-05-08T17:28:15.406Z

</documentlibrary:LastModifiedDate>

<documentlibrary:IsHidden>0</documentlibrary:IsHidden>

</Properties>

</Result>

<Result>

<Properties>

<documentlibrary:LinkId>\\somehost\directory\resource

</documentlibrary:LinkId>

<documentlibrary:DisplayName>resource

</documentlibrary:DisplayName>

<documentlibrary:IsFolder>1</documentlibrary:IsFolder>

<documentlibrary:CreationDate>2004-03-02T12:34:56.123Z

</documentlibrary:CreationDate>

```

        <documentlibrary:LastModifiedDate>2005-04-03T12:34:56.345Z
      </documentlibrary:LastModifiedDate>
      <documentlibrary:IsHidden>0</documentlibrary:IsHidden>
    </Properties>
  </Result>
</Result>
  <Properties>
    <documentlibrary:LinkId>\\somehost\directory\TestFile.txt
  </documentlibrary:LinkId>
    <documentlibrary:DisplayName>TestFile.txt
  </documentlibrary:DisplayName>
    <documentlibrary:IsFolder>0</documentlibrary:IsFolder>
    <documentlibrary:CreationDate>2004-03-02T12:34:56.123Z
  </documentlibrary:CreationDate>
    <documentlibrary:LastModifiedDate>2005-04-03T12:34:56.345Z
  </documentlibrary:LastModifiedDate>
    <documentlibrary:IsHidden>0</documentlibrary:IsHidden>
    <documentlibrary:ContentLength>88
  </documentlibrary:ContentLength>
    <documentlibrary:ContentType>text/plain
  </documentlibrary:ContentType>
  </Properties>
</Result>
<Range>0-2</Range>
<Total>3</Total>
</Store>
</Response>
</Search>

```

4.10.2 Fetching an Item Based on Metadata

When a document library is used to provide item or folder metadata, the client can retrieve a file within a document library by using the **ItemOperations** command and specifying the **LinkId** of the item. In this example, the client also specifies that the client only requires bytes from 10 through 19 of the item returned in this request.

```
POST /Microsoft-Server-ActiveSync?Cmd=ItemOperations&User=deviceuser&DeviceId=device1&DeviceType=SmartPhone HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1
```

```
<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns:documentlibrary="DocumentLibrary:"
xmlns="ItemOperations:">
  <Fetch>
    <Store>DocumentLibrary</Store>
    <documentlibrary:LinkId>\\somehost\directory\
ActiveSyncDocumentFetch.txt</documentlibrary:LinkId>
    <Options>
      <Range>10-19</Range>
    </Options>
  </Fetch>
</ItemOperations>
```

The response from the server contains the requested item. The binary content of the file is Base64-encoded and is included in the **Data** element.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 167
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 8.1
Date: Tue, 08 May 2007 17:28:53 GMT
```

```
<?xml version="1.0" encoding="utf-8"?>
<ItemOperations xmlns:documentlibrary="DocumentLibrary:"
xmlns="ItemOperations:">
  <Status>1</Status>
  <Response>
```

```

<Fetch>
  <Status>1</Status>
  <documentlibrary:LinkId>\\somehost\directory\
ActiveSyncDocumentFetch.txt</documentlibrary:LinkId>
  <Properties>
    <Range>10-19</Range>
    <Total>26</Total>
    <Data>S0xNTk9QUVJTVA==</Data>
    <Version>2005-04-03T12:34:56.345Z</Version>
  </Properties>
</Fetch>
</Response>
</ItemOperations>

```

4.11 Searching for an Item in the Mailbox by Using the Search Command

This section provides sample messages used to perform keyword searches and forward search results for items in the **mailbox**.

4.11.1 Keyword Search

In the following example, the client is searching the Inbox in the **mailbox** by using the keyword "Presentation". The client has asked for the first 5 results and specified that it wants text bodies returned for the results. Note that the content of the **FreeText** element is not case-sensitive.

Request

