WHITEPAPER SERIES

Internet Mail Access Protocol Version 4 (IMAP4) and Internet Electronic Mail

Version 1.0 **November 1998**

Hong Kong Computer Center, 20/F
54-62 Lockhart Road
Wan Chai
HONG KONG
Tel: +852 2520-0300
Fax: +852 2648-5913

The Peak Tower, 15/F
107 Alfaro Street
Salcedo Village, Makati City
PHILIPPINES
+63 (2) 811-3999
+63 (2) 811-3939

USA Support/Sales: +1 (408) 481-9985
USA Fax: +1 (888) 562-3561

Email: info@ima.com
Website: http://www.ima.com

Please send all comments and suggestions to the Whitepaper Series to *doc@ima.com*

TABLE OF CONTENTS

INTRODUCTION

RFC 2060, which obsoletes RFC 1730, defines the Internet Message Access Protocol Version 4 (IMAP4) as a protocol that allows manipulation of remote message folders, known as mailboxes, in a manner that is functionally equivalent to local mail folders. Unlike the POP3 protocol, IMAP4 has a number of features which make it more user-friendly. Whereas POP3 is designed primarily to function as an offline protocol, IMAP4 is used primarily as an on-line protocol, but can also support an offline mode as well.

In the online mode of IMAP4, the user manipulates messages/message folders on the server without having to download them to a local hard disk. It also supports efficient folder management by allowing users to create multi-level folders or mailboxes on the server (with proper authorization from the administrator) that can easily be renamed, moved, or deleted by them. Folder and message updates made through an IMAP4 client will be seen by other clients accessing the same message store.

These features are particularly useful to multiple-computer users. In IMAP4, a user with different computers – say a PC at home, Mac in the office, and a laptop on the road – can move freely among them and access the same message store or post office on the server. Moreover, IMAP4 allows multiple users to access a shared mailbox on the server from multiple platforms concurrently and view the actual status of each message.

Unlike POP3, IMAP4 has the ability to search for messages on the server using various message attributes such as message size, headers, and message sender, among others. It can also separate attached files from the text and header portions of the message. This is particularly useful for handling multipart MIME messages. In POP3, a message must be downloaded in its entirety, including unwanted embedded attachments, before they can be read. IMAP4 gives users the capability to retrieve the header or certain portions of a multipart MIME message first. Using this feature, users can scan the header information on a MIME message and decide which part(s) of the message they need.

Another advantage of IMAP4 over POP3 is that the former has a built-in security feature that prevents a user's password from being transmitted in the clear over the network. It uses an authentication mechanism that is based on a series of server challenges and client answers. POP3 has a similar feature invoked by the APOP command. However, very few email clients support this feature.

THE IMAP4 SESSION

The IMAP4 server listens on TCP port 143. The client/server network connection is followed by client/server interactions, which consist of a client command, server data, and a server completion result response. Interactions between the client and the server are in the form of lines, i.e. strings that end with a CRLF. The protocol receiver of an IMAP4 client or server either reads a line or a sequence of octets with a known count followed by such strings.

The client command begins an operation via the client protocol sender. Each client command is prefixed with an identifier called tag (e.g. A0001, A0002, etc.). The client generates a different tag for each command. There are two cases where a line from the client does not represent a complete command: when a command argument is quoted

with an octet count, and when the command arguments require server feedback. If either case is encountered, a command continuation request response is issued by the server to inform the client if it is ready for the octets and the remainder of the command. This response is prefixed with the token "+". The server, meanwhile, reads a command line from the client, parses the command and its arguments, and transmits server data and a server command completion result.

Data is sent by the server to the client and status responses that do not indicate command completion are prefixed with the token "*", which are called untagged responses. Server data may be sent as a response to a client command or it may be sent unilaterally by the server. The server completion result response indicates whether the operation is successful or not. There are three possible server completion responses: OK (indicating success), NO (indicating failure), or BAD (indicating protocol error).

The IMAP4 client reads a response line sent by the server, including data that was not requested. This data must be recorded so that the IMAP4 client can reference its recorded copy instead of sending a command to the server to request for the data.

