ing Group   M. Crispin
Comments: 2060   University of Washington
730   December 1996
andards Track

INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1

is Memo

ment specifies an Internet standards track protocol for the community, and requests discussion and suggestions for nts. Please refer to the current edition of the "Internet Protocol Standards" (STD 1) for the standardization state s of this protocol. Distribution of this memo is unlimited.

net Message Access Protocol, Version 4rev1 (IMAP4rev1) client to access and manipulate electronic mail messages on IMAP4rev1 permits manipulation of remote message folders, ailboxes", in a way that is functionally equivalent to local . IMAP4rev1 also provides the capability for an offline resynchronize with the server (see also [IMAP-DISC]).

includes operations for creating, deleting, and renaming ; checking for new messages; permanently removing messages; nd clearing flags; [RFC-822] and [MIME-IMB] parsing; ; and selective fetching of message attributes, texts, and thereof. Messages in IMAP4rev1 are accessed by the use of These numbers are either message sequence numbers or unique rs.

supports a single server. A mechanism for accessing tion information to support multiple IMAP4rev1 servers is in [ACAP].

does not specify a means of posting mail; this function is y a mail transfer protocol such as [SMTP].

is designed to be upwards compatible from the [IMAP2] and ed IMAP2bis protocols. In the course of the evolution of , some aspects in the earlier protocol have become obsolete. commands, responses, and data formats which an IMAP4rev1 ation may encounter when used with an earlier implementation ibed in [IMAP-OBSOLETE].

patibility issues with IMAP2bis, the most common variant of
er protocol, are discussed in [IMAP-COMPAT]. A full
n of compatibility issues with rare (and presumed extinct)
of [IMAP2] is in [IMAP-HISTORICAL]; this document is
of historical interest.

tents

otocol Specification

to Read This Document

nization of This Document

ment is written from the point of view of the implementor of
ev1 client or server. Beyond the protocol overview in
, it is not optimized for someone trying to understand the
of the protocol. The material in sections 3 through 5
the general context and definitions with which IMAP4rev1

6, 7, and 9 describe the IMAP commands, responses, and
espectively. The relationships among these are such that it
impossible to understand any of them separately. In
r, do not attempt to deduce command syntax from the command
lone; instead refer to the Formal Syntax section.

entions Used in This Document

es, "C:" and "S:" indicate lines sent by the client and
spectively.

wing terms are used in this document to signify the
nts of this specification.

or the adjective REQUIRED, means that the definition is olute requirement of the specification.

OT that the definition is an absolute prohibition of the
ication.

```
' means that there may exist valid reasons in particular
ıstances to ignore a particular item, but the full
ations MUST be understood and carefully weighed before
ng a different course.

' NOT means that there may exist valid reasons in
ular circumstances when the particular behavior is
able or even useful, but the full implications SHOULD be
tood and the case carefully weighed before implementing
havior described with this label.

r the adjective OPTIONAL, means that an item is truly
al.  One vendor may choose to include the item because a
ular marketplace requires it or because the vendor feels
t enhances the product while another vendor may omit the
tem.  An implementation which does not include a
ular option MUST be prepared to interoperate with another
entation which does include the option.

is used instead of "may" when referring to a possible
ıstance or situation, as opposed to an optional facility of
otocol.

 is used to refer to a human user, whereas "client" refers
 software being run by the user.

ction" refers to the entire sequence of client/server
ction from the initial establishment of the network
tion until its termination.  "Session" refers to the
ce of client/server interaction from the time that a mailbox
ected (SELECT or EXAMINE command) until the time that
ion ends (SELECT or EXAMINE of another mailbox, CLOSE
d, or connection termination).

cters are 7-bit US-ASCII unless otherwise specified.  Other
cter sets are indicated using a "CHARSET", as described in
-IMT] and defined in [CHARSET].  CHARSETs have important
ional semantics in addition to defining character set; refer
ese documents for more detail.
```

ocol Overview

Level

rev1 protocol assumes a reliable data stream such as
by TCP. When TCP is used, an IMAP4rev1 server listens on

ands and Responses

ev1 connection consists of the establishment of a
rver network connection, an initial greeting from the
nd client/server interactions. These client/server
ons consist of a client command, server data, and a server
n result response.

actions transmitted by client and server are in the form of
at is, strings that end with a CRLF. The protocol receiver
P4rev1 client or server is either reading a line, or is
sequence of octets with a known count followed by a line.

nt Protocol Sender and Server Protocol Receiver

t command begins an operation. Each client command is
with an identifier (typically a short alphanumeric string,
1, A0002, etc.) called a "tag". A different tag is
by the client for each command.

two cases in which a line from the client does not
a complete command. In one case, a command argument is
th an octet count (see the description of literal in String
a Formats); in the other case, the command arguments require
edback (see the AUTHENTICATE command). In either case, the
nds a command continuation request response if it is ready
ctets (if appropriate) and the remainder of the command.
onse is prefixed with the token "+".

If, instead, the server detected an error in the command, it
a BAD completion response with tag matching the command (as
bed below) to reject the command and prevent the client from
g any more of the command.

also possible for the server to send a completion response
me other command (if multiple commands are in progress), or
ed data. In either case, the command continuation request
ll pending; the client takes the appropriate action for the
se, and reads another response from the server. In all
the client MUST send a complete command (including
ing all command continuation request responses and command
uations for the command) before initiating a new command.

col receiver of an IMAP4rev1 server reads a command line
client, parses the command and its arguments, and transmits
ta and a server command completion result response.

er Protocol Sender and Client Protocol Receiver

smitted by the server to the client and status responses
ot indicate command completion are prefixed with the token
are called untagged responses.

ta MAY be sent as a result of a client command, or MAY be
aterally by the server. There is no syntactic difference
erver data that resulted from a specific command and server
were sent unilaterally.

r completion result response indicates the success or
f the operation. It is tagged with the same tag as the
mmand which began the operation. Thus, if more than one
s in progress, the tag in a server completion response
s the command to which the response applies. There are
sible server completion responses: OK (indicating success),
ating failure), or BAD (indicating protocol error such as
zed command or command syntax error).

ocol receiver of an IMAP4rev1 client reads a response line
server. It then takes action on the response based upon the
en of the response, which can be a tag, a "*", or a "+".

MUST be prepared to accept any server response at all times.
udes server data that was not requested. Server data SHOULD
ed, so that the client can reference its recorded copy
an sending a command to the server to request the data. In
of certain server data, the data MUST be recorded.

c is discussed in greater detail in the Server Responses

age Attributes

on to message text, each message has several attributes
d with it. These attributes may be retrieved individually
junction with other attributes or message texts.

age Numbers

in IMAP4rev1 are accessed by one of two numbers; the unique
r and the message sequence number.

