[MS-OXIMAP4]: Internet Message Access Protocol Version 4 (IMAP4) Extensions

---

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 Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Updated references.
12/03/2008	1.03		Minor editorial fixes.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	3.1.0	Minor	Updated the technical content.
02/10/2010	3.2.0	Minor	Updated the technical content.
05/05/2010	3.2.1	Editorial	Revised and edited the technical content.
08/04/2010	4.0	Major	Significantly changed the technical content.
11/03/2010	4.1	Minor	Clarified the meaning of the technical content.
03/18/2011	5.0	Major	Significantly changed the technical content.
08/05/2011	5.1	Minor	Clarified the meaning of the technical content.
10/07/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	6.0	Major	Significantly changed the technical content.
04/27/2012	6.1	Minor	Clarified the meaning of the technical content.
07/16/2012	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	6.2	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

The Internet Message Access Protocol Version 4 (IMAP4) Extensions provide an authentication mechanism based on the **NT LAN Manager (NTLM) Authentication Protocol**, a **delegate access** mechanism to allow a **delegate** to access a **delegator's** mailbox, and support for the IMAP UIDPLUS extension described in [RFC4315].

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

Augmented Backus-Naur Form (ABNF)
connection-oriented NTLM
domain
Hypertext Transfer Protocol (HTTP)
NT LAN Manager (NTLM) Authentication Protocol
user principal name (UPN)

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

base64 encoding
delegate
delegate access
delegator
Internet Message Access Protocol - Version 4 (IMAP4)
NTLM message
NTLM software

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

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.

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification".

[RFC1731] Myers, J., "IMAP4 Authentication Mechanisms", RFC 1731, December 1994, http://www.rfc-editor.org/rfc/rfc1731.txt

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

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL – VERSION 4rev1", RFC 3501, March 2003, http://www.rfc-editor.org/rfc/rfc3501.txt

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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

[RFC822] Crocker, D.H., "Standard for ARPA Internet Text Messages", STD 11, RFC 822, August 1982, http://www.ietf.org/rfc/rfc0822.txt

1.2.2 Informative References

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

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

[MS-OXPROTO] Microsoft Corporation, "Exchange Server Protocols System Overview".

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.ietf.org/rfc/rfc4648.txt

1.3 Overview

The **IMAP4** Extensions are composed of three distinct extensions:

- The Internet Message Access Protocol - Version 4 (IMAP4) NTLM extension
- The IMAP4 delegate access extension
- The IMAP UIDPLUS extension

The IMAP4 NTLM extension enables a client to authenticate to a server using NTLM authentication. It allows the client to send an **NTLM message** over a standard IMAP4 connection and the server to send a response indicating the success or failure of the authentication.

The IMAP4 delegate access extension enables a client to access a mailbox on the server as a user other than the mailbox owner. This enables client access in the scenario where the mailbox owner has granted delegate access to their mailbox.

The IMAP UIDPLUS extension described in [RFC4315] enables a client to selectively remove messages from the client.

1.4 Relationship to Other Protocols

The IMAP4 NTLM extension uses the IMAP4 AUTHENTICATE extension mechanism, described in [RFC1731], and is an embedded protocol. Unlike standalone application protocols, such as Telnet or **HTTP**, packets for this extension are embedded in IMAP4 commands and server responses.

The IMAP4 NTLM extension specifies only the sequence in which a client and a server are required to exchange NTLM messages to successfully authenticate the client to the server. It does not specify how the client obtains NTLM messages from the local **NTLM software** or how the server processes NTLM messages. The client and server implementations depend on the availability of an implementation of NTLM, as described in [MS-NLMP], to obtain and process NTLM messages and on the availability of **base64 encoding** and decoding mechanisms, as described in [RFC4648], to encode and decode the NTLM messages that are embedded in IMAP4 packets.

For conceptual background information and overviews of the relationships and interactions between this and other protocols, see [MS-OXPROTO].

1.5 Prerequisites/Preconditions

Clients and servers require access to an implementation of NTLM, as described in [MS-NLMP], that is capable of supporting **connection-oriented NTLM**.

1.6 Applicability Statement

The IMAP4 NTLM extension is applicable to scenarios where both the client and the server have access to NTLM software and NTLM authentication is desired.

The IMAP4 delegate access extension is applicable to scenarios where IMAP4 is used to access a mailbox owned by another user.

The IMAP UIDPLUS extension is applicable to scenarios where clients require greater control over which messages are removed from the server.

1.7 Versioning and Capability Negotiation

This specification covers versioning issues in the following areas:

- **Security and Authentication Methods:** The IMAP4 NTLM extension supports the NTLMv1 and NTLMv2 authentication methods, as described in [MS-NLMP].
- **Capability Negotiation:** IMAP4 does not support negotiation of which version of NTLM to use. Instead, the NTLM version has to be configured on both the client and the server prior to authentication. NTLM version mismatches are handled by the NTLM implementation, and not by IMAP4.