State Diagram

An IMAP4 server is always in one of four states: the non-authenticated state, the authenticated state, the selected state, or the logout state. If a client attempts a command in the wrong state, a protocol error occurs. Following is a brief description of each state:

1. *Non-Authenticated State*
   The client must supply authentication credentials in this state before most commands can be permitted. The IMAP4 session enters this state when connection starts.

   Take note, however, that if the connection has been pre-authenticated, the session goes directly to the Authenticated State.

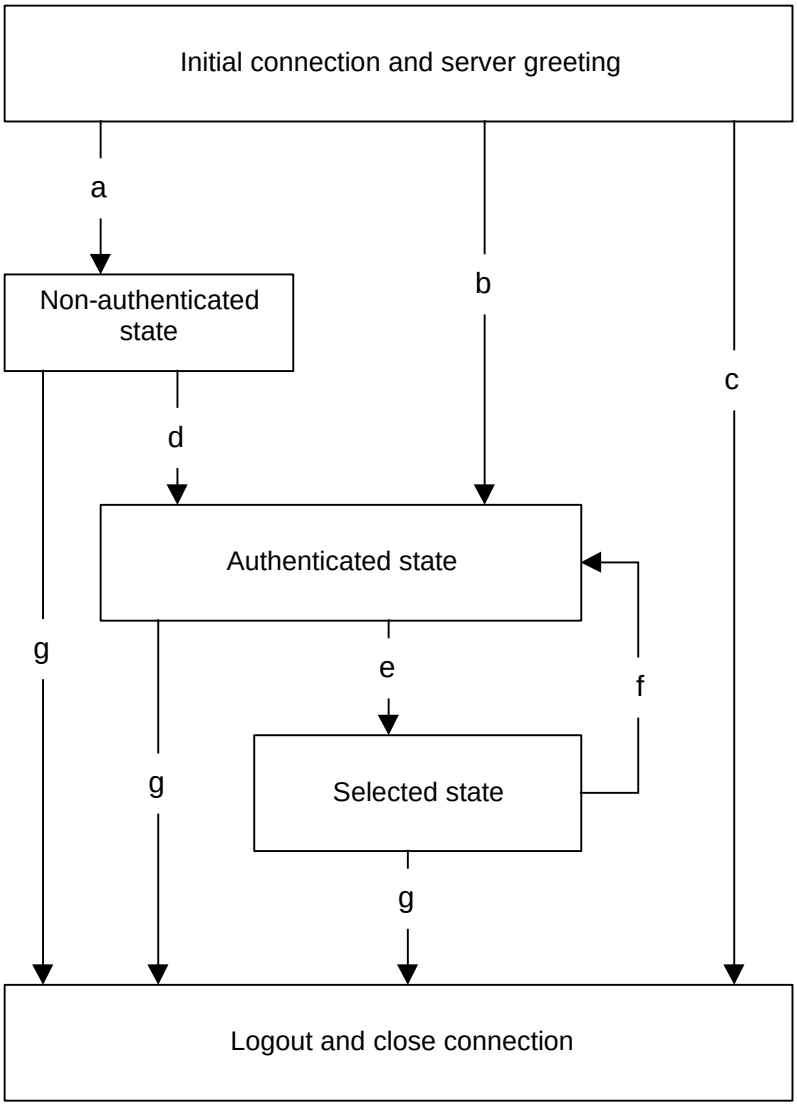
2. *Authenticated State*
   After the client has been authenticated, the session enters the Authenticated State. In this state, the client must first select a mailbox to access before commands that affect the messages can be permitted.

   The session enters the authenticated state when a pre-authenticated connection starts, when acceptable authentication credentials have been validated, or after an error in mailbox selection occurs.

3. *Selected State*
   The IMAP4 session enters this state after a mailbox has been sucessfully selected.

4. *Logout State*
   The connection is terminated by the server in this state either as a result of client request or by unilateral server decision.

The following figure shows the state diagram for a complete IMAP4 session.



a) connection without pre-authentication (OK greeting)
b) connection with pre-authentication (PREAUTH greeting)
c) connection is rejected (BYE greeting)
d) successful LOGIN or AUTHENTICATE command
e) successful SELECT or EXAMINE COMMAND
f) CLOSE command, or failed SELECT or EXAMINE command
g) LOGOUT command, connection terminates.