```

POST /Microsoft-Server-
ActiveSync?Cmd=Search&User=deviceuser&DeviceId=device1&DeviceType=Smart
Phone HTTP/1.1
Content-Type: application/vnd.ms-sync
MS-ASProtocolVersion: 12.1

<?xml version="1.0" encoding="utf-8"?>
<Search xmlns="Search:" xmlns:airsync="AirSync:"
xmlns:email="POOMMAIL:">
<Store>

```

```

<Name>Mailbox</Name>
  <Query>
    <And>
      <airsync:Class>Email</airsync:Class>
      <airsync:CollectionId>7</airsync:CollectionId>
      <FreeText>Presentation</FreeText>
    </And>
  </Query>
  <Options>
    <RebuildResults />
    <Range>0-4</Range>
  </Options>
</Store>
</Search>

```

Response

```

HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 423
Content-Type: application/vnd.ms-sync
Server: Microsoft-IIS/6.0
MS-Server-ActiveSync: 12.1
Date: Tue, 08 May 2007 17:42:07 GMT

```

```

<?xml version="1.0" encoding="utf-8"?><Search xmlns:airsync="AirSync:"
xmlns:email="POOMMAIL:" xmlns:airsyncbase="AirSyncBase:"
xmlns="Search:">
<Status>1</Status>
  <Response>
    <Store>
      <Status>1</Status>
      <Result>
        <airsync:Class>Email</airsync:Class>

```



```

<LongId>RgAAAACYWCHnyBZ%2fTq8bujFmR1EPBwBzyWfENpcEQ7
zUNyaWwM4BAAAA8FxEAABzyWfENpcEQ7zUNyaWwM4BAAAA8HACAAAJ</LongId>
<airsync:CollectionId>7</airsync:CollectionId>
<Properties>
  <email:To>"deviceuser" &lt;someone1@example.com>;
</email:To>
  <email:From>"deviceuser2"&lt;someone1@example.com>;
</email:From>
  <email:Subject>Presentation</email:Subject>
  <email:DateReceived>2007-05-08T17:41:58.000Z
</email:DateReceived>
  <email:DisplayTo>DeviceUserDisplayName</email:DisplayTo>
  <email:Read>1</email:Read>
  <airsynibase:Body>
    <airsynibase:Type>1</airsynibase:Type>
    <airsynibase:EstimatedDataSize>6
    </airsynibase:EstimatedDataSize>
    <airsynibase:Truncated>1</airsynibase:Truncated>
  </airsynibase:Body>
  <email:MessageClass>IPM.Note</email:MessageClass>
  <email:InternetCPID>28591</email:InternetCPID>
  <email:Flag />
  <email:ContentClass>urn:content-classes:message
</email:ContentClass>
  <airsynibase:NativeBodyType>1</airsynibase:NativeBodyType>
</Properties>
</Result>
<Range>0-0</Range>
<Total>1</Total>
</Store>
</Response>
</Search>

```

4.11.2 Forward a Search Result

The client can then take the **LongId** for any given search result and forward the item.

```
POST Microsoft-Server-
ActiveSync?User=rich&DeviceId=6F24CAD599A5BF1A690246B8C68F
AE8D&DeviceType=PocketPC&Cmd=SmartForward&LongId=RgAAADdpC58
tdlTTY7tQhya20GHBwAiBQ4MPELpSI0QbZGxqTWyAAAA8%2bycAABNRh2Abh
XqSqcG01BXnsqBAAAB2ytlAAAI&SaveInSent=T
MS-ASProtocolVersion: 12.1
Content-Type: message/rfc822
X-MS-PolicyKey: 3942919513
```

----- Start of Body -----

```
MIME-Version: 1.0
content-class:
From:
Subject: FW: rx
Date: Thu, 27 Apr 2006 13:11:01 -0800
Importance: normal
X-Priority: 3
To: <rich@adventure-works.com>
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain; charset="utf-8"
```

-----Original Message-----

```
From: Shola Aluko <shola@adventure-works.com>
Sent: Tuesday, April 25, 2006 10:43 AM
To: Rich Haddock <rich@adventure-works.com>
Subject: rx
```

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 0
```

Date: Thu, 27 Apr 2006 20:11:11 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
MS-Server-ActiveSync: 12.1
Cache-Control: private

4.12 Resolving Recipients by Using the ResolveRecipient Command

This section provides sample messages related to the **ResolveRecipients** command.

