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

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	3.2.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/03/2010	3.3	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction 5
1.1 Glossary 5
1.2 References 6
1.2.1 Normative References 6
1.2.2 Informative References 7
1.3 Overview 7
1.4 Relationship to Other Protocols 9
1.5 Prerequisites/Preconditions 9
1.6 Applicability Statement 9
1.7 Versioning and Capability Negotiation 9
1.8 Vendor-Extensible Fields 9
1.9 Standards Assignments 10

2 Messages 11
2.1 Transport 11
2.2 Message Syntax 11
2.2.1 AUTHENTICATE Extensions 11
2.2.2 IMAP4 Server Messages 13
2.2.3 IMAP4 Client Messages 13
2.2.4 IMAP4 Delegate Access 13
2.2.5 UIDPLUS Extension 14

3 Protocol Details 15
3.1 IMAP4 Client Details 15
3.1.1 Abstract Data Model 15
3.1.1.1 IMAP4 State Model 15
3.1.1.2 NTLM Subsystem Interaction 16
3.1.2 Timers 17
3.1.3 Initialization 17
3.1.4 Higher-Layer Triggered Events 17
3.1.5 Message Processing Events and Sequencing Rules 17
3.1.5.1 Receiving an IMAP4_NTLM_Supported_Response Message 17
3.1.5.2 Receiving an IMAP4_AUTHENTICATE_NTLM_Fail_Unsupported Message 17
3.1.5.3 Receiving an IMAP4_NTLM_AUTHENTICATE_Blob_Response Message 17
3.1.5.3.1 Error from NTLM 18
3.1.5.3.2 NTLM Reports Success and Returns an NTLM Message 18
3.1.5.4 Receiving an IMAP4_AUTHENTICATE_Succeeded_Response Message 18
3.1.5.5 Receiving an IMAP4_AUTHENTICATE_NTLM_Fail_Response Message 18
3.1.6 Timer Events 18
3.1.7 Other Local Events 18
3.2 IMAP4 Server Details 18
3.2.1 Abstract Data Model 18
3.2.1.1 IMAP4 State Model 18
3.2.1.2 NTLM Subsystem Interaction 20
3.2.2 Timers 20
3.2.3 Initialization 21
3.2.4 Higher-Layer Triggered Events 21
3.2.5 Message Processing Events and Sequencing Rules 21
3.2.5.1 Receiving an IMAP4_AUTHENTICATE_NTLM_Initiation_Command Message 21
3.2.5.2 Receiving an IMAP4_AUTHENTICATE_NTLM_Blob_Command Message 21

3.2.5.2.1 NTLM Returns Success, Returning an NTLM Message 22
3.2.5.2.2 NTLM Returns Success, Indicating Authentication Completed Successfully .. 22
3.2.5.2.3 NTLM Returns Status, Indicating User Name or Password Was Incorrect 22
3.2.5.2.4 NTLM Returns a Failure Status, Indicating Any Other Error 22
3.2.5.2.5 NTLM Reports Success, Returning an NTLM Message 22
3.2.5.3 Receiving an IMAP4_AUTHENTICATE_NTLM_Cancellation_Command Message .. 22
3.2.6 Timer Events 22
3.2.7 Other Local Events 22

4 Protocol Examples 23
4.1 IMAP4 Client Successfully Authenticating to an IMAP4 Server 23
4.2 IMAP4 Client Unsuccessfully Authenticating to an IMAP4 Server 24

5 Security 26
5.1 Security Considerations for Implementers 26
5.2 Index of Security Parameters 26

6 Appendix A: Product Behavior 27

7 Change Tracking 29

8 Index 31

1 Introduction

This document specifies the implementation of extensions to the IMAP4 protocol. The following extensions are specified:

- **NTLM** (Windows NT LAN Manager) authentication mechanism for the IMAP4 protocol. This is a proprietary extension used with the **IMAP4 AUTHENTICATE** command, as specified in [RFC1730] and [RFC1731].
- **Delegate access** mechanism for the IMAP4 protocol. This is a proprietary extension that is used with the **IMAP4 LOGIN** command, as specified in [RFC1730].
- **IMAP4 UIDPLUS** command extension, as specified in [RFC4315].

For the purpose of this document, the NTLM authentication mechanism for the IMAP4 protocol is referred to in subsequent sections as "NTLM IMAP4 extension".

1.1 Glossary

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

alias
ASCII
Augmented Backus-Naur Form (ABNF)
base64 encoding
best body
body part
Connection-Oriented NTLM
delegate
delegator
domain
flags
folder
header
Hypertext Markup Language (HTML)
Hypertext Transfer Protocol (HTTP)
Internet Message Access Protocol – Version 4 (IMAP4)
Internet Message Access Protocol – Version 4 Revision 1 (IMAP4rev1)
mailbox
MIME
NTLM
NTLM message
NTLM AUTHENTICATE_MESSAGE
NTLM CHALLENGE_MESSAGE
NTLM NEGOTIATE_MESSAGE
NTLM software
plain text
property (1)
stream (2)

The following terms are specific to this document:

delegate access: The access that is granted by a **delegator** to a **delegate** and which is used by the **delegate** to access the **delegator's** account.

IMAP4 response: A message sent by an **IMAP4** server in response to a message from an **IMAP4** client. The structure of this message, as described in [RFC3501], is as follows:

```
<+OK> <response text><CR><LF>
```

Or:

```
<-ERR><response text><CR><LF>
```

user principal name (UPN): A user account name (sometimes referred to as the user logon name) and a domain name identifying the domain in which the user account is located. This is the standard usage for logging on to a Windows domain. The format is as follows: someone@example.com (in the form of an e-mail address). In the Active Directory directory service, the UPN is the **userPrincipalName** attribute of the **account** object, as described in [MS-ADTS].

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.

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", July 2006, http://msdn.microsoft.com/en-us/library/cc200343.aspx

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification", July 2006, http://msdn.microsoft.com/en-us/library/cc236621.aspx

[MS-OXBBODY] Microsoft Corporation, "Best Body Retrieval Protocol Specification", April 2008.