Message Attributes

i. Message Numbers

IMAP4 messages are accessed by one or two numbers, namely the unique identifier and the message sequence number. The unique identifier (UID) is a 32-bit value, which when used with the unique identifier validity value (a unique identifier validity

value is assigned to each mailbox – this value is sent in an UIDVALIDITY response code in an OK untagged response during mailbox selection) forms a 64-bit value that is permanently guaranteed not to refer to any other message in the mailbox.

A unique identifier of a message must remain unchanged during the session and between sessions. If it is not possible to preserve a message's unique identifier, each subsequent session must then have a new unique identifier validity value that is larger than any that was used previously.

Take note that unique identifiers are assigned in a strictly ascending fashion in the mailbox, meaning each message added to the mailbox is assigned a higher UID than the previous messages.

The unique identifier for a particular message is used in determining the message sequence number for that message. The message sequence number is a relative position from 1 to the number of messages in the mailbox.

ii. Flags

Flags consist of a list of zero or more named tokens associated with the message. There are two types of flags used in IMAP4 – the system flag and the keyword.

A system flag is a flag name that begins with "\". The currently-defined system flags are:

\Seen – message has been read
\Answered – message has been answered
\Flagged – message is "flagged" for urgent/special attention
\Deleted – message is "deleted" for later removal by EXPUNGE
\Draft – message has not completed composition (marked as a draft)

A keyword, on the other hand, is defined by the server implementation. Servers may allow the client to define new keywords in the mailbox. A keyword does not begin with "\".

A flag of either type may be classified as permanent or session-only. Permanent flags can be added or removed by the client from the message flags. Changes to permanent flags will be seen by subsequent sessions. Changes to session flags are valid only for that particular session.

iii. Internal Date

This is the date and time when the message was received by the server (to differentiate it from the date and time in the RFC 822 header).

iv. Message Size

The number of octets in the message as defined in the RFC 822 format.

v. Envelope Structure

A parsed representation of the RFC 822 envelope information. This is not the same as the SMTP envelope of the message.

vi. Body Structure
This is a parsed representation of the body structure information of the MIME message.

Message Texts

IMAP is not only capable of fetching the full RFC 822 text of a message but can also be used to fetch portions of the full message text. Specifically, IMAP4 can fetch RFC 822 message header, the RFC 822 message body, or the MIME message header.

IMAP4 Data Formats

Textual commands and responses are used in IMAP4 sessions. Data in IMAP4 can either be in one of the following formats:

i. *Atom* – this format consists of one or more non-special characters.

ii. *Number* – a number consists of one or more digit characters and represents a numeric value.

iii. *String* – the string comes in two forms: the literal and the quoted string. The literal string is the more common form of string, and consists of a sequence of zero or more octets (including CR and LF). It is prefix-quoted with an octet count in the form of an open brace ("{").

A quoted string, on the other hand, consists of a sequence of zero or more 7-bit characters, excluding CR and LF, with double quote (<">) characters at each end.

The empty string is represented as either " " (a quoted string with zero characters between double quotes) or as {0} followed by CRLF (a literal with an octet count of 0).

iv. *Parenthesized list* – IMAP4 represents data structures as a "parenthesized list", which consists of a sequence of data items, delimited by space, and bounded at each end by parentheses. An empty list is represented by ().

v. *NIL* – this represents the non-existence of a particular data item that is represented either as a string or a parenthesized list. NIL is used to distinguish such an item from the empty string or the empty parenthesized list.

IMAP4 CLIENT COMMANDS

In an IMAP4 session, the client issues specific commands to prompt the server to issue responses that will facilitate the transfer of data from the server to the client.

Client Commands (any state)

The CAPABILITY, NOOP, and LOGOUT commands are valid in any of the four states. They are known as the universal commands.

- *CAPABILITY* – this command asks the server for a listing of capabilities supported by the server. After receiving this command, the server is required to send a single untagged CAPABILITY response with "IMAP4rev1" as one of the listed capabilities before the (tagged) OK response.