The client discovers whether the server supports NTLM authentication by sending the IMAP4 **CAPABILITY** command. The server responds with a list of supported features, among which authentication mechanisms are listed. If NTLM is supported, the server includes the word "AUTH=NTLM" in the list. The messages involved are described in section 2.2.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

These extensions use standard IANA port assignments for IMAP4, as listed in the following table. Port mapping is configurable so that nondefault values can be used.

Parameter	Value	Reference
IANA assigned port for IMAP	143	http://www.iana.org/assignments/port-numbers
IANA assigned port for IMAP4 over TLS/SSL	993	http://www.iana.org/assignments/port-numbers

2 Messages

2.1 Transport

The IMAP4 Extensions do not establish transport connections. Instead, messages are encapsulated in IMAP4 commands and responses.

2.2 Message Syntax

2.2.1 IMAP4 NTLM Extension Messages

The IMAP4 NTLM extension extends both the IMAP4 **AUTHENTICATE** command requests and responses and the IMAP4 **CAPABILITY** command responses. The **AUTHENTICATE** command extensibility framework is specified in [RFC1731].

Message syntax is shown in **Augmented Backus-Naur Form (ABNF)**, as specified in [RFC5234]. The ABNF rules specified here extend the ABNF rules specified in [RFC3501] section 9. All human readable strings are arbitrary and do not affect protocol functionality.

```
NTLM_auth_type                                = "NTLM"
IMAP4_AUTHENTICATE_NTLM_Initiation_Command    = tag AUTHENTICATE NTLM CRLF
IMAP4_AUTHENTICATE_NTLM_Supported_Response    = "+" CRLF
IMAP4_AUTHENTICATE_NTLM_Unsupported_Response  = tag BAD text CRLF
IMAP4_AUTHENTICATE_NTLM_Cancellation_Command  = "*" SP CRLF
IMAP4_AUTHENTICATE_NTLM_Cancelled_Response    = tag NO "The AUTH protocol exchange was
                                                canceled by the client." CRLF
IMAP4_AUTHENTICATE_NTLM_Blob_Command          = base64-encoded-NTLM-Message CRLF
IMAP4_AUTHENTICATE_NTLM_Blob_Response         = "+" SP base64-encoded-NTLM-Message CRLF
IMAP4_AUTHENTICATE_NTLM_Succeeded_Response    = tag OK "AUTHENTICATE completed." CRLF
IMAP4_AUTHENTICATE_NTLM_Fail_Response         = tag NO text CRLF
```

2.2.2 IMAP4 Delegate Access Extension Messages

The IMAP4 delegate access extension extends the **LOGIN** command, as specified in [RFC3501] section 6.2.3. Specifically, it extends the **user name** argument of the **LOGIN** command so that a delegate and a delegator can be specified in the login string. This extension only affects the arguments of the **LOGIN** command and does not change the specification of the **LOGIN** command in [RFC3501].

There are four formats for the **user name** argument when using delegate access with IMAP4. The message syntax for the four formats is shown in ABNF.

```
domain                      = 1*VCHAR  ; The name of the user's domain
delegateuseralias           = 1*VCHAR  ; The delegate's e-mail alias
delegateuserupn             = 1*VCHAR  ; The delegate's UPN
principaluseralias          = 1*VCHAR  ; The principal's e-mail alias
principaluserupn            = 1*VCHAR  ; The principal's UPN
password                    = 1*VCHAR  ; The delegate's password

delegate_spec               = (domain "/" delegateuseralias) / delegateuserupn
principal_spec              = principaluseralias / principaluserupn
IMAP4_DELEGATE_LOGIN_Command = "LOGIN" SP delegate_spec "/" principal_spec password
```

The "domain" part of the login string represents the delegate's **domain**.

The "delegateuserupn" part of the login string represents the **user principal name (UPN)** of the delegate, which is composed of the user's identifier and domain, as specified in [RFC822] section 6.1.

The "delegateuseralias" part of the login string represents the e-mail alias of the delegate.

The "principaluserupn" part of the login string represents the UPN of the primary account, which is composed of the primary account's identifier and domain, as specified in [RFC822] section 6.1.

The "principaluseralias" part of the login string represents the e-mail alias of the primary account.

2.2.3 IMAP UIDPLUS Extension Messages

The syntax for IMAP UIDPLUS extension messages is specified in [RFC4315].

3 Protocol Details

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 document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.1.1.1 IMAP4 NTLM Extension State Model

The following figure shows the client IMAP4 NTLM extension state model.