Unique Identifier (UID) Message Attribute

value assigned to each message, which when used with the
entifier validity value (see below) forms a 64-bit value

ermanently guaranteed not to refer to any other message in
ox. Unique identifiers are assigned in a strictly ascending
n the mailbox; as each message is added to the mailbox it is
a higher UID than the message(s) which were added
y.

ssage sequence numbers, unique identifiers are not
ly contiguous. Unique identifiers also persist across
  This permits a client to resynchronize its state from a
session with the server (e.g. disconnected or offline access
 this is discussed further in [IMAP-DISC].

d with every mailbox is a unique identifier validity value,
sent in an UIDVALIDITY response code in an OK untagged
at mailbox selection time. If unique identifiers from an
ession fail to persist to this session, the unique
r validity value MUST be greater than the one used in the
ession.

Unique identifiers MUST be strictly ascending in the mailbox
 times. If the physical message store is re-ordered by a
AP agent, this requires that the unique identifiers in the
x be regenerated, since the former unique identifers are no
 strictly ascending as a result of the re-ordering. Another
ce in which unique identifiers are regenerated is if the
e store has no mechanism to store unique identifiers.
gh this specification recognizes that this may be
dable in certain server environments, it STRONGLY ENCOURAGES
e store implementation techniques that avoid this problem.

r cause of non-persistance is if the mailbox is deleted and
mailbox with the same name is created at a later date, Since
me is the same, a client may not know that this is a new
x unless the unique identifier validity is different. A
alue to use for the unique identifier validity value is a
 representation of the creation date/time of the mailbox.
alright to use a constant such as 1, but only if it
teed that unique identifiers will never be reused, even in
se of a mailbox being deleted (or renamed) and a new mailbox
 same name created at some future time.

e identifier of a message MUST NOT change during the
and SHOULD NOT change between sessions. However, if it is

ble to preserve the unique identifier of a message in a
t session, each subsequent session MUST have a new unique
r validity value that is larger than any that was used
y.

Message Sequence Number Message Attribute

e position from 1 to the number of messages in the mailbox.
tion MUST be ordered by ascending unique identifier. As
message is added, it is assigned a message sequence number
higher than the number of messages in the mailbox before
message was added.

equence numbers can be reassigned during the session. For
when a message is permanently removed (expunged) from the
the message sequence number for all subsequent messages is
ed. Similarly, a new message can be assigned a message
number that was once held by some other message prior to an

on to accessing messages by relative position in the
message sequence numbers can be used in mathematical
ons. For example, if an untagged "EXISTS 11" is received,
ously an untagged "8 EXISTS" was received, three new
have arrived with message sequence numbers of 9, 10, and 11.
xample; if message 287 in a 523 message mailbox has UID
ere are exactly 286 messages which have lesser UIDs and 236
which have greater UIDs.

s Message Attribute

zero or more named tokens associated with the message. A
et by its addition to this list, and is cleared by its
There are two types of flags in IMAP4rev1. A flag of
pe may be permanent or session-only.

flag is a flag name that is pre-defined in this
tion. All system flags begin with "\". Certain system
eleted and \Seen) have special semantics described
. The currently-defined system flags are:

n	Message has been read
wered	Message has been answered
gged	Message is "flagged" for urgent/special attention
eted	Message is "deleted" for removal by later EXPUNGE

```
ft      Message has not completed composition (marked as a
        draft).
```

ent   Message is "recently" arrived in this mailbox. This
        session is the first session to have been notified
        about this message; subsequent sessions will not see
        \Recent set for this message. This flag can not be
        altered by the client.

        If it is not possible to determine whether or not
        this session is the first session to be notified
        about a message, then that message SHOULD be
        considered recent.

        If multiple connections have the same mailbox
        selected simultaneously, it is undefined which of
        these connections will see newly-arrives messages
        with \Recent set and which will see it without
        \Recent set.

ord is defined by the server implementation. Keywords do
gin with "\". Servers MAY permit the client to define new
ds in the mailbox (see the description of the
ENTFLAGS response code for more information).

may be permanent or session-only on a per-flag basis.
ent flags are those which the client can add or remove
he message flags permanently; that is, subsequent sessions
ee any change in permanent flags. Changes to session
are valid only in that session.

The \Recent system flag is a special case of a
n flag. \Recent can not be used as an argument in a
command, and thus can not be changed at all.

rnal Date Message Attribute

nal date and time of the message on the server. This is not
and time in the [RFC-822] header, but rather a date and time
lects when the message was received. In the case of
delivered via [SMTP], this SHOULD be the date and time of
ivery of the message as defined by [SMTP]. In the case of
delivered by the IMAP4rev1 COPY command, this SHOULD be the
date and time of the source message. In the case of
delivered by the IMAP4rev1 APPEND command, this SHOULD be
and time as specified in the APPEND command description.

cases are implementation defined.

-822] Size Message Attribute

r of octets in the message, as expressed in [RFC-822]

lope Structure Message Attribute

representation of the [RFC-822] envelope information (not to
ed with an [SMTP] envelope) of the message.

Structure Message Attribute

representation of the [MIME-IMB] body structure information
ssage.

age Texts

on to being able to fetch the full [RFC-822] text of a
IMAP4rev1 permits the fetching of portions of the full
ext. Specifically, it is possible to fetch the [RFC-822]
eader, [RFC-822] message body, a [MIME-IMB] body part, or a
] header.

e and Flow Diagram

ev1 server is in one of four states. Most commands are
only certain states. It is a protocol error for the client
t a command while the command is in an inappropriate state.
ase, a server will respond with a BAD or NO (depending upon
plementation) command completion result.