[RFC822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", RFC 822, August 1982, http://www.ietf.org/rfc/rfc0822.txt

[RFC1730] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4", RFC 1730, December 1994, ftp://ftp.rfc-editor.org/in-notes/rfc1730.txt

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

[RFC1734] Myers, J., "POP3 AUTHentication command", RFC 1734, December 1994, ftp://ftp.rfc-editor.org/in-notes/rfc1734.txt

[RFC2045] Freed, N., and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, http://www.ietf.org/rfc/rfc2045.txt

[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

[RFC2177] Leiba, B., "IMAP4 IDLE command", RFC 2177, June 1997, ftp://ftp.rfc-editor.org/in-notes/rfc2177.txt

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999, ftp://ftp.rfc-editor.org/in-notes/rfc2595.txt

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

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

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

1.2.2 Informative References

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

1.3 Overview

Client applications that connect to the **Internet Message Access Protocol - Version 4 (IMAP4)** service can use either standard **plain text** authentication, as specified in [RFC3501], or NTLM authentication.

The NTLM IMAP4 extension specifies how an IMAP4 client and IMAP4 server can use the NTLM Authentication protocol, as specified in [MS-NLMP], so that the IMAP4 server can authenticate the IMAP4 client. NTLM is a challenge/response authentication protocol that depends on the application layer protocols to transport NTLM packets from client to server, and from server to client.

This specification defines how the **IMAP4 AUTHENTICATE** command [RFC1731] is used to perform authentication by using the NTLM Authentication protocol. The IMAP4 Authentication command standard defines an extensibility mechanism for arbitrary authentication protocols to be plugged in to the core protocol.

This specification describes an embedded protocol in which NTLM authentication data is first transformed into a base64 representation, and then formatted by padding with IMAP4 keywords as defined by the AUTHENTICATE mechanism. The **base64 encoding** and the formatting are very rudimentary, and solely intended to make the NTLM data fit the framework specified in [RFC1731]. Figure 1 shows the sequence of transformations performed on an **NTLM message** to produce a message that can be sent over IMAP4.

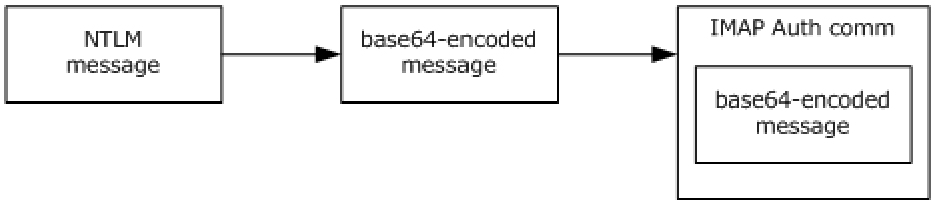

Figure 1: Relationship between NTLM message and IMAP4: NTLM Authentication protocol message

This specification describes a pass-through protocol that does not specify the structure of NTLM information. Instead, the protocol relies on the software that implements the NTLM Authentication protocol (as specified in [MS-NLMP]) to process each NTLM message to be sent or received.

This specification defines a server and a client role.

When IMAP4 performs an NTLM authentication, it has to interact with the NTLM subsystem appropriately. The following is an overview of this interaction.

If acting as an IMAP4 client:

1. The NTLM subsystem returns the first NTLM message to the client, to be sent to the server.
2. The client applies the base64 encoding and IMAP4-padding transformations to produce an IMAP4 message and send this message to the server.
3. The client waits for a response from the server. When the response is received, the client checks to determine whether the response indicates the end of authentication (success or failure), or that authentication is continuing.
4. If the authentication is continuing, the response message is stripped of the IMAP4 padding, base64 decoded, and passed into the NTLM subsystem, upon which the NTLM subsystem can return another NTLM message that has to be sent to the server. Steps 3 and 4 are repeated until authentication succeeds or fails.

If acting as an IMAP4 server:

1. The server then waits to receive the first IMAP4 authentication message from the client.
2. When an IMAP4 message is received from the client, the IMAP4 padding is removed, the message is base64 decoded, and the resulting NTLM message is passed into the NTLM subsystem.
3. The NTLM subsystem returns a status that indicates whether authentication completed.
4. If the authentication continues, the NTLM subsystem returns an NTLM message that has to be sent to the server. This message is base64 encoded, the IMAP4 padding is applied and sent to the client. Steps 2 through 4 are repeated until authentication succeeds or fails.

The sequence that follows shows the typical flow of packets between client and server after NTLM authentication has been selected.

1. The IMAP4 client sends an **NTLM NEGOTIATE_MESSAGE** embedded in an IMAP4_AUTHENTICATE_NTLM_Blob_Command packet to the server.
2. On receiving the IMAP4 packet with an NTLM NEGOTIATE_MESSAGE, the IMAP4 server sends an **NTLM CHALLENGE_MESSAGE** embedded in an IMAP4 packet to the client.
3. In response, the IMAP4 client sends an **NTLM AUTHENTICATE_MESSAGE** embedded in an IMAP4 packet.
4. The server then sends an **IMAP4 response** to the client to successfully complete the authentication process.

The NTLM NEGOTIATE_MESSAGE, NTLM CHALLENGE_MESSAGE, and NTLM AUTHENTICATE_MESSAGE packets contain NTLM authentication data that has to be processed by the **NTLM software** installed on the local computer. The way in which to retrieve and process NTLM messages is specified in [MS-NLMP].

Implementers of this specification have to conform to IMAP4 as specified [RFC1731] and [RFC3501], the **MIME**base64 encoding method as specified in [RFC2045], and the NTLM Authentication protocol as specified in [MS-NLMP].<1><2><3>

1.4 Relationship to Other Protocols

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

This specification defines how to perform delegate access with IMAP4.

IMAP4 specifies only the sequence in which an IMAP4 server and IMAP4 client 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 IMAP4 server processes NTLM messages. The IMAP4 client and IMAP4 server implementations depend on the availability of an implementation of the NTLM Authentication protocol (as specified in [MS-NLMP]) to obtain and process NTLM messages and on the availability of the base64 encoding and decoding mechanisms (as specified in [RFC2045]) to encode and decode the NTLM messages that are embedded in IMAP4 packets.

