[MS-OXCSYNC]:
Mailbox Synchronization Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Copyright © 2011 Microsoft Corporation.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Minor editorial fixes.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	4.1.0	Minor	Updated the technical content.
05/05/2010	4.1.1	Editorial	Revised and edited the technical content.
08/04/2010	5.0	Major	Significantly changed the technical content.
11/03/2010	6.0	Major	Significantly changed the technical content.
03/18/2011	7.0	Major	Significantly changed the technical content.

Copyright © 2011 Microsoft Corporation.

Table of Contents

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

1 Introduction

Mailbox synchronization is the process of keeping synchronized versions of **Message objects**, **Folder objects**, and their related properties between the client and the server. Changes that are made on the client are synchronized to the server and vice versa.

During the mailbox synchronization process, there might be occasions when an object has been modified by both the client and the server. This results in a conflict. The mailbox synchronization process also includes mechanisms to resolve these situations.

Sections 1.8, 2, and 3 of this specification are normative and contain RFC 2119 language. Section 1.5 and 1.9 are also normative but cannot contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

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

GUID
property set
remote procedure call (RPC)

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

EntryID
Folder object
ICS state
local replica
mailbox
Message object
messaging object
offline
reminder
remote operation (ROP)
replica
store
stream
synchronization download context
synchronization upload context

The following terms are specific to this document:

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

Copyright © 2011 Microsoft Corporation.

[MS-OXCFOLD] Microsoft Corporation, "Folder Object Protocol Specification", June 2008.

[MS-OXCFXICS] Microsoft Corporation, "Bulk Data Transfer Protocol Specification", June 2008.

[MS-OXCROPS] Microsoft Corporation, "Remote Operations (ROP) List and Encoding Protocol Specification", June 2008.

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

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OXCMSG] Microsoft Corporation, "Message and Attachment Object Protocol Specification", June 2008.

[MS-OXCRPC] Microsoft Corporation, "Wire Format Protocol Specification", June 2008.

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

1.3 Overview

Figure 1 illustrates how the Mailbox Synchronization Protocol can be used to synchronize **messaging object** data. This data might be the properties of all descendant Folder objects (hierarchy sync) or the Message objects contained within the folder (message sync). Each box in the figure represents a logical state in the process of synchronizing the folder.

Copyright © 2011 Microsoft Corporation.

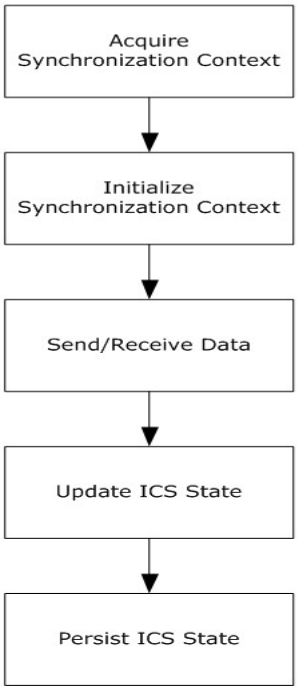

Figure 1: Synchronization states

The following list describes each state:

- Acquire Synchronization Context - To initiate the synchronization process, the client requests that the server acquire a synchronization download context and return a handle to the synchronization download context to the client.
- Initialize Synchronization Context – By using the synchronization download context, the client will start initialization. During initialization, the client might send the server the ICS state that was saved from a previous synchronization session. If the client passes an empty ICS state when initializing a download, all messaging object data that is relevant to that folder (and synchronization type) will be transmitted to the client.
- Send/Receive Data – By using the synchronization download context, data is now sent to or received from the server. The operations to accomplish this task depend on whether an upload or download is occurring.
- Update ICS State - When performing a download, the updated ICS state is sent to the client at the end of the data stream. If the client wants to update the state in the middle of the download, a new synchronization download context will have to be acquired from the server. Using this new context, the ICS state is downloaded from the server. When performing an upload, a

Copyright © 2011 Microsoft Corporation.

synchronization download context will have to be acquired from the server. By using the synchronization download context, the ICS state is downloaded from the server.

- Persist ICS State – By using whatever mechanism is prudent for the client to use, the ICS state is saved so that it can be used to initialize the synchronization context. This is required because the server does not maintain per-client state. It is worth noting that if the client initializes the next synchronization session without using the ICS state from the current session, the messaging object data that was processed in this synchronization session will be retransmitted in the next session.