Authenticated State

thenticated state, the client MUST supply authentication
ls before most commands will be permitted. This state is
hen a connection starts unless the connection has been pre-
ated.

enticated State

ticated state, the client is authenticated and MUST select a
o access before commands that affect messages will be
. This state is entered when a pre-authenticated connection

hen acceptable authentication credentials have been
 or after an error in selecting a mailbox.

cted State

ed state, a mailbox has been selected to access. This state
d when a mailbox has been successfully selected.

ut State

state, the connection is being terminated, and the server
e the connection. This state can be entered as a result of
request or by unilateral server decision.

```
+--------------------------------------+
|initial connection and server greeting|
+--------------------------------------+
          || (1)       || (2)        || (3)
          VV           ||            ||
+-----------------+    ||            ||
|non-authenticated|    ||            ||
+-----------------+    ||            ||
 || (7)   || (4)       ||            ||
 ||       VV           VV            ||
 ||     +----------------+           ||
 ||     | authenticated  |<=++       ||
 ||     +----------------+  ||       ||
 ||       || (7)   || (5)   || (6)   ||
 ||       ||       VV       ||       ||
 ||       ||    +--------+  ||       ||
 ||       ||    |selected|==++       ||
 ||       ||    +--------+           ||
 ||       ||       || (7)            ||
 VV       VV       VV                VV
+--------------------------------------+
|     logout and close connection      |
+--------------------------------------+

 connection without pre-authentication (OK greeting)
 pre-authenticated connection (PREAUTH greeting)
 rejected connection (BYE greeting)
 successful LOGIN or AUTHENTICATE command
 successful SELECT or EXAMINE command
 CLOSE command, or failed SELECT or EXAMINE command
 LOGOUT command, server shutdown, or connection closed
```

Formats

uses textual commands and responses. Data in IMAP4rev1 can
of several forms: atom, number, string, parenthesized list,

onsists of one or more non-special characters.

er

consists of one or more digit characters, and represents a
alue.

ng

is in one of two forms: literal and quoted string. The
orm is the general form of string. The quoted string form
ernative that avoids the overhead of processing a literal at
of limitations of characters that can be used in a quoted

is a sequence of zero or more octets (including CR and LF),
oted with an octet count in the form of an open brace ("{"),
r of octets, close brace ("}"), and CRLF. In the case of
transmitted from server to client, the CRLF is immediately
by the octet data. In the case of literals transmitted from
server, the client MUST wait to receive a command
ion request (described later in this document) before
he octet data (and the remainder of the command).

string is a sequence of zero or more 7-bit characters,
CR and LF, with double quote (<">) characters at each end.

string is represented as either "" (a quoted string with
acters between double quotes) or as {0} followed by CRLF (a
ith an octet count of 0).

Even if the octet count is 0, a client transmitting a
l MUST wait to receive a command continuation request.

t and Binary Strings

tual and binary mail is supported through the use of a
] content transfer encoding. IMAP4rev1 implementations MAY
8-bit or multi-octet characters in literals, but SHOULD do
hen the [CHARSET] is identified.

a BINARY body encoding is defined, unencoded binary strings
ermitted. A "binary string" is any string with NUL
s. Implementations MUST encode binary data into a textual
as BASE64 before transmitting the data. A string with an
amount of CTL characters MAY also be considered to be

nthesized List

ctures are represented as a "parenthesized list"; a sequence
tems, delimited by space, and bounded at each end by
es. A parenthesized list can contain other parenthesized
ing multiple levels of parentheses to indicate nesting.

list is represented as () -- a parenthesized list with no

al atom "NIL" represents the non-existence of a particular
that is represented as a string or parenthesized list, as
from the empty string "" or the empty parenthesized list ().

ational Considerations

box Naming

pretation of mailbox names is implementation-dependent.
the case-insensitive mailbox name INBOX is a special name
to mean "the primary mailbox for this user on this server".

box Hierarchy Naming

desired to export hierarchical mailbox names, mailbox names
eft-to-right hierarchical using a single character to
levels of hierarchy. The same hierarchy separator character
or all levels of hierarchy within a single name.

box Namespace Naming Convention

tion, the first hierarchical element of any mailbox name
ins with "#" identifies the "namespace" of the remainder of
This makes it possible to disambiguate between different

mailbox stores, each of which have their own namespaces.

ample, implementations which offer access to USENET
oups MAY use the "#news" namespace to partition the USENET
oup namespace from that of other mailboxes. Thus, the
ail.misc newsgroup would have an mailbox name of
.comp.mail.misc", and the name "comp.mail.misc" could refer
ifferent object (e.g. a user's private mailbox).

box International Naming Convention

tion, international mailbox names are specified using a
version of the UTF-7 encoding described in [UTF-7]. The
f these modifications is to correct the following problems
7:

-7 uses the "+" character for shifting; this conflicts with
common use of "+" in mailbox names, in particular USENET
sgroup names.

-7's encoding is BASE64 which uses the "/" character; this
flicts with the use of "/" as a popular hierarchy delimiter.

-7 prohibits the unencoded usage of "\"; this conflicts with
use of "\" as a popular hierarchy delimiter.

-7 prohibits the unencoded usage of "~"; this conflicts with
use of "~" in some servers as a home directory indicator.

-7 permits multiple alternate forms to represent the same
ing; in particular, printable US-ASCII chararacters can be
resented in encoded form.

ed UTF-7, printable US-ASCII characters except for "&"
themselves; that is, characters with octet values 0x20-0x25
0x7e. The character "&" (0x26) is represented by the two-
uence "&-".

characters (octet values 0x00-0x1f, 0x7f-0xff, and all
6-bit octets) are represented in modified BASE64, with a
odification from [UTF-7] that "," is used instead of "/".
BASE64 MUST NOT be used to represent any printing US-ASCII
which can represent itself.

ed to shift to modified BASE64 and "-" to shift back to US-

ll names start in US-ASCII, and MUST end in US-ASCII (that
e that ends with a Unicode 16-bit octet MUST end with a "-

ample, here is a mailbox name which mixes English, Japanese,
inese text: ~peter/mail/&ZeVnLIqe-/&U,BTFw-

lbox Size and Message Status Updates

me, a server can send data that the client did not request.
, such behavior is REQUIRED. For example, agents other than
r MAY add messages to the mailbox (e.g. new mail delivery),
e flags of message in the mailbox (e.g. simultaneous access
me mailbox by multiple agents), or even remove messages from
ox. A server MUST send mailbox size updates automatically
lbox size change is observed during the processing of a
 A server SHOULD send message flag updates automatically,
equiring the client to request such updates explicitly.
ules exist for server notification of a client about the
f messages to prevent synchronization errors; see the
on of the EXPUNGE response for more detail.

s of what implementation decisions a client makes on
ng data from the server, a client implementation MUST record
ize updates. It MUST NOT assume that any command after
ailbox selection will return the size of the mailbox.

onse when no Command in Progress

plementations are permitted to send an untagged response
or EXPUNGE) while there is no command in progress. Server
ations that send such responses MUST deal with flow control
tions. Specifically, they MUST either (1) verify that the
he data does not exceed the underlying transport's available
ze, or (2) use non-blocking writes.