1.5 Prerequisites/Preconditions

Because IMAP4 depends on NTLM to authenticate the client to the server, both server and client require access to an implementation of the NTLM Authentication protocol (as specified in [MS-NLMP]) that is capable of supporting **Connection-Oriented NTLM**.

1.6 Applicability Statement

The NTLM IMAP4 extension is required to be used only when implementing an IMAP4 client that has to authenticate to an IMAP4 server by using NTLM authentication.

1.7 Versioning and Capability Negotiation

This specification covers versioning issues in the following areas:

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

The client discovers whether the server supports NTLM AUTHENTICATE through the **IMAP4 CAPABILITY** command, upon which the server responds with a list of supported features, among which authentication mechanisms are listed. If NTLM is supported, the server will list the word "AUTH=NTLM" in the list. The messages involved are specified in 2.2 of this document.<4><5>

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

The following sections specify how the NTLM IMAP4 extension messages are transported, along with the NTLM IMAP4 extension message syntax.<6>

2.1 Transport

The NTLM IMAP4 extension does not establish transport connections. Instead, NTLM IMAP4 extension messages are encapsulated in IMAP4 commands and responses. The way in which NTLM IMAP4 extension messages are encapsulated in IMAP4 commands is specified in section 2.2 of this document.

2.2 Message Syntax

The NTLM IMAP4 extension messages are divided into the following three categories, depending on whether the message was sent by the server or the client:

- **AUTHENTICATE** extensions
- IMAP4 server messages
- IMAP4 client messages

The **IMAP4 LOGIN** command extension enables optional delegate access. The **LOGIN** command extension adds an additional optional parameter that identifies the principal in a delegate access scenario. Their **LOGIN** command has four extended formats, as specified in [RFC1730].

2.2.1 AUTHENTICATE Extensions

The first category of IMAP4 messages is messages that fall within the **AUTHENTICATE** extensibility framework. These messages are specified in [RFC1731]. Some messages have parameters that have to be customized by the extensibility mechanism (such as NTLM). The following customizations are introduced in this specification. The tag rule in syntax below is specified in [RFC3501], section 9. All human readable strings are arbitrary and do not affect protocol functionality.

- A client can query the server to see if NTLM is supported. This is accomplished by issuing the **CAPABILITY** command without any parameters. This is shown in **ABNF** (for more information about ABNF, see [RFC4234]) in the following example:

  client_capability = tag "CAPABILITY" CRLF

- The server responds to this message with an untagged message that has a list of supported capabilities, followed by a tagged confirmation message. The prose-val rule is specified in [RFC4234], section 4. This sequence is shown in ABNF format in the following example:

  server_capability = "* CAPABILITY IMAP4 IMAP4rev1" AUTH=NTLM / AUTH=GSSAPI / AUTH=PLAIN "IDLE NAMESPACE LITERAL+" CRLF

  server_confirmation = tag OK prose-val CRLF

- [RFC1731] defines the syntax of the **AUTHENTICATE** command to initiate authentication. The parameter "mechanism" is defined to be the string "NTLM" for the NTLM IMAP4 extension. The command to initiate an NTLM conversation by a client in ABNF is shown in the following example. This is referred to as IMAP4_AUTHENTICATE_NTLM_Initiation_Command in this document.

  IMAP4_AUTHENTICATE_NTLM_Initiation_Command = tag AUTHENTICATE NTLM CRLF

- If NTLM is supported, the IMAP4 server will respond with an IMAP4 message to indicate that NTLM is supported. The syntax of this command in ABNF form is shown in the following example. This is referred to as IMAP4_NTLM_Supported_Response in this document.

  IMAP4_NTLM_Supported_Response = "+" CRLF

- If NTLM is not supported, the IMAP4 server returns a failure status code as defined by [RFC1730], section 7.1.3. The only data in this message that is useful is the BAD response. The remaining data is human-readable data and has no bearing on the authentication. The syntax of this command in ABNF form is shown in the following example. This is referred to as IMAP4_AUTHENTICATE_NTLM_Unsupported_Response in this document.

  IMAP4_AUTHENTICATE_NTLM_Unsupported_Response = tag BAD prose-val CRLF

- At every point of time during the authentication exchange, the client MUST parse the responses in the messages sent by the server and interpret them, as defined by [RFC1731]. The responses define various states such as success in authenticating, failure to authenticate, and any other arbitrary failures that the software might encounter.

  The client might receive any one of the following tagged responses during authentication..

  - IMAP4_AUTHENTICATE_NTLM_Blob_Response: The '+' status code indicates ongoing authentication, and indicates that the <base64-encoded-NTLM-Message> is to be processed by the authentication subsystem. In this case, the client MUST de-encapsulate the data, and pass it to the NTLM subsystem.

    IMAP4_AUTHENTICATE_NTLM_Blob_Response = "+" base64-encoded-NTLM-Message CRLF

    base64-encoded-NTLM-Message = prose-val

  - IMAP4_AUTHENTICATE_NTLM_Fail_Response: This message indicates that the authentication has terminated unsuccessfully, either because the username or password was incorrect, or due to some other arbitrary error, such as a software or data corruption error.

    IMAP4_AUTHENTICATE_NTLM_Fail_Response = tag NO prose-val CRLF

  - IMAP4_AUTHENTICATE_NTLM_Succeeded_Response: This message follows the format of the **OK Response** as specified in [RFC1731] section 7.1.1. It indicates that the authentication negotiation has completed.

    IMAP4_AUTHENTICATE_NTLM_Succeeded_Response = tag OK "AUTHENTICATE completed." CRLF

  - IMAP4_AUTHENTICATE_NTLM_Cancelled_Response: This message indicates that the authentication negotiation has been canceled with the client.