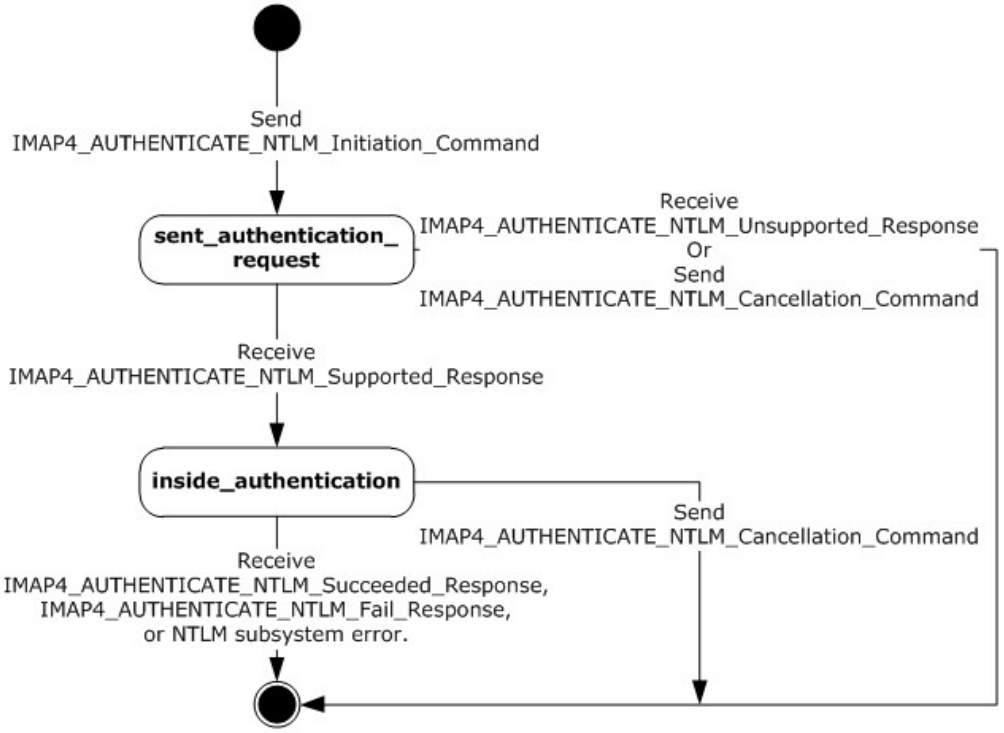

Figure 1: Client IMAP4 NTLM state model

The abstract data model for IMAP4 NTLM extension has the following states:

1. **Start**: State of the client before the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message has been sent.

2. **sent_authentication_request**: State of the client after the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message has been sent.

3. **inside_authentication**: State that is entered by a client after it has received an **IMAP4_AUTHENTICATE_NTLM_Supported_Response** message. In this state, the client initializes the NTLM subsystem and performs the following steps:

- Encapsulates the NTLM message, returned by the NTLM subsystem, into an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message and sends the message to the server. Waits for a response from the server.
- De-encapsulates the received **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message data (if any) from the server and converts it to NTLM message data.
- Passes the NTLM message data to the NTLM subsystem.
- Encapsulates the NTLM authenticate message, returned by the NTLM subsystem, into an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message.
- Sends the **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message to the server.

The **inside_authentication** state terminates when:

- An **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response** or **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message is received.
- Any failure is reported by the NTLM subsystem.

4. **completed_authentication**: State of the client on exiting the **inside_authentication** or the **sent_authentication_request** state. The rules for exiting the **inside_authentication** state are defined in section 3.1.5.1.4 and section 3.1.5.1.5. The behavior of IMAP4 in this state is outside the scope of this specification.

3.1.1.2 NTLM Subsystem Interaction

During the **inside_authentication** phase, the IMAP4 client invokes the NTLM subsystem and uses connection-oriented NTLM, as specified in [MS-NLMP].

All NTLM messages are encapsulated as specified in section 2.2.1. The data model, internal states, and sequencing of NTLM messages are specified in greater detail in [MS-NLMP].

1. The client initiates the authentication by invoking NTLM, after which NTLM will return the NTLM **NEGOTIATE_MESSAGE** message to be sent to the server.
2. Subsequently, the exchange of NTLM messages goes on as defined by NTLM, with the client encapsulating the NTLM messages before sending them to the server, and de-encapsulating IMAP4 messages to obtain the NTLM message before giving it to NTLM.
3. NTLM completes authentication, either successfully or unsuccessfully, as follows:
   - The server sends the **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response** to the client. On receiving this message, the client transitions to the **completed_authentication** state and MUST treat the authentication attempt as successful.
   - The server sends the **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message to the client. On receiving this message, the client transitions to the **completed_authentication** state and MUST treat the authentication attempt as failed.
   - Failures reported from the NTLM subsystem (which can occur for any reason, including incorrect data being passed in or implementation-specific errors) can be reported to the client by the NTLM subsystem. If the NTLM subsystem returns any failure status, the failure status MUST trigger the client to transition to the **completed_authentication** state.

3.1.2 Timers

None.

3.1.3 Initialization

Before attempting NTLM authentication via the IMAP4 NTLM extension, the client SHOULD send a **CAPABILITY** command, as specified in [RFC3501] section 6.1.1. If the server response does not contain a capability name that equals "AUTH=NTLM", the client SHOULD NOT attempt to use NTLM authentication.

3.1.4 Higher-Layer Triggered Events

When the client cancels authentication, it sends an **IMAP4_AUTHENTICATE_NTLM_Cancellation_Command** message to the server, as specified in section 2.2.1

When the client accesses a delegator's mailbox, it sends an **IMAP4_DELEGATE_LOGIN_Command** to the server, as specified in section 2.2.2.