logout Timer

er has an inactivity autologout timer, that timer MUST be of
30 minutes' duration. The receipt of ANY command from the
ring that interval SHOULD suffice to reset the autologout

iple Commands in Progress

t MAY send another command without waiting for the
n result response of a command, subject to ambiguity rules
w) and flow control constraints on the underlying data
Similarly, a server MAY begin processing another command
ocessing the current command to completion, subject to
rules. However, any command continuation request responses
nd continuations MUST be negotiated before any subsequent
s initiated.

tion is if an ambiguity would result because of a command
d affect the results of other commands. Clients MUST NOT
iple commands without waiting if an ambiguity would result.
rver detects a possible ambiguity, it MUST execute commands
tion in the order given by the client.

obvious example of ambiguity is when a command would affect
ts of another command; for example, a FETCH of a message's
a STORE of that same message's flags.

ious ambiguity occurs with commands that permit an untagged
esponse (commands other than FETCH, STORE, and SEARCH),
untagged EXPUNGE response can invalidate sequence numbers in
ent command. This is not a problem for FETCH, STORE, or
mmands because servers are prohibited from sending EXPUNGE
while any of those commands are in progress. Therefore, if
t sends any command other than FETCH, STORE, or SEARCH, it
for a response before sending a command with message
numbers.

le, the following non-waiting command sequences are invalid:

```
+ NOOP + STORE
+ COPY + FETCH
COPY
+ FETCH
```

wing are examples of valid non-waiting command sequences:

```
+ STORE + SEARCH + CHECK
+ COPY + EXPUNGE
```

nt Commands

commands are described in this section. Commands are
by the state in which the command is permitted. Commands
permitted in multiple states are listed in the minimum

state (for example, commands valid in authenticated and
state are listed in the authenticated state commands).

rguments, identified by "Arguments:" in the command
ons below, are described by function, not by syntax. The
yntax of command arguments is described in the Formal Syntax

nands cause specific server responses to be returned; these
ified by "Responses:" in the command descriptions below.
esponse descriptions in the Responses section for
on on these responses, and the Formal Syntax section for the
yntax of these responses. It is possible for server data to
itted as a result of any command; thus, commands that do not
lly require server data specify "no specific responses for
nand" instead of "none".

lt:" in the command description refers to the possible
atus responses to a command, and any special interpretation
status responses.

nt Commands - Any State

wing commands are valid in any state: CAPABILITY, NOOP, and

BILITY Command

: none

: REQUIRED untagged response: CAPABILITY

   OK - capability completed
   BAD - command unknown or arguments invalid

PABILITY command requests a listing of capabilities that the
supports. The server MUST send a single untagged
LITY response with "IMAP4rev1" as one of the listed
lities before the (tagged) OK response. This listing of
lities is not dependent upon connection state or user. It
refore not necessary to issue a CAPABILITY command more than
n a connection.

bility name which begins with "AUTH=" indicates that the
supports that particular authentication mechanism. All
ames are, by definition, part of this specification. For
e, the authorization capability for an experimental
ybloop" authenticator would be "AUTH=XBLURDYBLOOP" and not
=BLURDYBLOOP" or "XAUTH=XBLURDYBLOOP".

capability names refer to extensions, revisions, or
ents to this specification. See the documentation of the
LITY response for additional information. No capabilities,
the base IMAP4rev1 set defined in this specification, are
d without explicit client action to invoke the capability.

e section entitled "Client Commands -
mental/Expansion" for information about the form of site or
entation-specific capabilities.

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

Command

:   none

:   no specific responses for this command (but see below)

```
   OK - noop completed
   BAD - command unknown or arguments invalid
```

OP command always succeeds. It does nothing.

any command can return a status update as untagged data, the
ommand can be used as a periodic poll for new messages or
e status updates during a period of inactivity. The NOOP
d can also be used to reset any inactivity autologout timer
server.

```
   C: a002 NOOP
   S: a002 OK NOOP completed
      . . .
   C: a047 NOOP
   S: * 22 EXPUNGE
```

```
S: * 23 EXISTS
S: * 3 RECENT
S: * 14 FETCH (FLAGS (\Seen \Deleted))
S: a047 OK NOOP completed
```

UT Command

```
:   none

:   REQUIRED untagged response: BYE

    OK - logout completed
    BAD - command unknown or arguments invalid

GOUT command informs the server that the client is done with
nnection.  The server MUST send a BYE untagged response
 the (tagged) OK response, and then close the network
tion.

    C: A023 LOGOUT
    S: * BYE IMAP4rev1 Server logging out
    S: A023 OK LOGOUT completed
    (Server and client then close the connection)
```

nt Commands - Non-Authenticated State

```
thenticated state, the AUTHENTICATE or LOGIN command
es authentication and enter authenticated state.  The
ATE command provides a general mechanism for a variety of
ation techniques, whereas the LOGIN command uses the
al user name and plaintext password pair.

plementations MAY allow non-authenticated access to certain
.  The convention is to use a LOGIN command with the userid
s".  A password is REQUIRED.  It is implementation-dependent
irements, if any, are placed on the password and what access
ons are placed on anonymous users.

enticated (including as anonymous), it is not possible to
non-authenticated state.

on to the universal commands (CAPABILITY, NOOP, and LOGOUT),
wing commands are valid in non-authenticated state:
ATE and LOGIN.
```

ENTICATE Command

```
:  authentication mechanism name

:  continuation data can be requested

   OK - authenticate completed, now in authenticated state
   NO - authenticate failure: unsupported authentication
        mechanism, credentials rejected
  BAD - command unknown or arguments invalid,
        authentication exchange cancelled
```

THENTICATE command indicates an authentication mechanism,
s described in [IMAP-AUTH], to the server. If the server
ts the requested authentication mechanism, it performs an
tication protocol exchange to authenticate and identify the
. It MAY also negotiate an OPTIONAL protection mechanism
bsequent protocol interactions. If the requested
tication mechanism is not supported, the server SHOULD
the AUTHENTICATE command by sending a tagged NO response.

thentication protocol exchange consists of a series of
challenges and client answers that are specific to the
tication mechanism. A server challenge consists of a
d continuation request response with the "+" token followed
ASE64 encoded string. The client answer consists of a line
ting of a BASE64 encoded string. If the client wishes to
an authentication exchange, it issues a line with a single
If the server receives such an answer, it MUST reject the
TICATE command by sending a tagged BAD response.