  C: abcd CAPABILITY
  *S: * CAPABILITY IMAP4rev1 AUTH= KERBEROS_V4*
  S: abcd OK CAPABILITY completed

- *NOOP* – this command does nothing, thus, it always succeeds. But the NOOP command can be used as a periodic poll for new messages or message status updates during a period of inactivity. It can also be used to reset any inactivity autologout timer on the server. The following illustrates how the NOOP command functions:

  C: a002 NOOP
  S: a002 NOOP completed
  * * * *
  C: a047 NOOP
  *S: * 22 EXPUNGE*
  *S: * 23 EXISTS*
  *S: * 14 FETCH (FLAGS (\Seen \Deleted))*
  S: a047 OK NOOP completed

- *LOGOUT* – the client issues this command to inform the server that it is done with the connection. The server then closes the connection.

  C: A023 LOGOUT
  *S: * BYE IMAP4rev1 Server logging out*
  S: A023 OK LOGOUT completed

  (Server and client then close the connection)

Client Commands (Non-Authenticated State)

In addition to the universal commands, the following commands are valid in the non-authenticated state:

- *AUTHENTICATE* – if the server supports the authentication mechanism mentioned by the client in the AUTHENTICATE command, it performs an authentication protocol exchange to validate and identify the client (authentication mechanisms are described in RFC 1731). The server may also negotiate an optional protection mechanism for subsequent protocol interactions. If the server is incapable of supporting the requested authentication mechanism, it rejects the AUTHENTICATE command by issuing a tagged NO response. For example:

  C: A001 AUTHENTICATE KERBEROS_V4
  S: + AmFYig= =
  C: BAcAQU5EUk VXLkNNVS5FRF UAO CAsho 84kLN3/IJmrMG+25a4DT
  +nZImJjnTNH JUtx AA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
  WwuQ1MW iy6lesKvjL5rL 9WjXUb9M wT9bp ObYLGOKi1Qh

S: + or // EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw= =
S: A001 OK KERBEROS V4 authentication successful

Authentication and protection mechanisms are optional, and an authentication mechanism may be implemented without an accompanying protection mechanism. If the first AUTHENTICATE command fails, the client may issue another AUTHENTICATE command, or it may attempt to authenticate by means of the LOGIN command.

- *LOGIN* – the client uses this command to identify itself to the server. This command carries the plaintext password that authenticates the user.

  C: a001 LOGIN KEHRES TIM
  S: a001 OK LOGIN completed

Client Commands (Authenticated State)

Together with the universal commands, the following commands are valid in the authenticated state:

- *SELECT* – this command selects a mailbox so that messages in that mailbox can be accessed. Only one mailbox can be accessed at a time in a single connection. For example:

  C: A142 SELECT INBOX
  *S: * 172 EXISTS*
  *S: * 1 RECENT*
  *S: * OK [UNSEEN 12] Message 12 is first unseen*
  *S: * OK [UIDVALIDITY 3857529045] UIDs valid*
  *S: * FLAGS (\Answered \Flagged \Deleted\Seen \Draft)*
  *S: * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited*
  S: A142 OK [READ-WRITE] SELECT completed

- *EXAMINE* – this command is almost identical to the SELECT command. However, unlike the SELECT command, the EXAMINE commmand does not allow changes to the permanent state of the mailbox. For example:

  C: A932 EXAMINE blurdybloop
  *S: * 17 EXISTS*
  *S: * 2 RECENT*
  *S: * OK [UNSEEN 8] Message 8 is first unseen*
  *S: * OK [UIDVALIDITY 3857529045] UIDs valid*
  *S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)*
  *S: * OK [PERMANENTFLAGS ()] No permanent flags permitted*
  S: A932 OK [READ-ONLY] EXAMINE completed

- *CREATE* – this command creates a mailbox with the given name. However, any attempt to create INBOX or a mailbox with a name that refers to an existing mailbox will return an error or a tagged NO response. For example:

C: A003 CREATE philippines/
S: A003 OK CREATE completed

- *DELETE* – this command permanently removes the mailbox with the given name. After the mailbox has been deleted, the server issues a tagged OK response. Any attempt to delete INBOX or a non-existent mailbox will return an error or a tagged NO response. The following is an xample of a client issuing the DELETE command:

  *C: A682 LIST "" **
  *S: * LIST () "/" blurdybloop*
  *S: * LIST (\Noselect) "/" foo*
  *S: * LIST () "/" foo/bar*
  S: A682 OK LIST completed
  C: A683 DELETE blurdybloop
  S: A683 OK DELETE completed
  C: A684 DELETE foo
  S: A684 NO Name "foo" has inferior hierarchical names
  C: A685 DELETE foo/bar
  S: A685 OK DELETE Completed
  *C: A686 LIST "" **
  *S: * LIST (\Noselect) "/" foo*
  S: A686 OK LIST completed
  C: A687 DELETE foo
  S: A687 OK DELETE Completed

- *RENAME* – this command renames a mailbox. Any attempt, however, to rename a non-existent mailbox or to rename an existing mailbox to a mailbox name that is already in use will return a tagged NO response. Renaming INBOX is allowed – by doing so, all messages in INBOX will be moved to a new mailbox with the given name, leaving INBOX empty.

  *C: A682 LIST "" **
  *S: * LIST () "/" blurdybloop*
  *S: * LIST (\Noselect) "/" foo*
  *S: * LIST () "/" foo/bar*
  S: A682 OK LIST completed
  C: A683 RENAME blurdybloop sarasoop
  S: A683 OK RENAME completed
  C: A684 RENAME foo zowie
  S: A684 OK RENAME Completed
  *C: A685 LIST "" **
  *S: * LIST () "/" sarasoop*
  *S: * LIST (\Noselect) "/" zowie*
  *S: * LIST () "/" zowie/bar*
  S: A685 OK LIST completed

- *SUBSCRIBE* – this command adds the specified mailbox name to the server's list of "active" or "subscribed" mailboxes as returned by the LSUB command.

  C: A002 SUBSCRIBE #news.comp.mail.mime
  S: A002 OK SUBSCRIBE completed

- *LIST* – this command gives a subset of names from the complete list of all the names available to the client.

  C: A101 LIST "" ""
  *S: * LIST (\Noselect) "/" ""*
  S: A101 OK LIST Completed
  C: A102 LIST #news.comp.mail.misc ""
  *S: * LIST (\Noselect) "." #news.*
  S: A102 OK LIST Completed
  C: A103 LIST /usr/staff/jones ""
  *S: * LIST (\Noselect) "/" /*
  S: A103 OK LIST Completed
  C: A202 LIST ~/Mail/ %
  *S: * LIST (\Noselect) "/" ~/Mail/foo*
  *S: * LIST () "/" ~/Mail/meetings*
  S: A202 OK LIST completed

- *LSUB* – this command gives a subset of names from the set of names that has been declared “active” or “subscribed” by the user. Zero or more untagged LSUB lines are returned.

  C: A002 LSUB "#news." "comp.mail."*
  *S: * LSUB () "." #news.comp.mail.mime*
  *S: * LSUB () "." #news.comp.mail.misc*
  S: A002 OK LSUB completed

- *STATUS* – this command asks the server for the status of the specified mailbox. It eliminates the need to get another connection in order do an EXAMINE command on another mailbox. The STATUS command does not affect the state of any messages in the queried mailbox.

  C: A042 STATUS blurdybloop (UIDNEXT MESSAGES)
  *S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)*
  S: A042 OK STATUS completed

- *APPEND* – this command appends the literal argument as a new message to the end of the specified destination mailbox. The literal argument must be in the format of an RFC 822 message for the APPEND command to be valid.