3.1.5 Message Processing Events and Sequencing Rules

Message processing events and sequencing rules are divided into the following three categories:

- Receiving IMAP4 NTLM extension messages (section 3.1.5.1)
- Receiving IMAP4 delegate access extension messages (section 3.1.5.2)
- Receiving IMAP UIDPLUS extension messages (section 3.1.5.3)

3.1.5.1 Receiving an IMAP4 NTLM Extension Message

The IMAP4 NTLM extension is driven by a series of message exchanges between an IMAP4 server and an IMAP4 client. The rules governing the sequencing of commands and the internal states of the client and server are defined by a combination of the rules defined in [RFC1731] and [MS-NLMP]. Section 3.1.1.1 and section 3.1.1.2 define how those rules govern IMAP4 authentication.

If the client receives a message that is not expected for its current state, the client MUST cancel the authentication process and transition to **completed_authentication** state.

3.1.5.1.1 Receiving an IMAP4_AUTHENTICATE_NTLM_Supported_Response Message

The expected state is **sent_authentication_request**.

On receiving an **IMAP4_AUTHENTICATE_NTLM_Supported_Response** message, a client MUST generate the first NTLM message by calling the NTLM subsystem. The NTLM subsystem then generates a **NEGOTIATE_MESSAGE** NTLM message, as specified in [MS-NLMP]. The client encodes the NTLM message with base64 encoding, encapsulates it in an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message, and sends it to the server.

The client changes state to **inside_authentication**.

3.1.5.1.2 Receiving an IMAP4_AUTHENTICATE_NTLM_Unsupported_Response Message

The expected state is **sent_authentication_request**.

On receiving an **IMAP4_AUTHENTICATE_NTLM_Unsupported_Response** message, a client MUST abort the NTLM authentication attempt and change state to **complete_authentication**.

3.1.5.1.3 Receiving an IMAP4_AUTHENTICATE_NTLM_Blob_Response Message

The expected state is **inside_authentication**.

On receiving an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message, a client MUST de-encapsulate it to obtain the embedded base64-encoded NTLM message, decode it, and pass it to the NTLM subsystem for processing.

If the NTLM subsystem is successful in handling the message, it returns an NTLM **AUTHENTICATE_MESSAGE** message. The client then encodes the NTLM message with base64 encoding, encapsulates it in an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message, and sends it to the server. The internal state of the client does not change.

If the NTLM subsystem encounters an error when the **CHALLENGE_MESSAGE** NTLM message from the **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message is handled, the client MUST treat the authentication attempt as a failed attempt and transition to **completed_authentication** state.

3.1.5.1.3.1 Error from NTLM

If the NTLM subsystem reports an error, the client MUST change its internal state to **completed_authentication** and consider the authentication to have failed. The client can then take any action it considers appropriate; these extensions do not mandate any specific course of action.

Typical actions are to try other IMAP4 commands that are not related to authentication or to disconnect the connection.

3.1.5.1.3.2 NTLM Reports Success and Returns an NTLM Message

If NTLM reports success, the NTLM message it returns MUST be encapsulated in an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message and sent to the server. No change occurs in the state of the client.

3.1.5.1.4 Receiving an IMAP4_AUTHENTICATE_NTLM_Succeeded_Response Message

The expected state is **inside_authentication**.

If the client receives an **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response** message, the client MUST change its internal state to **completed_authentication** and consider the authentication to have succeeded. The client can then take any action it considers appropriate. These extensions do not mandate any specific course of action.

3.1.5.1.5 Receiving an IMAP4_AUTHENTICATE_NTLM_Fail_Response Message

The expected state is **inside_authentication**.

If the client receives an **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message, the client MUST change its internal state to **completed_authentication** and consider the authentication to have failed. The client can then take any action it considers appropriate. These extensions do not mandate any specific course of action.

3.1.5.2 Receiving IMAP4 Delegate Access Extension Messages

The client SHOULD handle server responses to the **LOGIN** command as specified in [RFC3501].

3.1.5.3 Receiving IMAP UIDPLUS Extension Messages

The client SHOULD handle server responses to the **UID EXPUNGE** command as specified in [RFC4315].

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 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 described in this document.

3.2.1.1 IMAP4 NTLM Extension State Model

The following figure shows the server IMAP4 NTLM extension state model.