ection mechanism provides integrity and privacy protection
connection. If a protection mechanism is negotiated, it is
d to all subsequent data sent over the connection. The
tion mechanism takes effect immediately following the CRLF
oncludes the authentication exchange for the client, and the
f the tagged OK response for the server. Once the
tion mechanism is in effect, the stream of command and
se octets is processed into buffers of ciphertext. Each
is transferred over the connection as a stream of octets
ded with a four octet field in network byte order that
ents the length of the following data. The maximum
text buffer length is defined by the protection mechanism.

tication mechanisms are OPTIONAL. Protection mechanisms are
PTIONAL; an authentication mechanism MAY be implemented
t any protection mechanism. If an AUTHENTICATE command
with a NO response, the client MAY try another

```
ntication mechanism by issuing another AUTHENTICATE command,
' attempt to authenticate by using the LOGIN command.  In
words, the client MAY request authentication types in
sing order of preference, with the LOGIN command as a last
.

   S: * OK KerberosV4 IMAP4rev1 Server
   C: A001 AUTHENTICATE KERBEROS_V4
   S: + AmFYig==
   C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
      +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
      WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
   S: + or//EoAADZI=
   C: DiAF5A4gA+oOIALuBkAAmw==
   S: A001 OK Kerberos V4 authentication successful

the line breaks in the first client answer are for editorial
y and are not in real authenticators.
```

N Command

```
:  user name
   password

:  no specific responses for this command

   OK - login completed, now in authenticated state
   NO - login failure: user name or password rejected
   BAD - command unknown or arguments invalid

GIN command identifies the client to the server and carries
aintext password authenticating this user.

   C: a001 LOGIN SMITH SESAME
   S: a001 OK LOGIN completed
```

nt Commands - Authenticated State

```
ticated state, commands that manipulate mailboxes as atomic
are permitted.  Of these commands, the SELECT and EXAMINE
will select a mailbox for access and enter selected state.

on to the universal commands (CAPABILITY, NOOP, and LOGOUT),
```

wing commands are valid in authenticated state: SELECT,
CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB,
nd APPEND.

CT Command

```
:  mailbox name

:  REQUIRED untagged responses: FLAGS, EXISTS, RECENT
   OPTIONAL OK untagged responses: UNSEEN, PERMANENTFLAGS

   OK - select completed, now in selected state
   NO - select failure, now in authenticated state: no
        such mailbox, can't access mailbox
   BAD - command unknown or arguments invalid

T command selects a mailbox so that messages in the
an be accessed.  Before returning an OK to the client,
r MUST send the following untagged data to the client:

      Defined flags in the mailbox.  See the description
      of the FLAGS response for more detail.

ISTS  The number of messages in the mailbox.  See the
      description of the EXISTS response for more detail.

CENT  The number of messages with the \Recent flag set.
      See the description of the RECENT response for more
      detail.

DVALIDITY <n>]
      The unique identifier validity value.  See the
      description of the UID command for more detail.

 the initial state of the mailbox at the client.

r SHOULD also send an UNSEEN response code in an OK
response, indicating the message sequence number of the
een message in the mailbox.

ient can not change the permanent state of one or more of
 listed in the FLAGS untagged response, the server SHOULD
RMANENTFLAGS response code in an OK untagged response,
he flags that the client can change permanently.

mailbox can be selected at a time in a connection;
ous access to multiple mailboxes requires multiple
```

ons. The SELECT command automatically deselects any
 selected mailbox before attempting the new selection.
tly, if a mailbox is selected and a SELECT command that
 attempted, no mailbox is selected.

ient is permitted to modify the mailbox, the server
efix the text of the tagged OK response with the
EAD-WRITE]" response code.

client is not permitted to modify the mailbox but is
ted read access, the mailbox is selected as read-only, and
rver MUST prefix the text of the tagged OK response to
with the "[READ-ONLY]" response code. Read-only access
h SELECT differs from the EXAMINE command in that certain
nly mailboxes MAY permit the change of permanent state on a
er (as opposed to global) basis. Netnews messages marked in
er-based .newsrc file are an example of such per-user
ent state that can be modified with read-only mailboxes.

```
   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
```

INE Command

```
:  mailbox name

:  REQUIRED untagged responses: FLAGS, EXISTS, RECENT
   OPTIONAL OK untagged responses: UNSEEN, PERMANENTFLAGS

   OK - examine completed, now in selected state
   NO - examine failure, now in authenticated state: no
        such mailbox, can't access mailbox
   BAD - command unknown or arguments invalid
```

AMINE command is identical to SELECT and returns the same
; however, the selected mailbox is identified as read-only.
nges to the permanent state of the mailbox, including
er state, are permitted.

xt of the tagged OK response to the EXAMINE command MUST
with the "[READ-ONLY]" response code.

```
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
```

TE Command

: mailbox name

: no specific responses for this command

OK - create completed
NO - create failure: can't create mailbox with that name
BAD - command unknown or arguments invalid

EATE command creates a mailbox with the given name. An OK
se is returned only if a new mailbox with that name has been
d. It is an error to attempt to create INBOX or a mailbox
 name that refers to an extant mailbox. Any error in
on will return a tagged NO response.

 mailbox name is suffixed with the server's hierarchy
tor character (as returned from the server by a LIST
d), this is a declaration that the client intends to create
x names under this name in the hierarchy. Server
entations that do not require this declaration MUST ignore

 server's hierarchy separator character appears elsewhere in
me, the server SHOULD create any superior hierarchical names
re needed for the CREATE command to complete successfully.
er words, an attempt to create "foo/bar/zap" on a server in
 "/" is the hierarchy separator character SHOULD create foo/
o/bar/ if they do not already exist.

ew mailbox is created with the same name as a mailbox which

leted, its unique identifiers MUST be greater than any
identifiers used in the previous incarnation of the mailbox
the new incarnation has a different unique identifier
ty value. See the description of the UID command for more
.

```
   C: A003 CREATE owatagusiam/
   S: A003 OK CREATE completed
   C: A004 CREATE owatagusiam/blurdybloop
   S: A004 OK CREATE completed
```

the interpretation of this example depends on whether "/"
turned as the hierarchy separator from LIST. If "/" is the
chy separator, a new level of hierarchy named "owatagusiam"
member called "blurdybloop" is created. Otherwise, two
xes at the same hierarchy level are created.

TE Command

```
:  mailbox name

:  no specific responses for this command

   OK - delete completed
   NO - delete failure: can't delete mailbox with that name
   BAD - command unknown or arguments invalid
```

LETE command permanently removes the mailbox with the given
A tagged OK response is returned only if the mailbox has
leleted. It is an error to attempt to delete INBOX or a
x name that does not exist.