4.12.1 RecipientCount Response for a GAL Entry

The following example shows two **recipients** that are being returned to the client. In the “Testers” **distribution list**, there are three recipients but only two have valid certificates.

```
<?xml version="1.0" encoding="utf-8"?>
<ResolveRecipients xmlns="ResolveRecipients:">
  <Status>1</Status>
  <Response>
    <To>Testers</To>
    <Status>1</Status>
    <RecipientCount>2</RecipientCount>
    <Recipient>
      <Type>1</Type>
      <DisplayName>Testers</DisplayName>
      <EmailAddress>testers@example.com</EmailAddress>
      <Certificates>
        <Status>1</Status>
        <CertificateCount>2</CertificateCount>
        <RecipientCount>3</RecipientCount>
        <MiniCertificate>AAAAAEfXfBA=</MiniCertificate>
      </Certificates>
    </Recipient>
    <Recipient>
      ...
    </Recipient>
```

```
</Response>
<ResolveRecipients>
```

4.12.2 Response for a Contact Entry

The following example shows a response for a **contact** entry.

```
<Response>
  <To>Contact</To>
  <Status>1</Status>
  <RecipientCount>1</RecipientCount>
  <Recipient>
    <Type>2</Type>
    <DisplayName>James Smith</DisplayName>
    <EmailAddress>jsmith@example.com</EmailAddress>
  </Recipient>
</Response>
```

5 Security

5.1 Security Considerations for Implementers

The device honors all policies sent down by the server, or send up the appropriate status codes indicating the non-success.

5.2 Index of Security Parameters

Security Parameter	Section
Provision Command	2.2.1.12
ValidateCert Command	2.2.1.20

6 Appendix A: Office/Exchange Behavior

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

- Office 2003 with Service Pack 3 applied
- Exchange 2003 with Service Pack 2 applied
- Office 2007 with Service Pack 1 applied

- Exchange 2007 with Service Pack 1 applied

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Office/Exchange behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies Office/Exchange does not follow the prescription.

<1> Section 2.2.1.1: For details about the Autodiscover service, see [AUTODISCOVER].

<2> Section 2.2.1.1.2.2: In Exchange 2007, the **Culture** element always returns "en:en", regardless of the culture that is sent by the client.

<3> Section 2.2.1.1.2.12: In Exchange 2007, this value is retrieved from the **MobileClientCertTemplateName** attribute of the ActiveSync virtual directory object.

<4> Section 2.2.1.1.2.15: In Exchange 2007, the URL is retrieved from the **ExternalUrl** attribute of the ActiveSync virtual directory object in Active Directory.

<5> Section 2.2.1.1.2.15: The URL value for the CertEnroll **Type** is retrieved from the **MobileClientCertificateAuthorityURL** attribute of the ActiveSync virtual directory object in Active Directory.

<6> Section 2.2.1.1.2.16: If the **Type** element value is MobileSync, then the **Url** element value is retrieved from the **ExternalUrl** attribute of the **ActiveSync** virtual directory object in Active Directory that is a child object of the server that services the user's mailbox. If the **Type** element value is CertEnroll, then the **Url** value is retrieved from the **MobileClientCertificateAuthorityURL** attribute of the **ActiveSync** virtual directory object in Active Directory that is a child object of the server that services the user's mailbox.

<7> Section 2.2.1.8.2.2: The location is indicated by a server ID (**ServerId** element) if an ActiveSync ID is being used to identify the item.

<8> Section 2.2.1.8.3.5: The location is indicated by a server ID (**ServerId** element) if an ActiveSync ID is being used to identify the item.