1.4 Relationship to Other Protocols

The Mailbox Synchronization Protocol specification relies on an understanding of how to work with folders and messages as described in [MS-OXCMSG] and [MS-OXCFOLD]. Additionally, this specification relies on understanding the Bulk Data Transfer protocol as described in [MS-OXCFXICS].

1.5 Prerequisites/Preconditions

This protocol specification assumes the messaging client has previously acquired a handle to the folder for which it needs to synchronize data. Folder handle acquisition is specified in [MS-OXCFOLD].

1.6 Applicability Statement

One would implement this protocol to achieve detection of folder and message changes from a known state. In addition, these changes can be stored by the client in such a way as to constitute a **replica** of the server data. This replica might be implemented to facilitate creation, modification, or deletion of folders and messages while in an **offline** state.

1.7 Versioning and Capability Negotiation

The Bulk Data Transfer Protocol does not support detection of server capabilities. Instead, the client determines the server version to which it has connected. The client limits its behavior to the capabilities of the server version to which it has connected.

Several replication features were added for servers with a major version of eight (8). These features included partial item download and elimination of double-upload on send. For more information, see section 3.1.5.3.4 and [MS-OXCFXICS] section 3.3.5.5.4.2.1.2.

To determine the server version, inspect the version information that was returned when the connection to the server was established. For more information, see [MS-OXCRPC].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Copyright © 2011 Microsoft Corporation.

2 Messages

Synchronization is accomplished by using messages that are defined in [MS-OXCROPS] and [MS-OXCFXICS].

Copyright © 2011 Microsoft Corporation.

3 Protocol Details

When participating in synchronization, the client has several responsibilities in addition to the actual act of synchronization. These include: ID assignment, change tracking, conflict resolution, and ICS state storage.

3.1 Client 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 specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described.

3.1.1.1 Folder Scope

The synchronization portions of the Bulk Data Transfer Protocol are fundamentally associated with a folder. To initiate a synchronization action, the client must first acquire a handle to a folder with which to synchronize. When replicating contents, only the messages contained in that folder will be synchronized. When replicating hierarchy, all descendants of that folder will be synchronized. Hierarchy synchronization will only synchronize the folder structure of the hierarchy and the properties of the folders. The messages contained in the folder will not be synchronized when performing a hierarchy synchronization.

3.1.1.2 ICS State

The ICS state is a logical bookmark describing the point at which the last synchronization completed. This state is presented to a server in a subsequent synchronization session, enabling only changes made since that state was saved to be sent to the client. If the ICS state is not saved by the client, then all the messages in the folder at the time of synchronization will be sent to the client. Therefore, clients SHOULD save this state at the end of each synchronization session.

3.1.1.3 Synchronization Upload and Download Contexts

These contexts are the basis for all synchronization. A **synchronization upload context** is obtained from the server to perform the upload of changes. Conversely, synchronization download contexts are used to initiate the download of server changes to the client.

3.1.1.4 Data Stream

The synchronization portions of the Bulk Data Transfer Protocol are extensions to the core fast transfer portions of the Bulk Data Transfer Protocol. Upon configuration of a synchronization download context (and subsequent initialization using the persisted ICS state), the client will use **RopFastTransferSourceGetBuffer** ([MS-OXCROPS] section 2.2.12.3) to read a stream of data containing all the changes. This data stream is parsed according to [MS-OXCFXICS] in order to see the message and folder changes at the object level.

3.1.1.5 Client Specific Handling

Clients MAY choose to handle pieces of the data **stream** in specific ways to improve the efficiency or user experience. Examples of this from [MS-OXCFXICS] are progress information and bad item notifications. A possible implementation will use progress information to compute and inform the end user how much time is left in the synchronization session. Bad item notifications can be used to

Copyright © 2011 Microsoft Corporation.

move the problem items to another folder to avoid repeated errors on subsequent synchronization sessions.

3.1.2 Timers

None.

3.1.3 Initialization

3.1.3.1 Synchronization Setup Details

3.1.3.1.1 ID Reservation

When items are created in the **local replica**, IDs MUST be assigned in a way that is compatible with the server. It is acceptable for the underlying storage to assign an ID that is only for local use. However, all communications with the server must be based on server-compatible IDs. To facilitate this, a client MUST reserve server IDs and use those when new items are created locally. This is accomplished by using **RopGetLocalReplicaIds**, as specified in [MS-OXCFXICS]. Note that each time this call is made, ID ranges on the server are consumed. The client SHOULD NOT use **RopGetLocalReplicaIds** until the bulk of the previously reserved IDs are consumed. If **RopGetLocalReplicaIds** is called prematurely, resources will be unnecessarily consumed on the server.