  C: A003 APPEND saved-messages (\Seen) {310}
  C: Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
  C: From: Fred Foobar <foobar@Blurdybloop.COM>
  C: Subject: afternoon meeting
  C: To: mooch@owatagu.siam.edu
  C: Message-Id: <B27397-0100000@Blurdybloop.COM>
  C: MIME-Version: 1.0
  C: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
  C:
  C: Hello Joe, do you think we can meet at 3:30 tomorrow?
  C:
  S: A003 OK APPEND completed

Client Commands (Selected State)

Commands that allow messages to be manipulated are allowed in the selected state during an IMAP4 session. These commands are:

- *CHECK* – this command asks the server for a checkpoint of the currently selected mailbox. A checkpoint refers to any implementation-dependent housekeeping associated with the mailbox (e.g. resolving the server's in-memory state of the mailbox with the state on its disk).

  C: FXXZ CHECK
  S: FXXZ OK CHECK Completed

- *CLOSE* – when this command is issued, all messages in the selected mailbox that have the \Deleted flag set are removed, and the session goes back to the authenticated state from the selected state.

  C: A341 CLOSE
  S: A341 OK CLOSE completed

- *EXPUNGE* – this command permanently removes all messages that have the \Deleted flag set from the mailbox currently selected. An untagged EXPUNGE response is sent for each message removed before returning an OK response to the client:

  C: A202 EXPUNGE
  *S: * 3 EXPUNGE*
  *S: * 3 EXPUNGE*
  *S: * 5 EXPUNGE*
  *S: * 8 EXPUNGE*
  S: A202 OK EXPUNGE completed

- *SEARCH* – this command searches the mailbox for messages matching the given search criteria. The server issues an untagged SEARCH response that contains a listing of message sequence numbers corresponding to messages that match the search criteria.

  C: A282 SEARCH FLAGGED SINCE 1-Feb-1994 NOT FROM "Smith"
  *S: * SEARCH 2 84 882*
  S: A282 OK SEARCH completed

- *FETCH* – this command retrieves data associated with a message in the mailbox. These data may include the RFC header, flags, or envelope among others. The data items to be fetched can either be a single atom or a parenthesized list.

  C: A654 FETCH 2:4 (FLAGS BODY [HEADER.FIELDS (DATE FROM)])
  *S: * 2 FETCH ....*
  *S: * 3 FETCH ....*
  *S: * 4 FETCH ....*
  S: A654 OK FETCH completed

- *STORE* – this command alters data associated with a message in the mailbox. Normally, the STORE command will return the updated value of the data with an untagged FETCH response.

  C: A003 STORE 2:4 +FLAGS (\Deleted)
  *S: * 2 FETCH FLAGS (\Deleted \Seen)*
  *S: * 3 FETCH FLAGS (\Deleted)*
  *S: * 4 FETCH FLAGS (\Deleted \Flagged \Seen)*
  S: A003 OK STORE completed

- *COPY* – when this command is issued, the specified message(s) is copied to the end of the specified destination mailbox. If the COPY command is unsuccessful for any reason, server implementations MUST restore the destination mailbox to its state before the COPY attempt.

  C: A003 COPY 2:4 MEETING
  S: A003 OK COPY completed

- *UID* – The UID command comes in two forms. In the first form, it takes as its arguments a COPY, FETCH, or STORE command with arguments that are appropriate for the associated command. However, the numbers in the message set argument are unique identifiers instead of message sequence numbers. In the second form, the UID command takes a SEARCH command with SEARCH command arguments. The interpretation of the arguments is the same as with SEARCH. The numbers returned in a SEARCH response for a UID SEARCH command, however, are unique identifiers instead of message sequence numbers.

  C: A999 UID FETCH 4827313:4828442 FLAGS
  *S: * 23 FETCH (FLAGS (\Seen) UID 4827313)*
  *S: * 24 FETCH (FLAGS (\Seen) UID 4827943)*
  *S: * 25 FETCH (FLAGS (\Seen) UID 4828442)*
  S: A999 UID FETCH completed

Client Commands (Experimental/Expansion)

Any client command prefixed with an X is an experimental command. Commands which are not part of this specification, a standard or standards-track revision of this specification, or an IESG-approved experimental protocol, must use the X prefix:

C: a441 CAPABILITY
*S: * CAPABILITY IMAP4rev1 AUTH = KERBEROS _V4 XPIG-LATIN*
S: a441 OK CAPABILITY completed
C: A442 XPIG-LATIN
*S: * XPIG-LATIN ow-nay eaking-spay ig-payatin -lay*
S: A442 OK XPIG-LATIN ompleted-cay

In IMAP4, multiple commands are allowed, that is a client can send another command even if the result of the previous command is not yet completed. Likewise, the server may begin processing another command before it completes the processing of the current command. The exception to this is when an ambiguity is likely to result because

of a command that would affect the results of other commands. If the server detects a possible ambiguity, it must complete the commands in the order given by the client.