<9> Section 2.2.1.13.1.6: Some fields that are **ANR**-indexed in Active Directory by default in Exchange 2007 are as follows: **Name**, **Alias**, **Email**, **Office**. The ANR property set that can be indexed is definable by the administrator and it can be extended to include other fields.

<10> Section 2.2.1.13.2.12: Some fields that are ANR-indexed in Active Directory by default are as follows: **Name**, **Alias**, **Email**, **Office**. The ANR property set that can be indexed is definable by the administrator and can be extended to include other fields.

<11> Section 2.2.1.14: The **Search** command provides support for the following:

- The ability to search the Exchange mailbox

-
- The ability to browse the Microsoft Windows SharePoint Services technology Document Libraries or Universal Naming Convention (UNC) Shares

Mailbox and Windows SharePoint Services and UNC are represented as new stores within the **Search** command, and each has associated options, query, and schema.

<12> Section 2.2.1.14.1.2: While the **Or** keyword is supported in the protocol, Exchange 2007 does not support the **Or** keyword and will always return a SearchTooComplex, status 7. **And** or **Or** operations can only be used as the top level node (immediately under **Query**) – this is currently a server-side limitation.

<13> Section 2.2.1.14.1.2: Properties are indexed by Exchange 2007 Content Indexing (CI).

<14> Section 2.2.1.14.9: **And** operations can only be used as the top level node (immediately under **Query**) – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<15> Section 2.2.1.14.10: **Or** operations can only be used as the top level node (immediately under **Query**) – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<16> Section 2.2.1.14.1.11: The **Class** element cannot be under a **Query** or **Or** node, but rather be under the topmost **And** – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<17> Section 2.2.1.14.1.11: For an Exchange 2007 Mailbox search, the following classes are supported:

- Email
- Calendar
- Contacts
- Tasks

<18> Section 2.2.1.14.1.13: The **EqualTo** element cannot be under a **Query** or **Or** node, but rather be under the topmost **And** – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<19> Section 2.2.1.14.1.14: The **GreaterThan** element cannot be under a **Query** or **Or** node, but rather be under the topmost **And** – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<20> Section 2.2.1.14.1.15: The **LessThan** element cannot be under a **Query** or **Or** node, but rather be under the topmost **And** – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<21> Section 2.2.1.14.1.17: The **FreeText** element cannot be under a **Query** or **Or** node, but rather be under the topmost **And** – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<22> Section 2.2.1.14.1.18: The **CollectionId** element cannot be under a **Query** or **Or** node, but rather be under the topmost **And** – this is currently a server-side limitation. The server responds with SearchTooComplex, status 7.

<23> Section 2.2.1.16: In Microsoft Exchange Server 2007, the **Settings** command is used to perform the following operations:

- Get or set the out of office (OOF) settings for the user.
- Send device information to the computer that is running Exchange Server for display in the user and IT interfaces.
- Implement the device password—that is, the personal identification number (PIN)—recovery.
- Retrieve a list of a user's e-mail addresses.

<24> Multiple: Exchange 2007 requires that the reply message for unknown external and known external audiences be the same.

<25> Section 2.2.1.16.1.15: This information is also reflected in Microsoft Office Outlook® Web Access mobile device console.

<26> Multiple: Explicitly setting the SaveInSent parameter to 'F' currently results in an error.

<27> Section 2.2.1.1.2: In Exchange 2007, this information is retrieved from Active Directory directory service information on the ActiveSync virtual directory object.

<28> Section 4.1.6: In Exchange 2007, this page is configurable by using the 401-1.htm Web page that is installed in the Help subdirectory of the Autodiscover physical directory.

Index

- Introduction, 6
 - Applicability Statement, 10
 - Glossary, 6
 - Prerequisites/Preconditions, 10
 - Protocol Overview, 9
 - References, 7
 - Relationship to Other Protocols, 9
 - Standards Assignments, 10
 - Vendor-Extensible Fields, 10
 - Versioning and Capability Negotiation, 10
- Messages, 10
- Office/Exchange Behavior, 332
- Protocol Details, 273
- Protocol Examples, 283
- Security, 332