LETE command MUST NOT remove inferior hierarchical names.
ample, if a mailbox "foo" has an inferior "foo.bar"
ing "." is the hierarchy delimiter character), removing
MUST NOT remove "foo.bar". It is an error to attempt to
a name that has inferior hierarchical names and also has
loselect mailbox name attribute (see the description of the
esponse for more details).

permitted to delete a name that has inferior hierarchical
and does not have the \Noselect mailbox name attribute. In
ase, all messages in that mailbox are removed, and the name
cquire the \Noselect mailbox name attribute.

lue of the highest-used unique identifier of the deleted
x MUST be preserved so that a new mailbox created with the
ame will not reuse the identifiers of the former
ation, UNLESS the new incarnation has a different unique

fier validity value. See the description of the UID command
re detail.

```
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


C: A82 LIST "" *
S: * LIST () "." blurdybloop
S: * LIST () "." foo
S: * LIST () "." foo.bar
S: A82 OK LIST completed
C: A83 DELETE blurdybloop
S: A83 OK DELETE completed
C: A84 DELETE foo
S: A84 OK DELETE Completed
C: A85 LIST "" *
S: * LIST () "." foo.bar
S: A85 OK LIST completed
C: A86 LIST "" %
S: * LIST (\Noselect) "." foo
S: A86 OK LIST completed
```

ME Command

```
:  existing mailbox name
   new mailbox name

:  no specific responses for this command

   OK - rename completed
   NO - rename failure: can't rename mailbox with that name,
        can't rename to mailbox with that name
   BAD - command unknown or arguments invalid

NAME command changes the name of a mailbox.  A tagged OK
se is returned only if the mailbox has been renamed.  It is
```

or to attempt to rename from a mailbox name that does not
or to a mailbox name that already exists. Any error in
ng will return a tagged NO response.

name has inferior hierarchical names, then the inferior
chical names MUST also be renamed. For example, a rename of
to "zap" will rename "foo/bar" (assuming "/" is the
chy delimiter character) to "zap/bar".

lue of the highest-used unique identifier of the old mailbox
UST be preserved so that a new mailbox created with the same
ill not reuse the identifiers of the former incarnation,
the new incarnation has a different unique identifier
ty value. See the description of the UID command for more
.

ng INBOX is permitted, and has special behavior. It moves
ssages in INBOX to a new mailbox with the given name,
g INBOX empty. If the server implementation supports
or hierarchical names of INBOX, these are unaffected by a
of INBOX.

```
   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
```

```
   C: Z432 LIST "" *
   S: * LIST () "." INBOX
   S: * LIST () "." INBOX.bar
   S: Z432 OK LIST completed
   C: Z433 RENAME INBOX old-mail
   S: Z433 OK RENAME completed
   C: Z434 LIST "" *
   S: * LIST () "." INBOX
   S: * LIST () "." INBOX.bar
   S: * LIST () "." old-mail
   S: Z434 OK LIST completed
```

CRIBE Command

```
:  mailbox

:  no specific responses for this command

   OK - subscribe completed
   NO - subscribe failure: can't subscribe to that name
   BAD - command unknown or arguments invalid
```

BSCRIBE command adds the specified mailbox name to the
's set of "active" or "subscribed" mailboxes as returned by
UB command. This command returns a tagged OK response only
subscription is successful.

er MAY validate the mailbox argument to SUBSCRIBE to verify
t exists. However, it MUST NOT unilaterally remove an
ng mailbox name from the subscription list even if a mailbox
t name no longer exists.

this requirement is because some server sites may routinely
a mailbox with a well-known name (e.g. "system-alerts")
its contents expire, with the intention of recreating it
ew contents are appropriate.

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

BSCRIBE Command

```
:  mailbox name

:  no specific responses for this command

   OK - unsubscribe completed
   NO - unsubscribe failure: can't unsubscribe that name
   BAD - command unknown or arguments invalid
```

SUBSCRIBE command removes the specified mailbox name from
rver's set of "active" or "subscribed" mailboxes as returned
LSUB command. This command returns a tagged OK response
f the unsubscription is successful.

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

T Command

```
:  reference name
   mailbox name with possible wildcards

:  untagged responses: LIST

   OK - list completed
   NO - list failure: can't list that reference or name
   BAD - command unknown or arguments invalid
```

ST command returns a subset of names from the complete set
names available to the client. Zero or more untagged LIST
s are returned, containing the name attributes, hierarchy
ter, and name; see the description of the LIST reply for
etail.

ST command SHOULD return its data quickly, without undue
  For example, it SHOULD NOT go to excess trouble to
ate \Marked or \Unmarked status or perform other processing;
h name requires 1 second of processing, then a list of 1200
would take 20 minutes!

ty ("" string) reference name argument indicates that the
x name is interpreted as by SELECT. The returned mailbox

MUST match the supplied mailbox name pattern. A non-empty
nce name argument is the name of a mailbox or a level of
x hierarchy, and indicates a context in which the mailbox
s interpreted in an implementation-defined manner.

ty ("" string) mailbox name argument is a special request to
the hierarchy delimiter and the root name of the name given
reference. The value returned as the root MAY be null if
ference is non-rooted or is null. In all cases, the
chy delimiter is returned. This permits a client to get the
chy delimiter even when no mailboxes by that name currently

ference and mailbox name arguments are interpreted, in an
entation-dependent fashion, into a canonical form that
ents an unambiguous left-to-right hierarchy. The returned
x names will be in the interpreted form.

rt of the reference argument that is included in the
reted form SHOULD prefix the interpreted form. It SHOULD
e in the same form as the reference name argument. This
ermits the client to determine if the returned mailbox name
the context of the reference argument, or if something about
ilbox argument overrode the reference argument. Without
ule, the client would have to have knowledge of the server's
semantics including what characters are "breakouts" that
de a naming context.

ample, here are some examples of how references and mailbox
might be interpreted on a UNIX-based server:

```
   Reference     Mailbox Name  Interpretation
   ------------  ------------  --------------
   ~smith/Mail/  foo.*         ~smith/Mail/foo.*
   archive/      %             archive/%
   #news.        comp.mail.*   #news.comp.mail.*
   ~smith/Mail/  /usr/doc/foo  /usr/doc/foo
   archive/      ~fred/Mail/*  ~fred/Mail/*
```

rst three examples demonstrate interpretations in the
t of the reference argument. Note that "~smith/Mail" SHOULD
transformed into something like "/u2/users/smith/Mail", or
ld be impossible for the client to determine that the
retation was in the context of the reference.