IMAP4 SERVER RESPONSES

There are three types of server responses: status responses, server data, and command continuation request. The IMAP4 client must be prepared to accept all types of server responses at all times.

Status Responses

Status responses are OK, NO, BAD, PREAUTH, and BYE. PREAUTH and BYE are always untagged, while the others may be tagged or untagged.

Status responses may sometimes include an optional "response code". This response code consists of data inside square brackets in the form of an atom, possibly followed by a space and arguments. The response code contains additional information or status codes for client software beyond the OK/NO/BAD condition, and are defined when there is a specific action that an IMAP4 client can take based upon the additional information.

- *OK* – when tagged, this response indicates successful completion of the associated command. An untagged OK response is also used as one of three possible greetings at connection startup. For example:

  *S: * OK IMAP4rev1 server ready*
  C: A001 LOGIN fred blurdybloop
  *S: * OK [ALERT] System shutdown in 10 minutes*
  S: A001 OK LOGIN Completed

- *NO* – this response indicates an operational error message from the IMAP4 server. When tagged, it indicates that the associated command was not completed. The untagged form indicates a warning, but the command may still be completed successfully.

  C: A222 COPY 1:2 owatagusiam
  *S: * NO Disk is 98% full, please delete unnecessary data*
  S: A222 OK COPY completed

- *BAD* – a tagged BAD response indicates a protocol-level error in the client command. An untagged form indicates an error for which the associated command cannot be determined. It can also indicate an internal server failure.

  C: A443 EXPUNGE
  *S: * BAD Disk crash, attempting salvage to a new disk!*
  *S: * OK Salvage successful, no data lost*
  S: A443 OK Expunge completed

- *PREAUTH* – this response indicates that the connection has been authenticated via external means and that no LOGIN command is needed.

  *S: * PREAUTH IMAP4rev1 server logged in as Smith*

- *BYE* – this response indicates that the server is about to terminate the connection. For example:

  *S: * BYE Autologout; idle for too long*

Server and Mailbox Status

These responses facilitate the transfer of data from the server to the client. They are always untagged.

- *CAPABILITY* – this is issued by the server as a response to the CAPABILITY command. It contains a listing of capability names supported by the server.

  *S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 XPIG-LATIN*

- *LIST* – this response returns a single name matching the LIST specification in the LIST command. It is possible to have multiple LIST responses to a single LIST command.

  *S: * LIST (\Noselect) "/" ~/Mail/foo*

- *LSUB* – this command returns a single name that matches the LSUB specification in the LSUB command. It is possible to have multiple LSUB responses to a single LSUB command.

  *S: * LSUB () "." #news.comp.mail.misc*

- *STATUS* – this response returns the mailbox name specified by the STATUS command together with the requested mailbox status information.

  *S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)*

- *SEARCH* – this response contains numbers that refer to those messages that match the criteria specified by the SEARCH command.

  *S: * SEARCH 2 3 6*

- *FLAGS* – this is issued as a result of SELECT or EXAMINE command. The flag parenthesized list identifies the flags that are applicable for the specified mailbox.

  *S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)*

Mailbox Size

These responses inform the client of the changes in the size of a specific mailbox. They are always untagged.

- *EXISTS* – this response reports the number of messages in the mailbox. It occurs as a result of a SELECT or EXAMINE command, and if the size of the mailbox changes (e.g. new mail).

  *S: * 23 EXISTS*

- *RECENT* – this response reports the number of messages with the \Recent flag set. It occurs as a result of a SELECT or EXAMINE command, and if the size of the mailbox changes (e.g. receipt of new mail).