The IDs received can be used for both folders and messages. The IDs MUST be consumed by adding one to the global counter (GLOBCNT) of the base ID returned from **RopGetLocalReplicaIds** for each ID used. The client MUST NOT reuse IDs. To avoid fragmenting the ID set, clients SHOULD use contiguous subranges for items within the same folder.

A client MAY use **RopSetLocalReplicaMidsetDeleted** to inform the server of IDs that were previously reserved that the client will not be using in a given folder, as specified in [MS-OXCFXICS].

3.1.3.1.1.1 Back-in-Time Detection

Due to the ID reservation and the fact that a client MUST NOT reuse IDs, the server MUST implement some mechanism to detect client rollback. One possible source of rollback is restoring a local replica from a backup after that replica was used past the point in time when it was backed up.

A sample implementation is a property on both the local replica and the server that stores a counter specific to each replica. When the replica connects to the server, it verifies that the counter is greater than or equal to the server counter. If the client counter is ever less than the server counter, rollback has occurred and that client replica will be abandoned.

3.1.3.1.1.2 Mailbox Validation

A client MUST NOT let a replica of one mailbox synchronize with a different mailbox. For this reason, the client MUST identify the mailbox that belongs to any given replica.

This can be accomplished by using the mailbox instance **GUID**, which is returned from **RopLogon**, as specified in [MS-OXCSTOR]. The mailbox would be associated with the local replica and compared to the mailbox GUID after **RopLogon** has been completed.

Copyright © 2011 Microsoft Corporation.

3.1.3.2 Synchronization Configuration Details

3.1.3.2.1 Acquiring Synchronization Contexts

If this is an upload synchronization, a handle to the synchronization upload context MUST be obtained by using **RopSynchronizationOpenCollector**, as specified in [MS-OXCFXICS]. If this is a download synchronization, a handle to the synchronization download context MUST be obtained using **RopSynchronizationConfigure**, as specified in [MS-OXCFXICS].

3.1.3.2.2 Synchronization Context Initialization

First, the client MUST use the persisted ICS state. If this is a first-time synchronization, the client MUST pass an empty ICS state.

To initialize a synchronization download context, the handle to the context MUST be used to pass the ICS state up to the server by using **RopSynchronizationUploadStateStreamBegin**, as specified in [MS-OXCFXICS]. Next, use **RopSynchronizationUploadStateStreamContinue**, as specified in [MS-OXCFXICS]. Finally, after the ICS state upload is complete, **RopSynchronizationUploadStateStreamEnd** MUST be used.

For a download synchronization, initialization MUST occur before **RopFastTransferSourceGetBuffer** (or any other ICS **ROP**) can be issued.

3.1.4 Higher-Layer Triggered Events

The following events apply to synchronization:

- Initialization
- Cancellation

Initialization occurs when a client initializes a synchronization session. This can occur as the result of user action, an application-specific timer, or a response to a notification from the server.

Cancellation occurs as a result of the user terminating the client or requesting that synchronization operations stop (for a transition to offline state, for example).

When a cancellation occurs on a download, the client SHOULD NOT update the ICS state by using **RopSynchronizationGetTransferState** if there is any data left unprocessed from the result of a **RopFastTransferSourceGetBuffer**, as specified in [MS-OXCFXICS]. This is because the server does not know if you successfully committed all items. It is possible to miss items if **RopSynchronizationGetTransferState** is used at a time when synchronization buffers are only partially processed.

Because the state cannot be updated safely, the items that were downloaded during the canceled synchronization session will be downloaded again during the next download. There are several ways to deal with this. One approach is checkpointing, which is defined in [MS-OXCFXICS] section 3.3.5.3.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Order of Operations

When performing synchronization, a client SHOULD perform upload before download. This allows conflicts to be resolved before data is received by the client. In this way, wire efficiency can be increased, as an additional post-resolution upload will not be required.

Copyright © 2011 Microsoft Corporation.