aracter "*" is a wildcard, and matches zero or more
ters at this position. The character "%" is similar to "*",
does not match a hierarchy delimiter. If the "%" wildcard

last character of a mailbox name argument, matching levels
rarchy are also returned. If these levels of hierarchy are
so selectable mailboxes, they are returned with the
ect mailbox name attribute (see the description of the LIST
se for more details).

 implementations are permitted to "hide" otherwise
ible mailboxes from the wildcard characters, by preventing
n characters or names from matching a wildcard in certain
ions. For example, a UNIX-based server might restrict the
retation of "*" so that an initial "/" character does not

ecial name INBOX is included in the output from LIST, if
is supported by this server for this user and if the
ase string "INBOX" matches the interpreted reference and
x name arguments with wildcards as described above. The
ia for omitting INBOX is whether SELECT INBOX will return
e; it is not relevant whether the user's real INBOX resides
s or some other server.

```
   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
```

Command

```
:  reference name
   mailbox name with possible wildcards

:  untagged responses: LSUB

   OK - lsub completed
   NO - lsub failure: can't list that reference or name
   BAD - command unknown or arguments invalid
```

UB command returns a subset of names from the set of names
he user has declared as being "active" or "subscribed".
r more untagged LSUB replies are returned. The arguments to

re in the same form as those for LIST.

er MAY validate the subscribed names to see if they still
 If a name does not exist, it SHOULD be flagged with the
ect attribute in the LSUB response. The server MUST NOT

```
erally remove an existing mailbox name from the subscription
even if a mailbox by that name no longer exists.

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

US Command

```
:  mailbox name
   status data item names

:  untagged responses: STATUS

   OK - status completed
   NO - status failure: no status for that name
   BAD - command unknown or arguments invalid
```

```
ATUS command requests the status of the indicated mailbox.
s not change the currently selected mailbox, nor does it
 the state of any messages in the queried mailbox (in
ular, STATUS MUST NOT cause messages to lose the \Recent


ATUS command provides an alternative to opening a second
ev1 connection and doing an EXAMINE command on a mailbox to
that mailbox's status without deselecting the current
x in the first IMAP4rev1 connection.

 the LIST command, the STATUS command is not guaranteed to
t in its response.  In some implementations, the server is
d to open the mailbox read-only internally to obtain certain
 information.  Also unlike the LIST command, the STATUS
d does not accept wildcards.

rrently defined status data items that can be requested are:

ES       The number of messages in the mailbox.

         The number of messages with the \Recent flag set.

T        The next UID value that will be assigned to a new
```

message in the mailbox. It is guaranteed that this
value will not change unless new messages are added
to the mailbox; and that it will change when new
messages are added even if those new messages are
subsequently expunged.

```
IDITY    The unique identifier validity value of the
         mailbox.

         The number of messages which do not have the \Seen
         flag set.


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

ND Command

```
:  mailbox name
   OPTIONAL flag parenthesized list
   OPTIONAL date/time string
   message literal

:  no specific responses for this command

   OK - append completed
   NO - append error: can't append to that mailbox, error
        in flags or date/time or message text
   BAD - command unknown or arguments invalid
```

PEND command appends the literal argument as a new message
end of the specified destination mailbox. This argument
be in the format of an [RFC-822] message. 8-bit characters
rmitted in the message. A server implementation that is
to preserve 8-bit data properly MUST be able to reversibly
t 8-bit APPEND data to 7-bit using a [MIME-IMB] content
er encoding.

There MAY be exceptions, e.g. draft messages, in which
ed [RFC-822] header lines are omitted in the message literal
nt to APPEND. The full implications of doing so MUST be
tood and carefully weighed.

parenthesized list is specified, the flags SHOULD be set in
ting message; otherwise, the flag list of the resulting
s set empty by default.

_time is specified, the internal date SHOULD be set in the

message; otherwise, the internal date of the resulting
s set to the current date and time by default.

pend is unsuccessful for any reason, the mailbox MUST be
to its state before the APPEND attempt; no partial appending
ted.

stination mailbox does not exist, a server MUST return an
d MUST NOT automatically create the mailbox. Unless it is
hat the destination mailbox can not be created, the server
the response code "[TRYCREATE]" as the prefix of the text
gged NO response. This gives a hint to the client that it
pt a CREATE command and retry the APPEND if the CREATE is
l.

ilbox is currently selected, the normal new mail actions
cur. Specifically, the server SHOULD notify the client
ly via an untagged EXISTS response. If the server does not
e client MAY issue a NOOP command (or failing that, a CHECK
after one or more APPEND commands.

```
   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
```

the APPEND command is not used for message delivery, because
s not provide a mechanism to transfer [SMTP] envelope
ation.

nt Commands - Selected State

ed state, commands that manipulate messages in a mailbox are
.

on to the universal commands (CAPABILITY, NOOP, and LOGOUT),
uthenticated state commands (SELECT, EXAMINE, CREATE,
ENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, and

the following commands are valid in the selected state:
OSE, EXPUNGE, SEARCH, FETCH, STORE, COPY, and UID.

K Command

: none

: no specific responses for this command

   OK - check completed
   BAD - command unknown or arguments invalid

ECK command requests a checkpoint of the currently selected
x. A checkpoint refers to any implementation-dependent
eeping associated with the mailbox (e.g. resolving the
's in-memory state of the mailbox with the state on its
that is not normally executed as part of each command. A
oint MAY take a non-instantaneous amount of real time to
te. If a server implementation has no such housekeeping
erations, CHECK is equivalent to NOOP.

is no guarantee that an EXISTS untagged response will happen
esult of CHECK. NOOP, not CHECK, SHOULD be used for new
olling.

```
   C: FXXZ CHECK
   S: FXXZ OK CHECK Completed
```

E Command

: none

: no specific responses for this command

   OK - close completed, now in authenticated state
   NO - close failure: no mailbox selected
   BAD - command unknown or arguments invalid

OSE command permanently removes from the currently selected
x all messages that have the \Deleted flag set, and returns
hentiated state from selected state. No untagged EXPUNGE
ses are sent.

sages are removed, and no error is given, if the mailbox is
ed by an EXAMINE command or is otherwise selected read-only.

f a mailbox is selected, a SELECT, EXAMINE, or LOGOUT
d MAY be issued without previously issuing a CLOSE command.
LECT, EXAMINE, and LOGOUT commands implicitly close the
tly selected mailbox without doing an expunge. However,
any messages are deleted, a CLOSE-LOGOUT or CLOSE-SELECT

ıce is considerably faster than an EXPUNGE-LOGOUT or
ıE-SELECT because no untagged EXPUNGE responses (which the
would probably ignore) are sent.

```
   C: A341 CLOSE
   S: A341 OK CLOSE completed
```

ıNGE Command

: none

: untagged responses: EXPUNGE

```
   OK - expunge completed
   NO - expunge failure: can't expunge (e.g. permission
        denied)
   BAD - command unknown or arguments invalid
```

ıPUNGE command permanently removes from the currently
ed mailbox all messages that have the \Deleted flag set.
returning an OK to the client, an untagged EXPUNGE response
ıt for each message that is removed.