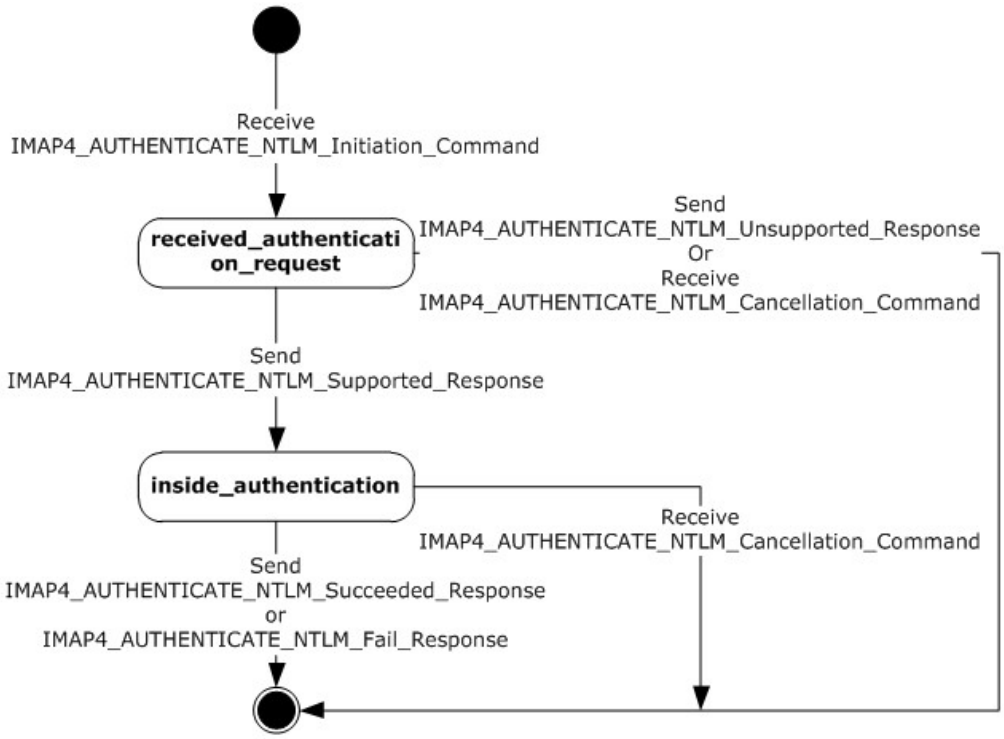

Figure 2: Server IMAP4 NTLM state model

The abstract data model for the IMAP4 NTLM extension has the following states:

1. **Start**: State of the server before the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message has been received.

2. **received_authentication_request**: State of the server after the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message has been received.

3. **inside_authentication**: State entered by a server after it has sent an **IMAP4_AUTHENTICATE_NTLM_Supported_Response** message. In this state, the server initializes the NTLM subsystem and performs the following steps:

   - Waits for a message from the client.
   - De-encapsulates the received **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message from the client and obtains the embedded NTLM message data.
   - Passes the NTLM message data to the NTLM subsystem.
   - Encapsulates the NTLM message returned by the NTLM subsystem into an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message.
   - Sends the **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message to the client.

   This state terminates when one of the following occurs:

   - The NTLM subsystem reports completion with either a success or failed authentication status, upon which it sends the client the **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response**

message or the **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message, as specified in [RFC1731].

- Any failure is reported by the NTLM subsystem.

4. **completed_authentication**: State of the server on exiting the **inside_authentication** or the **received_authentication_request** state. The rules for exiting the **inside_authentication** state are defined in section 3.2.5.1.2.2, section 3.2.5.1.2.3, section 3.2.5.1.2.4, and section 3.2.5.1.3. The behavior of IMAP4 in this state is outside the scope of this protocol.

3.2.1.2 NTLM Subsystem Interaction

During the **inside_authentication** state, the server invokes the NTLM subsystem and uses connection-oriented NTLM, as specified in [MS-NLMP].

The following is a description of how the IMAP4 NTLM extension uses NTLM. For more details, see [MS-NLMP].

1. The server, on receiving the NTLM **NEGOTIATE_MESSAGE** message, passes it to the NTLM subsystem and is returned the NTLM **CHALLENGE_MESSAGE** message, if the NTLM **NEGOTIATE_MESSAGE** message was valid.

2. Subsequently, the exchange of NTLM messages goes on as defined by NTLM, with the server encapsulating the NTLM messages that are returned by NTLM before sending them to the client.

3. When NTLM completes authentication, either successfully or unsuccessfully, the NTLM subsystem notifies the server.

   - On successful completion, the server MUST exit the **inside_authentication** state and enter the **completed_authentication** state and send the **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response** message to the client.

   - If a failure occurs due to an incorrect password error, as specified in [MS-NLMP], the server MUST enter the **completed_authentication** state and send the client an **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message.

   - If a failure occurs on the server due to any reason other than the incorrect password error, the server enters the **completed_authentication** state and sends the client an **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Message processing events and sequencing rules are divided into the following three categories:

- Receiving IMAP4 NTLM extension messages (section 3.2.5.1)
- Receiving IMAP4 delegate access extension messages (section 3.2.5.2)
- Receiving IMAP UIDPLUS extension messages (section 3.2.5.3)

3.2.5.1 Receiving an IMAP4 NTLM Extension Message

Servers SHOULD<1> support the IMAP4 NTLM extension. The IMAP4 NTLM extension is driven by a series of message exchanges between a server and a client. The rules governing the sequencing of commands and the internal states of the client and server are defined by a combination of the rules specified in [RFC1731] and [MS-NLMP]. Section 3.2.1.1 and section 3.2.1.2 define how those rules govern IMAP4 authentication.

If the server receives a message that is not expected for its current state, the server MUST cancel the authentication process and transition to **completed_authentication** state.

3.2.5.1.1 Receiving an IMAP4_AUTHENTICATE_NTLM_Initiation_Command Message

The expected state is **start**.

On receiving the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message, the server MUST reply with the **IMAP4_AUTHENTICATE_NTLM_Supported_Response** message, if it supports the IMAP4 NTLM extension, and change its state to the **inside_authentication** state.