    IMAP4_AUTHENTICATE_NTLM_Cancelled_Response = tag NO " The AUTH protocol exchange was canceled by the client." CRLF

  - NTLM messages encapsulated by the client and sent to the server are referred to as IMAP4_AUTHENTICATE_NTLM_Blob_Command in this document. They have the following syntax defined in ABNF.

    IMAP4_AUTHENTICATE_NTLM_Blob_Command = base64-encoded-NTLM-Message CRLF

    base64-encoded-NTLM-Message = prose-val

- The client is able to cancel the authentication request by issuing an IMAP4_AUTHENTICATE_NTLM_Cancellation_Command. This has the following syntax defined in ABNF:

  IMAP4_AUTHENTICATE_NTLM_Cancellation_Command = "*" CRLF

2.2.2 IMAP4 Server Messages

This section defines the creation of IMAP4_AUTHENTICATE_NTLM_Blob_Response messages. These are NTLM messages that are sent by the server and MUST be encapsulated as follows to conform to syntax specified by the AUTHENTICATE mechanism.

1. Base64 encode the NTLM message data. This is needed because NTLM messages contain data outside the **ASCII** character range, whereas IMAP4 only supports ASCII characters.<7><8>
2. To the base64 encoded string, prefix the IMAP4 response code with a plus sign (+).
3. Suffix the <CR> and <LF> character (ASCII values 0x0D and 0x0A) as required by IMAP4.

The ABNF definition of a server message is as follows:

+ <base64-encoded-NTLM-message><CR><LF>

De-encapsulation of these messages by the client follows the reverse logic:

1. Remove the <CR> and <LF> character (ASCII values 0x0D and 0x0A).
2. Remove the IMAP4 response code (+).
3. Decode the base64 encoded IMAP4 data to produce the original NTLM message data.<9><10>

2.2.3 IMAP4 Client Messages

This section defines the processing of IMAP4_AUTHENTICATE_NTLM_Blob_Command messages. These NTLM messages that are sent by the client are encapsulated as follows to conform to the AUTHENTICATE mechanism:

1. Base64 encode the NTLM message data. This is needed because NTLM messages contain data outside the ASCII character range, whereas IMAP4 only supports ASCII characters.
2. Send the base64-encoded string.
3. Suffix the <CR> and <LF> character (ASCII values 0x0D and 0x0A), as required by IMAP4.

The ABNF definition of a client message is as follows:

<base64-encoded-NTLM-Message><CR><LF>

De-encapsulation of these messages by the client follows the reverse logic:

1. Remove the <CR> and <LF> character (ASCII values 0x0D and 0x0A).
2. Base64-decode the IMAP4 data to produce the original NTLM message data.

2.2.4 IMAP4 Delegate Access

The IMAP4 Delegate Access extension extends the **LOGIN** command, as specified in [RFC3501] section 6.2.3. Specifically, the IMAP4 Delegate Access extension extends the **user name** argument of the **LOGIN** command so that a **delegate** and **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 using delegate access with IMAP4. In every case, the part after the last "/" of the user string is the **mailbox** identity in either **alias** or user principal name (UPN) format. The four formats are as follows:

- LOGIN domain/delegateuseralias/principalalias
- LOGIN domain/delegateuseralias/principalupn
- LOGIN delegateuserupn/principalalias
- LOGIN delegateuserupn/principalupn

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

The delegateuserupn part of the login string represents the 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 represent the email alias of the delegate.

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

The prinicpaluseralias part of the login string represents the e-mail alias of the delegator.

2.2.5 UIDPLUS Extension

The server fully supports the **IMAP4 UIDPLUS** command extension, as specified in [RFC4315], and MUST implement the response codes specified in [RFC4315] section 3. <11>

3 Protocol Details

3.1 IMAP4 Client Details

3.1.1 Abstract Data Model

3.1.1.1 IMAP4 State Model

Figure 2 shows the client IMAP4 state model.

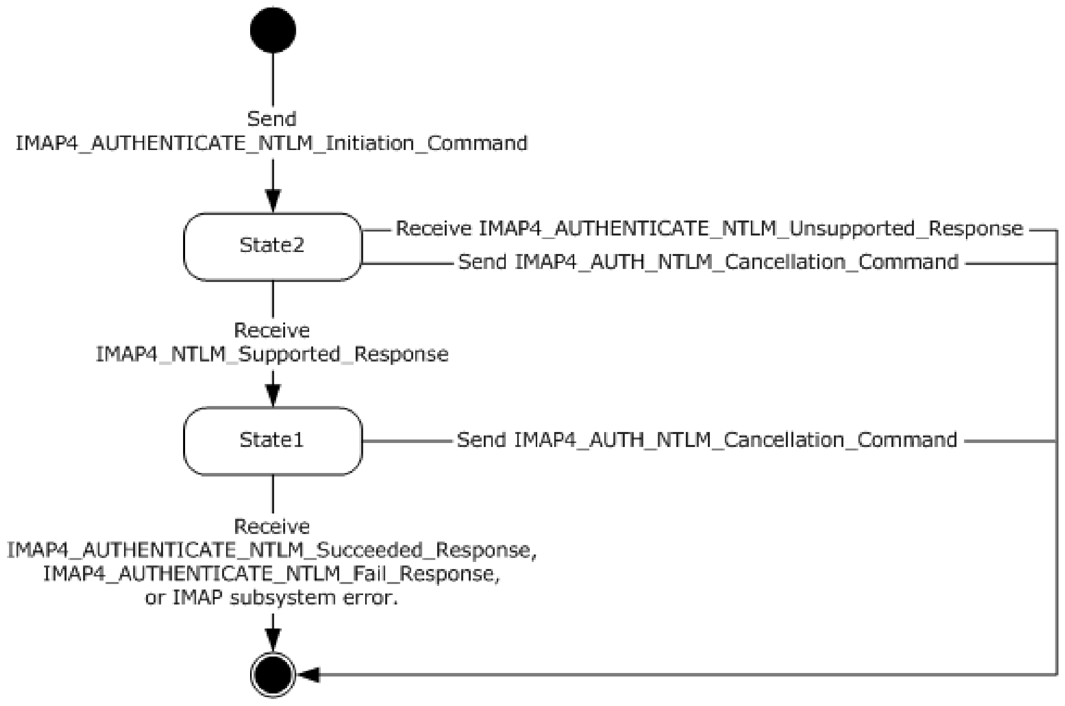
Send
IMAP4_AUTHENTICATE_NTLM_Initiation_Command
State2
Receive IMAP4_AUTHENTICATE_NTLM_Unsupported_Response
Send IMAP4_AUTH_NTLM_Cancellation_Command
Receive
IMAP4_NTLM_Supported_Response
State1
Send IMAP4_AUTH_NTLM_Cancellation_Command
Receive
IMAP4_AUTHENTICATE_NTLM_Succeeded_Response,
IMAP4_AUTHENTICATE_NTLM_Fail_Response,
or IMAP subsystem error.