  *S: * 5 RECENT*

Message Status

These responses facilitate the transfer of message data from the server to the client. They are always untagged.

- *EXPUNGE* – this response reports that the specified message sequence number has been permanently removed from the mailbox.

  *S: * 44 EXPUNGE*

- *FETCH* – this response retrieves data about a message for the client. The data are pairs of data item names and their values in parentheses. These data items may include the RFC header, body type, or envelope, among others.

  *S: * 23 FETCH (FLAGS (\Seen) RFC822.SIZE 44827)*

Command Continuation Request

This form of response indicates that the IMAP4 server is ready to accept the continuation of a command from the client. The remainder of this response is a line of text. This response is used in the AUTHORIZATION command to transmit server data to the client, and request additional client data.

The command continuation request response is indicated by a "+" token instead of a tag. For example:

C: A001 LOGIN {11}
S: + Ready for additional command text
C: FRED FOOBAR {7}
S: + Ready for additional command text
C: fat man
S: A001 OK LOGIN completed

SECURITY ISSUES

If the LOGIN command is used in lieu of the AUTHENTICATE command, all IMAP protocol transactions are sent over the network in the clear, just like in POP3. The AUTHENTICATE command provides a high level of security by invoking an authentication mechanism. If this authentication mechanism is supported by the server, it performs an authentication protocol exchange to validate and identify the client.

The authentication protocol exchange is comprised by a series of server challenges and client answers that are specific to the authentication mechanism. A server challenge consist of a command continuation request response with the “+” token followed by a BASE64 encoded string. The client answer consists of a line made up of a BASE64

encoded string. The client issues a line with a single "*" if it wishes to cancel an authentication exchange. In this case, the server must reject the AUTHENTICATE command by issuing a tagged BAD response. If an AUTHENTICATE command fails due to invalid credentials, the server error message must not give details why the credentials are invalid. Similarly, if a LOGIN command fails, the server error message must not specify that the user name, as opposed to the password is invalid.

CONCLUSION

The advanced folder management abilities of IMAP4 alone make it an excellent alternative over POP3. With folder management, users can manage their messages more efficiently using multi-level or hierarchical mailboxes on the server (unlike POP3, which allows only a single INBOX folder for each user on the server), eliminating the need to download messages to a local hard disk. These mailboxes can be created, renamed, deleted, or moved by the user himself. In addition, IMAP4 also has provisions for sharing mailboxes or messages, and it also supports concurrent access to multiple mail servers by allowing the client to poll these servers periodically, features that are not supported by POP3.

But IMAP4 has more to offer than folder management capability. It has the capability to monitor message-status information and it can review message headers and attachments prior to downloading, providing users with the option to download the most important messages or message body parts first. Moreover, IMAP4 server-based searching also significantly reduces the processing overhead on the client machine for large searches through a message repository. IMAP4 is particulary useful to roaming users who use several computers to access their email, say a laptop, a Mac, or a PC. For example, a salesman who uses a laptop on the road may be using a PC in the office. With POP3, he must download all messages to his laptop's hard disk in order to read them while traveling. Upon reaching the office he must again download all the messages on the server (both the old ones and the new ones) plus their attachments to his PC's hard disk before he can read them using that machine. In this setup, messages are distributed among several client machines, making it difficult to update the mailbox. With IMAP4, messages can be manipulated on the server. Thus, the user does not need to download the messages to a local hard disk every time he moves from laptop to PC or vice versa.

However, there are some critical issues that must be considered by the gateway/network administrator when moving from POP3 to IMAP4. On such issue is the scalability of the IMAP4 server. Since the IMAP4 mail server will no longer be a simple mail drop, it will need more disk space than what is required by a POP3 server to accommodate the users' short-term mail archives.

Another issue is the capability of the IMAP4 server to support additional processing loads due to server-side folder searches that will be carried out by the users. This feature will result in higher memory, CPU, and disk I/O utilization.

A quota management scheme must also be implemented by the gateway/network administrator to ensure that there is enough disk space not only for the users' incoming messages but also for storing read messages as well. The scheme should include provisions for creating limits on disk space and for setting expiration/deletion times for old messages.