If the server does not support the IMAP4 NTLM extension, it MUST respond with the **IMAP4_AUTHENTICATE_NTLM_Unsupported_Response** message, and change internal state to **completed_authentication**.

3.2.5.1.2 Receiving an IMAP4_AUTHENTICATE_NTLM_Blob_Command Message

The expected state is **inside_authentication**.

On receiving the **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message, the server de-encapsulates the message to obtain the embedded NTLM message and passes it to the NTLM subsystem. The server then takes action based on the response from the NTLM subsystem, as specified in the following table.

NTLM subsystem response	Server action
Success, returning an NTLM message	As specified in section 3.2.5.1.2.1
Success, indicating authentication complete	As specified in section 3.2.5.1.2.2
Failure, indicating user name or password incorrect	As specified in section 3.2.5.1.2.3
Failure for any reason other than incorrect user name or password	As specified in section 3.2.5.1.2.4

.

3.2.5.1.2.1 NTLM Returns Success, Returning an NTLM Message

If the server passes an **NEGOTIATE_MESSAGE** NTLM message to the NTLM subsystem, the NTLM subsystem returns an **NTLM CHALLENGE_MESSAGE** message. The server encapsulates the

CHALLENGE_MESSAGE message in an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message and sends it to the client. The server does not change its internal state.

3.2.5.1.2.2 NTLM Returns Success, Indicating Authentication Completed Successfully

If the server passes an **AUTHENTICATE_MESSAGE** NTLM message with the correct user name and password to the NTLM subsystem, the NTLM subsystem returns success. The server MUST return the **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response** message and change its internal state to **completed_authentication**.

3.2.5.1.2.3 NTLM Returns Status, Indicating User Name or Password Was Incorrect

If the server passes an **AUTHENTICATE_MESSAGE** NTLM message and the NTLM subsystem returns status that indicates that the user name or password was incorrect, the server MUST return the **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message and change its internal state to **completed_authentication**.

3.2.5.1.2.4 NTLM Returns a Failure Status, Indicating Any Other Error

If the server passes an **AUTHENTICATE_MESSAGE** NTLM message and the NTLM subsystem returns failure status that indicates an error other than the user name or password being incorrect, the server MUST return the **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message and change its internal state to **completed_authentication**.

3.2.5.1.3 Receiving an IMAP4_AUTHENTICATE_NTLM_Cancellation_Command Message

The expected states are **received_authentication_request** or **inside_authentication**.

On receiving the **IMAP4_AUTHENTICATE_NTLM_Cancellation_Command** message, the server MUST change to **completed_authentication** state and send an **IMAP4_AUTHENTICATE_NTLM_Cancelled_Response** message to the client.

3.2.5.2 Receiving an IMAP4 Delegate Access Extension Message

Servers SHOULD<2> support the IMAP4 delegate access extension. When the server receives the **IMAP4_DELEGATE_LOGIN_Command** message, it SHOULD take the following actions:

1. Authenticate the delegate using the delegate's alias or UPN and password.

2. Verify that the delegate has access to the delegator's mailbox.

If the authentication succeeds and the delegate has access to the delegator's mailbox, the server returns an OK response, as specified in [RFC3501] section 6.2.3. If the authentication does not succeed or the delegate does not have access to the delegator's mailbox, the server returns a NO response.

3.2.5.3 Receiving an IMAP UIDPLUS Extension Message

The server SHOULD<3> support the IMAP UIDPLUS extension. Message processing and sequencing rules are specified in [RFC4315]. The server SHOULD implement the response codes specified in [RFC4315] section 3 except for the UIDNOTSTICKY response code.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 IMAP4 NTLM Extension

The following sections describe operations used in a common scenario to illustrate the function of the IMAP4 NTLM extension.

4.1.1 Client Successfully Authenticating to a Server

The following example illustrates an IMAP4 NTLM extension scenario in which a client successfully authenticates to a server by using NTLM.

The client sends an **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message to the server.

```
AUTHENTICATE NTLM
```

The server sends the **IMAP4_AUTHENTICATE_NTLM_Supported_Response** message, indicating that it can perform NTLM authentication.

```
+
```

The client sends an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an **NEGOTIATE_MESSAGE** NTLM message that is encoded with base64 encoding.

```
TlRMTVNTUAABAAAAB4IIogAAAAAAAAAAAAAAAAAAAAAFASgKAAAADw==

00000000:4e 54 4c 4d 53 53 50 00 01 00 00 00 07 82 08 a2     NTLMSSP......,.¢
00000010:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
00000020:05 01 28 0a 00 00 00 0f ..(.....
```