Figure 2: Client IMAP4 state model

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

1. Start:

   This is the state of the client before the IMAP4_AUTHENTICATE_Initiation_Command has been sent.

2. State 2: sent_authentication_request.

   This is the state of the client after the IMAP4_AUTHENTICATE_Initiation_Command has been sent.

3. State 1: inside_authentication

   This is the state that is entered by a client after it has received an IMAP4_NTLM_Supported_Command. In this state, the client initializes the NTLM subsystem and repeats the following steps:

- Encapsulates the NTLM message, returned by the NTLM subsystem, into an IMAP4 message. Waits for a response from the server.
- De-encapsulates received IMAP4 message data (if any) from the other party and converts it to NTLM message data.
- Passes it to the NTLM subsystem.
- Sends the IMAP4 message to the other party.

This state terminates when:

- For the server: The NTLM subsystem reports completion with either a success or failed authentication status, upon which it sends the client an IMAP4_AUTHENTICATE_NTLM_Suceeded_Response or IMAP4_AUTHENTICATE_NTLM_Fail_Response, as specified in [RFC1731].
- For the client: an IMAP4_AUTHENTICATE_NTLM_Suceeded_Response or IMAP4_AUTHENTICATE_NTLM_Fail_Response is received.
- For either client or server: when any failure is reported by the NTLM subsystem.

4. Stop: completed_authentication.

This is the state of the client on exiting the inside_authentication or the sent_authentication_request state. The rules for how the inside_authentication state is exited are defined in section 3.1.5. The behavior of IMAP4 in this state is outside the scope of this specification—it represents the end state of the authentication protocol.

3.1.1.2 NTLM Subsystem Interaction

During the **inside_authentication** phase, the IMAP4 client invokes the NTLM subsystem as specified in [MS-NLMP]. The NTLM protocol is used with these options:

1. The negotiation is a Connection-Oriented NTLM negotiation.
2. None of the **flags** specified in [MS-NLMP] are specific to NTLM.

The following is a description of how IMAP4 uses NTLM. All NTLM messages are encapsulated as specified in section 2.1. [MS-NLMP] describes the data model, internal states, and sequencing of NTLM messages in greater detail.

1. The client initiates the authentication by invoking NTLM, after which NTLM will return the NTLM NEGOTIATE_MESSAGE to be sent to the server.
2. Subsequently, the exchange of NTLM messages goes on as defined by the NTLM protocol, with the IMAP4 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. The NTLM protocol 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** 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 package (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 system. If the NTLM system 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

None.

3.1.4 Higher-Layer Triggered Events

When the client receives a **IMAP4_NTLM_AUTHENTICATE_Blob_Response** message that contains an NTLM CHALLENGE_MESSAGE, it is passed to the NTLM system. If the NTLM system is successful in handling the message, the NTLM system returns a NTLM AUTHENTICATE_MESSAGE. The successful creation of the NTLM AUTHENTICATE_MESSAGE that is passed to the client triggers the client to send an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains the NTLM AUTHENTICATE_MESSAGE. If the NTLM system encounters an error when the NTLM CHALLENGE_MESSAGE from the **IMAP4_NTLM_AUTHENTICATE_Blob_Response** message is handled, then the failure returned by the NTLM system MUST trigger the client to transition to the **completed** state.

3.1.5 Message Processing Events and Sequencing Rules

The NTLM IMAP4 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 [RFC1731] and [MS-NLMP]. Section 3.1.1 defines how the rules specified in [RFC1731] and [MS-NLMP] 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 the completed_authentication state.

3.1.5.1 Receiving an IMAP4_NTLM_Supported_Response Message

The expected state is sent_authentication_request.

On receiving this message, a client MUST generate the first NTLM message by calling the NTLM subsystem. The NTLM subsystem then generates NTLM NEGOTIATE_MESSAGE, as specified in [MS-NLMP]. The NTLM message is then encapsulated as defined in this specification and sent to the server.

The state of the client is changed to "inside_authentication".

3.1.5.2 Receiving an IMAP4_AUTHENTICATE_NTLM_Fail_Unsupported Message

The expected state is sent_authentication_request.

On receiving this message, a client MUST abort the NTLM authentication attempt.

3.1.5.3 Receiving an IMAP4_NTLM_AUTHENTICATE_Blob_Response Message

The expected state is inside_authentication.

On receiving this message, a client MUST de-encapsulate it to obtain the embedded NTLM message, and pass it to the NTLM subsystem for processing. The NTLM subsystem can then either report an error, or report success and return an NTLM message to be sent to the server.

3.1.5.3.1 Error from NTLM

If the NTLM subsystem reports an error, the client MUST change its internal state to "completed_authentication" and consider that the authentication has failed. The client can then take any action it considers appropriate; this document does not mandate any specific course of action.

Typical actions are to try other (non-authentication-related) IMAP4 commands, or to disconnect the connection.

3.1.5.3.2 NTLM Reports Success and Returns an NTLM Message

The NTLM message MUST be encapsulated and sent to the server. No change occurs in the state of the client.

3.1.5.4 Receiving an IMAP4_AUTHENTICATE_Succeeded_Response Message

Expected state: inside_authentication.

The IMAP4 client MUST change its internal state to completed_authentication and consider that the authentication has succeeded. The client can then take any action it considers appropriate. This document does not mandate any specific course of action.