3.1.5.2 Upload Details

Client upload details are specified in [MS-OXCFXICS] section 3.3.5.5.4.

3.1.5.3 Download Details

When performing a download, a client MUST use **RopFastTransferSourceGetBuffer**, as specified in [MS-OXCFXICS]. The response will be a stream of data as previously mentioned in section 3.1.1.4.

The following diagram shows the download details.

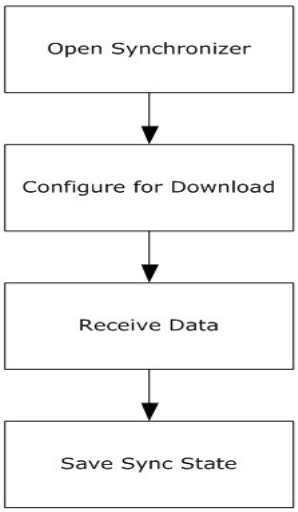

Figure 2: Download details

3.1.5.3.1 Response Processing and Expected Sequencing

Response processing occurs as defined in [MS-OXCFXICS]. There are several best practices for dealing with particular ICS primitives. These are specified in the following sections.

3.1.5.3.2 Property Deletes

Except as noted in section 3.1.5.3.4, servers that implement ICS do not have the ability to tell the client which properties on a folder/message have been deleted. As a result, the client MUST remove all properties and replace them with the complete set of downloaded properties supplied by the server. Failure to do so risks leaving items in an inconsistent state.

Copyright © 2011 Microsoft Corporation.

3.1.5.3.3 Skip Bad Item

When the server fails to process an item for download, the synchronization will generally fail. If, however, the client passes the **SendOptions RecoverMode** flag in the **RopSynchronizationConfigure** call, the server SHOULD place an **FxErrorInfo** primitive in the data stream, as specified in [MS-OXCFXICS] section 2.2.4.1.4. Using the folder ID (FID) and message ID (MID) from that primitive, the client SHOULD move the item into a folder that is not synchronized. In this way, the error will not continue to occur during each subsequent synchronization.

3.1.5.3.4 Partial Item Download

Users typically modify items in relatively minor ways. Flagging, adding **reminders**, or adding categories are common ways in which users modify items. Servers with a major version number of seven (7) or earlier would have synchronized this modification as a complete re-download of the whole item. Version eight (8) servers introduced partial item download, as specified in [MS-OXCFXICS] section 2.2.3.2.1.1.2. To get this behavior, the client MUST pass the **SendOptions PartialItem** in the **RopSynchronizationConfigure** call.

This method of download groups message properties into property groups that are defined by the server. When processing the data stream, the client will observe multiple sets of property group definitions (there might be multiple groups, as the server is free to define new groups at any time). Sometime later in the stream, the client will observe the index of the set of property groups that applies for the subsequent messages. This will give the client enough information to handle partial items. While processing the data stream for message changes, the client MAY observe a tag that informs the client that they are about to receive a message header for a partial change. After the header has been received, the client will receive the partial message data, as specified by [MS-OXCFXICS].

When processing a partial change, the client MUST NOT delete all properties on the local item, as specified in section 3.1.5.3.2. Instead, the client MUST only delete the properties on the local item that are defined in the group that is specified by the server.

3.1.5.3.5 Client Headers

It is often desirable to avoid downloading a message in its entirety when just a few properties will be sufficient for the user (for example, to, from, subject, received time). This is due to the fact that the user might have a slow connection or might delete most of their mail without reading it. The Bulk Data Transfer Protocol [MS-OXCFXICS] has a provision for this type of synchronization. To do this, the client MUST include the **property set** and specify the **SynchronizationFlag** *OnlySpecifiedProperties* parameter [MS-OXCFXICS] section 2.2.3.2.1.1) in a use of **RopSynchronizationConfigure**.

3.1.5.3.6 Client Header Completion

When clients download client headers, they will often have to turn those client headers into full items. To do this, a client MUST use **RopFastTransferSourceCopyMessages**, as specified in [MS-OXCFXICS]. In addition, the client MUST pass the **CopyFlags** *SendEntryId* ([MS-OXCFXICS] section 2.2.3.1.1.3.1) to **RopFastTransferSourceCopyMessages**. This will tell the server to pass the original **EntryID** of the item to the client. The messages will come down in the fast transfer datastream with an additional property, **PidTagOriginalEntryId**, containing the EntryID. This is required because the messages downloaded via **RopFastTransferSourceCopyMessages** are considered to be message copies and the FX stream contains no correlating identification properties. The **PidTagOriginalEntryId** property is one way the client can associate the full message item being downloaded from the server with the message header stored in the client.