The server sends an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message that contains an **CHALLENGE_MESSAGE** NTLM message that is encoded with base64 encoding.

```
+ TlRMTVNTUAACAAAAFAAUADgAAAAFgoqinziKqGYjdlEAAAAAAAAAAGQAZABMAAAABQ
LODgAAAA9UAEUAUwBUAFMARQBSAFYARQBSAAIAFABUAEUAUwBUAFMARQBSAFYARQBSAA
EAFABUAEUAUwBUAFMARQBSAFYARQBSAAQAFABUAGUAcwB0AFMAZQByAHYAZQByAAMAFA
BUAGUAcwB0AFMAZQByAHYAZQByAAAAAAA=

00000000:4e 54 4c 4d 53 53 50 00 02 00 00 00 14 00 14 00     NTLMSSP.........
00000010:38 00 00 00 05 82 8a a2 9f 38 8a a8 66 23 76 51     8....,Š¢Ÿ8Š¨f#vQ
00000020:00 00 00 00 00 00 00 00 64 00 64 00 4c 00 00 00     ........d.d.L...
00000030:05 02 ce 0e 00 00 00 0f 54 00 45 00 53 00 54 00     ..Î.....T.E.S.T.
00000040:53 00 45 00 52 00 56 00 45 00 52 00 02 00 14 00     S.E.R.V.E.R.....
00000050:54 00 45 00 53 00 54 00 53 00 45 00 52 00 56 00     T.E.S.T.S.E.R.V.
00000060:45 00 52 00 01 00 14 00 54 00 45 00 53 00 54 00     E.R.....T.E.S.T.
00000070:53 00 45 00 52 00 56 00 45 00 52 00 04 00 14 00     S.E.R.V.E.R.....
00000080:54 00 65 00 73 00 74 00 53 00 65 00 72 00 76 00     T.e.s.t.S.e.r.v.
00000090:65 00 72 00 03 00 14 00 54 00 65 00 73 00 74 00     e.r.....T.e.s.t.
000000a0:53 00 65 00 72 00 76 00 65 00 72 00 00 00 00 00     S.e.r.v.e.r....
```

The client sends an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an **AUTHENTICATE_MESSAGE** NTLM message that is encoded with base64 encoding.

```
TlRMTVNTUAADAAAAGAAYAGIAAAAYABgAegAAAAAAAABIAAAACAAIAEgAAAASABIAUAAA
AAAAAACSAAAABYKIogUBKAoAAAAPdQBzAGUAcgBOAEYALQBDAEwASQBFAE4AVABKMiQ4
djhcSgAAAAAAAAAAAAAAAAAAAAC7zUSgB0Auy98bRi6h3mwHMJfbKNtxmmo=

00000000:4e 54 4c 4d 53 53 50 00 03 00 00 00 18 00 18 00     NTLMSSP.........
00000010:62 00 00 00 18 00 18 00 7a 00 00 00 00 00 00 00     b.......z.......
00000020:48 00 00 00 08 00 08 00 48 00 00 00 12 00 12 00     H.......H.......
00000030:50 00 00 00 00 00 00 00 92 00 00 00 05 82 88 a2     P.......'....,^¢
00000040:05 01 28 0a 00 00 00 0f 75 00 73 00 65 00 72 00     ..(.....u.s.e.r.
00000050:4e 00 46 00 2d 00 43 00 4c 00 49 00 45 00 4e 00     N.F.-.C.L.I.E.N.
00000060:54 00 4a 32 24 38 76 38 5c 4a 00 00 00 00 00 00     T.J2$8v8\J......
00000070:00 00 00 00 00 00 00 00 00 00 bb cd 44 a0 07 40     ..........»ÍD .@
00000080:2e cb df 1b 46 2e a1 de 6c 07 30 97 db 28 db 71     .ËßF.¡Þl.0—Û(Ûq
00000090:9a 6a                                               šj
```

The server sends an **IMAP4_AUTHENTICATE_NTLM_Succeeded_Response** message.

```
1 OK AUTHENTICATE completed.
```

4.1.2 Client Unsuccessfully Authenticating to a Server

The following example illustrates an IMAP4 NTLM extension scenario in which an client tries NTLM authentication to a server and the authentication fails.

The client sends an **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** message to the server.

```
 AUTHENTICATE NTLM
```

The server sends the **IMAP4_AUTHENTICATE_NTLM_Supported_Response** message, indicating that it can perform NTLM authentication.

```
+
```

The client sends an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an **NEGOTIATE_MESSAGE** NTLM message that is encoded with base64 encoding.

```
TlRMTVNTUAABAAAAB4IIogAAAAAAAAAAAAAAAAAAAAAFASgKAAAADw==

00000000:4e 54 4c 4d 53 53 50 00 01 00 00 00 07 82 08 a2     NTLMSSP......,.¢
00000010:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
00000020:05 01 28 0a 00 00 00 0f                             ..(.....
```