3.1.5.5 Receiving an IMAP4_AUTHENTICATE_NTLM_Fail_Response Message

Expected state: inside_authentication.

The IMAP4 client MUST change its internal state to completed_authentication and consider that the authentication has failed. The client can then take any action it considers appropriate. This document does not mandate any specific course of action.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 IMAP4 Server Details

3.2.1 Abstract Data Model

3.2.1.1 IMAP4 State Model

Figure 3 shows the server IMAP4 state model.

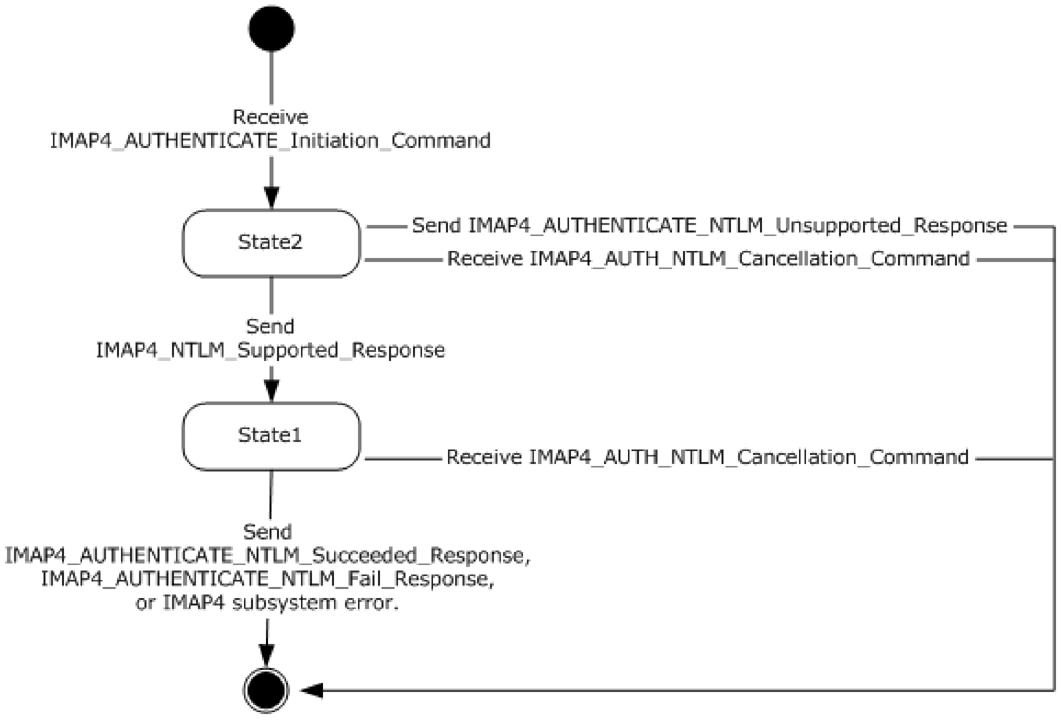
Receive IMAP4_AUTHENTICATE_Initiation_Command
State2
Send IMAP4_AUTHENTICATE_NTLM_Unsupported_Response
Receive IMAP4_AUTH_NTLM_Cancellation_Command
Send IMAP4_NTLM_Supported_Response
State1
Receive IMAP4_AUTH_NTLM_Cancellation_Command
Send IMAP4_AUTHENTICATE_NTLM_Succeeded_Response, IMAP4_AUTHENTICATE_NTLM_Fail_Response, or IMAP4 subsystem error.

Figure 3: Server IMAP4 state model

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

1. Start.

   This is the state of the server before the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** has been received.

2. State 2: **received_authentication_request**.

   This is the state of the client after the **IMAP4_AUTHENTICATE_NTLM_Initiation_Command** has been received.

3. State 1: **inside_authentication**.

   This is the state entered by a server after it has sent an **IMAP4_NTLM_Supported_Response**. In this state, the server initializes the NTLM subsystem and repeats the following steps:

   - Waits for a message from the client.
   - De-encapsulates received IMAP4 message-data party and obtains the embedded NTLM message data.
   - Passes it to the NTLM subsystem.
   - Encapsulates the NTLM message returned by the NTLM subsystem into an IMAP4 message.
   - Sends the IMAP4 message to the other party.

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** or **IMAP4_AUTHENTICATE_NTLM_Fail_Response**, as specified in [RFC1731].
- Any failure is reported by the NTLM subsystem.

4. Stop: **completed_authentication**.

   This is the state of the server on exiting the **inside_authentication** or the **sent_authentication_request** state. The rules for how the **inside_authentication** state is exited are defined in section 3.2.5. The behavior of IMAP4 in this state is defined in [RFC1731]—it represents the **end_state** of the authentication protocol.

3.2.1.2 NTLM Subsystem Interaction

During the inside_authentication state, the IMAP4 server invokes the NTLM subsystem as specified in [MS-NLMP]. The NTLM protocol is used with the following options:

1. The negotiation is a Connection-Oriented NTLM negotiation.
2. None of the flags specified in [MS-NLMP] are specific to NTLM.

The following is a description of how IMAP4 uses NTLM. For more details, see [MS-NLMP], which describes the data-model and sequencing of NTLM packets in greater detail.