Copyright © 2011 Microsoft Corporation.

3.1.5.3.7 Throttling

RopFastTransferSourceGetBuffer can return an error, ecServerBusy (0x8004010B). This indicates that the server is too busy at the current time to process the request. This MAY happen at high-load times, such as at the beginning of a work day after a holiday. When this occurs, the response to **RopFastTransferSourceGetBuffer** will contain a nonzero 'backoff'. This is a time in milliseconds, suggested by the server, that the client SHOULD wait before re-executing **RopFastTransferSourceGetBuffer**. Failure to wait can result in the server becoming overloaded and failing future requests.

3.1.5.3.8 Validation/Ignoring of Flags

When calling **RopSynchronizationConfigure**, as specified in [MS-OXCFXICS], several flag fields are validated differently by the server. The fast transfer send option (**SendOptions**) flags are strictly validated. This is to say that any unused bits must be set to 0 (zero). The server MUST fail any call where unused/unknown flags are set. The 'extra' flags parameter (SynchronizationExtraFlag) is not strictly validated. Unused/unknown bits MUST be ignored. This can be leveraged by clients to make version detection less necessary. This allows the client the flexibility of loosely asking for certain behavior in the ulExtra parameter. However, the client MUST detect the response in the download stream if the server has honored the requested behavior. This enables the client to request behavior from an older server that does not support the behavior without having to strictly check the version of the server.

3.1.5.3.9 ICS State Handling

After an upload finishes, the client SHOULD use **RopSynchronizationGetTransferState**, as specified in [MS-OXCFXICS]. Additionally, whenever the client has no outstanding data in the synchronization buffer, the client can use **RopSynchronizationGetTransferState**. The response will be the new ICS state. When a download successfully finishes, the ICS state will be included in the last buffer or buffers returned from **RopFastTransferSourceGetBuffer**, as specified in [MS-OXCFXICS].

The client SHOULD persist this state in the local data **store** and use it upon initializing the next synchronization action. Failure to do so will result in all items being returned to the client on the next download synchronization.

3.1.5.4 Client Conflict Resolution

A client can try to automatically resolve conflicting changes as defined in [MS-OXCFXICS] section 3.3.5.5.4.2.2.3, thereby avoiding the creation of a conflict resolve message.

When a client is unable to automatically resolve all conflicting changes, it SHOULD have a strategy to preserve alternate versions of messages, which SHOULD be made accessible to the user in case they prefer the alternate version.

A client can, for example, pick a "winner" message based on **PidTagLastModificationTime** and leave this in place of the modified item. The other version of the message, deemed the "loser," can be moved to a folder that contains previous versions of conflicting messages. If a strategy similar to this is implemented, these items SHOULD be linked by using the **PidTagConflictItems** property.

3.1.6 Timer Events

None.

Copyright © 2011 Microsoft Corporation.

3.1.7 Other Local Events

3.1.7.1 Connectivity Loss

If the connection is lost or any other recoverable error takes place, it will not be possible to update the ICS state. Because the state cannot be updated, the items that were downloaded during the aborted synchronization session will be re-downloaded during the next session. This can be dealt with in several ways. One way is checkpointing, which is defined in [MS-OXCFXICS] section 3.3.5.3.

3.2 Server Details

The server details for synchronization are specified in [MS-OXCFXICS].

Copyright © 2011 Microsoft Corporation.

4 Protocol Examples

Copyright © 2011 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the Mailbox Synchronization Protocol. General security considerations that pertain to the underlying **RPC**-based transport apply as described in [MS-OXCROPS].

5.2 Index of Security Parameters

None.

Copyright © 2011 Microsoft Corporation.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

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

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

Copyright © 2011 Microsoft Corporation.

7 Change Tracking

This section identifies changes that were made to the [MS-OXCSYNC] protocol document between the November 2010 and March 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

Copyright © 2011 Microsoft Corporation.

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Added information about which sections of the specification are normative and can contain RFC 2119 language.	Y	New content added for template compliance.

Copyright © 2011 Microsoft Corporation.

8 Index

Copyright © 2011 Microsoft Corporation.