The server sends an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message that contains an **CHALLENGE_MESSAGE** NTLM message that is encoded with base64 encoding.

```
+ TlRMTVNTUAACAAAAFAAUADgAAAAFgoqieUWd5ES4Bi0AAAAAAAAAAGQAZABMAA
AABQLODgAAAA9UAEUAUwBUAFMARQBSAFYARQBSAAIAFABUAEUAUwBUAFMARQBSAF
YARQBSAAEAFABUAEUAUwBUAFMARQBSAFYARQBSAAQAFABUAGUAcwB0AFMAZQByAH
```

```
YAZQByAAMAFABUAGUAcwB0AFMAZQByAHYAZQByAAAAAAA=

00000000:4e 54 4c 4d 53 53 50 00 02 00 00 00 14 00 14 00     NTLMSSP.........
00000010:38 00 00 00 05 82 8a a2 79 45 9d e4 44 b8 06 2d     8....,Š¢yE•äD¸.-
00000020:00 00 00 00 00 00 00 00 64 00 64 00 4c 00 00 00     ........d.d.L...
00000030:05 02 ce 0e 00 00 00 0f 54 00 45 00 53 00 54 00     ..Î.....T.E.S.T.
00000040:53 00 45 00 52 00 56 00 45 00 52 00 02 00 14 00     S.E.R.V.E.R.....
00000050:54 00 45 00 53 00 54 00 53 00 45 00 52 00 56 00     T.E.S.T.S.E.R.V.
00000060:45 00 52 00 01 00 14 00 54 00 45 00 53 00 54 00     E.R.....T.E.S.T.
00000070:53 00 45 00 52 00 56 00 45 00 52 00 04 00 14 00     S.E.R.V.E.R.....
00000080:54 00 65 00 73 00 74 00 53 00 65 00 72 00 76 00     T.e.s.t.S.e.r.v.
00000090:65 00 72 00 03 00 14 00 54 00 65 00 73 00 74 00     e.r.....T.e.s.t.
000000a0:53 00 65 00 72 00 76 00 65 00 72 00 00 00 00 00     S.e.r.v.e.r.....
```

The client sends an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an **AUTHENTICATE_MESSAGE** NTLM message that is encoded with base64 encoding.

```
TlRMTVNTUAADAAAAGAAYAGIAAAAYABgAegAAAAAAAABIAAAACAAIAEgAAAASABIA
UAAAAAAAAACSAAAABYKIogUBKAoAAAAPdQBzAGUAcgBOAEYALQBDAEwASQBFAE4A
VAAOarJ6lZ5ZNwAAAAAAAAAAAAAAAAAAAAACD9mD8jmWs4FkZe59/nNb1cF2HkL0C
GZw=

00000000:4e 54 4c 4d 53 53 50 00 03 00 00 00 18 00 18 00     NTLMSSP.........
00000010:62 00 00 00 18 00 18 00 7a 00 00 00 00 00 00 00     b.......z.......
00000020:48 00 00 00 08 00 08 00 48 00 00 00 12 00 12 00     H.......H.......
00000030:50 00 00 00 00 00 00 00 92 00 00 00 05 82 88 a2     P.......'....,ˆ¢
00000040:05 01 28 0a 00 00 00 0f 75 00 73 00 65 00 72 00     ..(.....u.s.e.r.
00000050:4e 00 46 00 2d 00 43 00 4c 00 49 00 45 00 4e 00     N.F.-.C.L.I.E.N.
00000060:54 00 0e 6a b2 7a 95 9e 59 37 00 00 00 00 00 00     T..j²z•ţY7......
00000070:00 00 00 00 00 00 00 00 00 00 83 f6 60 fc 8e 65     ..........ƒö`üŢe
00000080:ac e0 59 19 7b 9f 7f 9c d6 f5 70 5d 87 90 bd 02     ¬àY.{ŸœeÖõp]‡•½.
00000090:19 9c                                               .oe
```

The server sends an **IMAP4_AUTHENTICATE_NTLM_Fail_Response** message.

```
1 NO AUTHENTICATE failed.
```

4.2 IMAP4 Delegate Access Extension

In this scenario, Jason Carlson is using an IMAP4 client to access his email. His coworker, David Jones, has granted Jason delegate access to his mailbox. Jason uses his client to access David's mailbox.

```
0001 LOGIN contoso/jason/david P@ssw0rd
```

The server responds with a successful LOGIN response as described in [RFC3501].

```
0001 OK LOGIN completed.
```

4.3 IMAP UIDPLUS Extension

For examples using the IMAP UIDPLUS extension, see [RFC4315].

5 Security

5.1 Security Considerations for Implementers

Implementers have to be aware of the security considerations of using NTLM authentication, as described in [MS-NLMP].

5.2 Index of Security Parameters

Security parameter	Section
NTLM	Section 3.1.5.1 and section 3.2.5.1

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

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.

<1> Section 3.2.5.1: The initial release version of Exchange 2010 does not support the IMAP4 NTLM extension. Exchange 2010 SP1 supports the IMAP4 NTLM extension.

<2> Section 3.2.5.2: Exchange 2007 and Exchange 2007 SP1 do not support the IMAP4 delegate access extension. Exchange 2007 SP2 supports the IMAP4 delegate access extension.

<3> Section 3.2.5.3: Exchange 2003, Exchange 2007, and Exchange 2010 do not support the IMAP UIDPLUS extension. Exchange 2010 SP1 supports the IMAP UIDPLUS extension.

7 Change Tracking

This section identifies changes that were made to the [MS-OXIMAP4] protocol document between the July 2012 and October 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2.2 Informative References	Added the reference [MS-OXPROTO].	N	Content updated.
1.4 Relationship to Other Protocols	Added informative reference information for overview of relationships between this and other protocols.	N	Content updated.

8 Index

P

R

S

T

V