1. The server, on receiving the NTLM NEGOTIATE_MESSAGE, passes it to the NTLM subsystem and is returned the NTLM CHALLENGE_MESSAGE, if the NTLM NEGOTIATE_MESSAGE was valid.
2. Subsequently, the exchange of NTLM messages goes on as defined by the NTLM protocol, with the IMAP4 server encapsulating the NTLM messages that are returned by NTLM before sending them to the client.
3. When the NTLM protocol completes authentication, either successfully or unsuccessfully, the NTLM subsystem notifies IMAP4.
   - On successful completion, the server MUST exit the inside_authentication state and enter the completed_authentication state and send the IMAP4_AUTHENTICATE_Success_Response to the client. On receiving this message, the client MUST also transition to the completed_authentication state.
   - If a failure occurs due to an incorrect password error, as described in [MS-NLMP], the server SHOULD enter the completed_authentication state and send the client an IMAP4_AUTHENTICATE_Failure_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_Failure_Response message. On receiving this message, the client MUST enter the completed_authentication state.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

When the server receives a **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an NTLM NEGOTIATE_MESSAGE, it is passed to the NTLM system. If the NTLM system is successful in handling the message, the NTLM system returns a NTLM CHALLENGE_MESSAGE. The NTLM system returning the NTLM CHALLENGE_MESSAGE triggers the server to send an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message that contains the NTLM CHALLENGE_MESSAGE.When the server receives an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an NTLM AUTHENTICATE_MESSAGE, it is passed to the NTLM system. If the NTLM system is successful in handling the message, the NTLM system returns information that the client successfully logged on. The NTLM system successful logon triggers the server to send an **IMAP4_AUTHENTICATION_SUCCEEDED_RESPONSE** message. The server state is changed to the completed state.When the server receives an **IMAP4_AUTHENTICATE_NTLM_Blob_Command** message that contains an NTLM AUTHENTICATE_MESSAGE, it is passed to the NTLM system. If the NTLM system handles the NTLM AUTHENTICATE_MESSAGE and the message has an incorrect user name or password, the NTLM system MUST terminate authentication. The NTLM system informs the server that authentication has been stopped which triggers the server to send an **IMAP4_AUTHENTICATE_NTLM_Failed_Response** message to the client. The server state is changed to the completed state.If the NTLM system returns any failure status, the failure status MUST trigger the server to send an **IMAP4_AUTHENTICATE_NTLM_Failed_Response** message to the client. The server state is changed to the completed state.

3.2.5 Message Processing Events and Sequencing Rules

The NTLM IMAP4 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 [RFC1731] and [MS-NLMP]. Section 3.1.1 defines how the rules specified in [RFC1731] and [MS-NLMP] 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 the completed state.

3.2.5.1 Receiving an IMAP4_AUTHENTICATE_NTLM_Initiation_Command Message

The expected state is **start**.

On receiving this message, the server MUST reply with the **IMAP4_NTLM_Supported_Response**, if it supports NTLM, and change its state to the **inside_authentication** state.

If the server does not support NTLM, it MUST respond with the **IMAP4_NTLM_AUTHENTICATE_Unsupported_Response**, and the internal state transitions to **completed_authentication**.

3.2.5.2 Receiving an IMAP4_AUTHENTICATE_NTLM_Blob_Command Message

The expected state is inside_authentication.

On receiving this message, a server MUST de-encapsulate the message, obtain the embedded NTLM message, and pass it to the NTLM subsystem. The NTLM subsystem MUST perform one of the following:

1. Report success in processing the message and return an NTLM message to continue authentication.
2. Report that authentication completed
3. Report that authentication failed due to a bad user name or password, as specified in [MS-NLMP].
4. Report that the authentication failed due to some other software error or message corruption.

3.2.5.2.1 NTLM Returns Success, Returning an NTLM Message

The NTLM message MUST be encapsulated and sent to the client in an **IMAP4_AUTHENTICATE_NTLM_Blob_Response** message.

3.2.5.2.2 NTLM Returns Success, Indicating Authentication Completed Successfully

The server MUST return the IMAP4_NTLM_AUTHENTICATE_Succeeded_Response and change its internal state to completed_authentication.

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

The server MUST return the IMAP4_AUTHENTICATE_NTML_Failed_Response and change its internal state to completed_authentication.

3.2.5.2.4 NTLM Returns a Failure Status, Indicating Any Other Error

The server MUST return the IMAP4_AUTHENTICATE_NTLM_Failed_Response and change its internal state to completed_authentication.

3.2.5.2.5 NTLM Reports Success, Returning an NTLM Message

The NTLM message SHOULD be encapsulated and sent to the server. No change occurs in the state of the client.

3.2.5.3 Receiving an IMAP4_AUTHENTICATE_NTLM_Cancellation_Command Message

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

On receiving this message, the server MUST change its state to the **completed** state and send an **IMAP4_AUTHENTICATE_NTLM_Cancelled_Response** message to the client.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

The following sections describe operations used in a common scenario to illustrate the function of the **Internet Message Access Protocol – Version 4 Revision 1 (IMAP4rev1)**.

4.1 IMAP4 Client Successfully Authenticating to an IMAP4 Server

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

1. The client sends an IMAP4_AUTHENTICATE_NTLM_Initiation_Command to the server. This command is specified in [RFC1731] and does not carry any IMAP4-specific data. It is included in this example to provide a better understanding of the **IMAP4 NTLM** initiation command.

```
1 AUTHENTICATE NTLM
```

2. The server sends the IMAP4_NTLM_Supported_Response message, indicating that it can perform NTLM authentication.

```
+
```

3. The client sends an IMAP4_AUTHENTICATE_NTLM_Blob_Command message that contains a base64-encoded NTLM NEGOTIATE_MESSAGE.

```
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 ..(.....
```

4. The server sends an IMAP4_AUTHENTICATE_NTLM_Blob_Response message that contains a base64-encoded NTLM CHALLENGE_MESSAGE.

```
+ 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....
```

5. The client sends an IMAP4_AUTHENTICATE_NTLM_Blob_Command message that contains a base64-encoded NTLM AUTHENTICATE_MESSAGE.

```
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.......'....,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 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     .Ëß.F.¡Þl.0—Û(Ûq
00000090:9a 6a šj
```

6. The server sends an IMAP4_AUTHENTICATION_SUCCEEDED_RESPONSE message.

```
1 OK AUTHENTICATE completed.
```

4.2 IMAP4 Client Unsuccessfully Authenticating to an IMAP4 Server

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

The client sends an IMAP4_AUTHENTICATE_NTLM_Initiation_Command to the server. This command is defined in [RFC1731] and does not carry any IMAP4-specific data. It is included in this example to provide a better understanding.

The server sends the IMAP4_NTLM_Supported_Response message, indicating that it can perform NTLM authentication.

1. The client sends an IMAP4_AUTHENTICATE_NTLM_Blob_Command message that contains a base64-encoded NTLM NEGOTIATE_MESSAGE.

```
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 ..(.....
```

2. The server sends an IMAP4_AUTHENTICATE_NTLM_Blob_Response message that contains a base64-encoded NTLM CHALLENGE_MESSAGE.

```
+ 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.....
```

3. The client sends an IMAP4_AUTHENTICATE_NTLM_Blob_Command message that contains a base64-encoded NTLM AUTHENTICATE_MESSAGE.

```
TlRMTVNTUAADAAAAGAAYAGIAAAAYABgAegAAAAAAAABIAAAACAAIAEgAAAASABIA
UAAAAAAAAACSAAAABYKIogUBKAoAAAAPdQBzAGUAcgBOAEYALQBDAEwASQBFAE4A
VAAOarJ6lZ5ZNwAAAAAAAAAAAAAAAAAAAACD9mD8jmWs4FkZe59/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.{ŸoeÖõp]‡•½.
00000090:19 9c .oe
```

4. The server sends an IMAP4_AUTHENTICATE_Failed_Response message.

```
1 NO AUTHENTICATE failed.
```

5 Security

The following sections specify security considerations for implementers of the NTLM IMAP4 extension.

5.1 Security Considerations for Implementers

Implementers have to be aware of the security considerations of using NTLM authentication. Information about the security considerations for using NTLM authentication is specified in [MS-NLMP].

5.2 Index of Security Parameters

Security Parameter	Section
NTLM	2 and 3

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products:

- Microsoft® Office Outlook® 2003
- Microsoft® Exchange Server 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Exchange Server 2007
- Microsoft® Outlook® 2010
- Microsoft® Exchange Server 2010
- Microsoft® Exchange Server 2010 Service Pack 1 (SP1)

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. 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 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 product does not follow the prescription.

<1> Section 1.3: [RFC3501] does not impose any limitations on **folder** hierarchy depth on Exchange 2007 or Exchange 2010. However, the folder hierarchy depth in Exchange 2007 and Exchange 2010 is limited to 31 levels.

<2> Section 1.3: In Exchange 2007 and Exchange 2010, the folder name is limited to 250 characters. Therefore, folder names cannot be created that are longer than 250 characters. However, if a folder name is created via a different protocol that exceeds this limit, the folder name is truncated when the folder is retrieved.

<3> Section 1.3: In Exchange 2007 and Exchange 2010, the maximum IMAP4 command size, including arguments, is limited to 10240 characters.

<4> Section 1.7: Exchange 2007, Exchange 2010, and Office Outlook 2003 clients mutually supported [RFC3501], [RFC1731], and [RFC2177]. Exchange 2007, Exchange 2010, and Office Outlook 2007 clients mutually supported [RFC3501], [RFC1731], [RFC2177], and [RFC2595].

<5> Section 1.7: Exchange 2010 is not [RFC822]-compliant by default. Exchange 2010 can be made [RFC822]-compliant by setting EnableExactRFC822Size to TRUE.

<6> Section 2: Exchange 2010 does not support NTLM authentication.

<7> Section 2.2.2: In Exchange 2003, messages received by using MAPI are converted to MIME the first time they are retrieved and subsequently stored. The MIME size might be different before the message is retrieved than after it is converted to MIME.

<8> Section 2.2.2: In Exchange 2003, Exchange 2007, and Exchange 2010, special characters are allowed in folder names. Special characters can be used by clients to create folder names. Therefore, in order to be able to list and access those folders via IMAP4, special characters have to

be supported. All characters listed in [RFC1734] are supported in folder names, except the folder delimiter character, "/" (forward slash). The folder names with special characters might be required to be enclosed in quotes (" ") or sent as literals.

<9> Section 2.2.2: In Exchange 2007 and Exchange 2010, messages are not stored in MIME format. Messages are converted from MAPI to MIME before being sent to the client. When the client requests the size of the message before retrieving the actual message itself, the MIME size provided is the size associated with the Message as a MAPI **property**. When the client retrieves the message, the message is converted from MAPI to MIME and the message size calculated thereafter can be different than the size calculated from the MAPI property. Additionally, if the message is modified by MAPI, the size is likely to change again and thus the corresponding MIME size after actual conversion changes from MAPI to MIME.

<10> Section 2.2.2: In Exchange 2003, the MIME **stream** is preserved. In Exchange 2007 and Exchange 2010, the MIME stream is not preserved. Only content of the **best body**, as specified in [MS-OXBBODY], is preserved. The order in which the best **body part** is selected is as follows:

- Enriched Text Format
- HTML
- plain text

Therefore, the MIME stream is regenerated every time a message is retrieved. The alternative body parts are regenerated on demand as the message is retrieved. In Exchange 2010, the MIME **headers** and body parts of messages are stored if the headers existed when the message was delivered.

<11> Section 2.2.5: Exchange 2003, Exchange 2007, and Exchange 2010 do not support the **IMAP4 UIDPLUS** command extension.-

7 Change Tracking

This section identifies changes that were made to the [MS-OXIMAP4] protocol document between the August 2010 and November 2010 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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 revision 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
1.2.1 Normative References	Added reference information for [MS-ADTS].	N	Content updated.

8 Index

A

Applicability 9

C

Capability negotiation 9
Change tracking 29

E

Examples
  overview 23

F

Fields – vendor-extensible 9

G

Glossary 5

I

IMAP4 Client
  overview 15
IMAP4 Server
  overview 18
Implementer – security considerations 26
Index of security parameters 26
Informative references 7
Introduction 5

M

Messages
  overview 11
Messaging
  transport 11

N

Normative references 6

O

Overview (synopsis) 7

P

Parameters – security index 26
Preconditions 9
Prerequisites 9
Product behavior 27

R

References
  informative 7
  normative 6
Relationship to other protocols 9

S

Security
  implementer considerations 26
  overview 26
  parameter index 26
Standards Assignments 10

T

Tracking changes 29
Transport 11

V

Vendor-extensible fields 9
Versioning 9