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

in this example, messages 3, 4, 7, and 11 had the
ed flag set. See the description of the EXPUNGE
ıse for further explanation.

ıCH Command

: OPTIONAL [CHARSET] specification
searching criteria (one or more)

: REQUIRED untagged response: SEARCH

```
   OK - search completed
   NO - search error: can't search that [CHARSET] or
```

```
     criteria
BAD - command unknown or arguments invalid
```

```
ARCH command searches the mailbox for messages that match
ven searching criteria.  Searching criteria consist of one
e search keys.  The untagged SEARCH response from the server
ns a listing of message sequence numbers corresponding to
messages that match the searching criteria.

ultiple keys are specified, the result is the intersection
unction) of all the messages that match those keys.  For
e, the criteria DELETED FROM "SMITH" SINCE 1-Feb-1994 refers
 deleted messages from Smith that were placed in the mailbox
February 1, 1994.  A search key can also be a parenthesized
f one or more search keys (e.g. for use with the OR and NOT


 implementations MAY exclude [MIME-IMB] body parts with
al content media types other than TEXT and MESSAGE from
eration in SEARCH matching.

TIONAL [CHARSET] specification consists of the word
ET" followed by a registered [CHARSET].  It indicates the
ET] of the strings that appear in the search criteria.
IMB] content transfer encodings, and [MIME-HDRS] strings in
22]/[MIME-IMB] headers, MUST be decoded before comparing
n a [CHARSET] other than US-ASCII.  US-ASCII MUST be
ted; other [CHARSET]s MAY be supported.  If the server does
pport the specified [CHARSET], it MUST return a tagged NO
se (not a BAD).

 search keys that use strings, a message matches the key if
ring is a substring of the field.  The matching is case-
itive.

fined search keys are as follows.  Refer to the Formal
 section for the precise syntactic definitions of the
nts.

ge set>  Messages with message sequence numbers
         corresponding to the specified message sequence
         number set

         All messages in the mailbox; the default initial
         key for ANDing.
```

```
ED        Messages with the \Answered flag set.

tring>    Messages that contain the specified string in the
          envelope structure's BCC field.
```

```
 <date>  Messages whose internal date is earlier than the
         specified date.

string>  Messages that contain the specified string in the
         body of the message.

ring>    Messages that contain the specified string in the
         envelope structure's CC field.

D        Messages with the \Deleted flag set.

         Messages with the \Draft flag set.

D        Messages with the \Flagged flag set.

string>  Messages that contain the specified string in the
         envelope structure's FROM field.

 <field-name> <string>
         Messages that have a header with the specified
         field-name (as defined in [RFC-822]) and that
         contains the specified string in the [RFC-822]
         field-body.

D <flag> Messages with the specified keyword set.

 <n>     Messages with an [RFC-822] size larger than the
         specified number of octets.

         Messages that have the \Recent flag set but not the
         \Seen flag.  This is functionally equivalent to
         "(RECENT UNSEEN)".

earch-key>
         Messages that do not match the specified search
         key.

         Messages that do not have the \Recent flag set.
         This is functionally equivalent to "NOT RECENT" (as
         opposed to "NOT NEW").

te>      Messages whose internal date is within the
         specified date.
```

```
arch-key1> <search-key2>
         Messages that match either search key.

         Messages that have the \Recent flag set.
         Messages that have the \Seen flag set.

FORE <date>
         Messages whose [RFC-822] Date: header is earlier
         than the specified date.

 <date>  Messages whose [RFC-822] Date: header is within the
         specified date.

NCE <date>
         Messages whose [RFC-822] Date: header is within or
         later than the specified date.

<date>   Messages whose internal date is within or later
         than the specified date.

R <n>    Messages with an [RFC-822] size smaller than the
         specified number of octets.

T <string>
         Messages that contain the specified string in the
         envelope structure's SUBJECT field.

string>  Messages that contain the specified string in the
         header or body of the message.

ring>    Messages that contain the specified string in the
         envelope structure's TO field.

essage set>
         Messages with unique identifiers corresponding to
         the specified unique identifier set.

ERED     Messages that do not have the \Answered flag set.

TED      Messages that do not have the \Deleted flag set.

T        Messages that do not have the \Draft flag set.

GED      Messages that do not have the \Flagged flag set.

ORD <flag>
         Messages that do not have the specified keyword
```

set.

Messages that do not have the \Seen flag set.

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

H Command

```
:  message set
   message data item names

:  untagged responses: FETCH

   OK - fetch completed
   NO - fetch error: can't fetch that data
   BAD - command unknown or arguments invalid
```

TCH command retrieves data associated with a message in the
x. The data items to be fetched can be either a single atom
arenthesized list.

rrently defined data items that can be fetched are:

```
         Macro equivalent to: (FLAGS INTERNALDATE
         RFC822.SIZE ENVELOPE)

         Non-extensible form of BODYSTRUCTURE.

section>]<<partial>>
         The text of a particular body section.  The section
         specification is a set of zero or more part
         specifiers delimited by periods.  A part specifier
         is either a part number or one of the following:
         HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, and
         TEXT.  An empty section specification refers to the
         entire message, including the header.

         Every message has at least one part number.
         Non-[MIME-IMB] messages, and non-multipart
         [MIME-IMB] messages with no encapsulated message,
         only have a part 1.

         Multipart messages are assigned consecutive part
         numbers, as they occur in the message.  If a
         particular part is of type message or multipart,
```

its parts MUST be indicated by a period followed by
the part number within that nested multipart part.

A part of type MESSAGE/RFC822 also has nested part numbers, referring to parts of the MESSAGE part's body.

The HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, and TEXT part specifiers can be the sole part specifier or can be prefixed by one or more numeric part specifiers, provided that the numeric part specifier refers to a part of type MESSAGE/RFC822. The MIME part specifier MUST be prefixed by one or more numeric part specifiers.

The HEADER, HEADER.FIELDS, and HEADER.FIELDS.NOT part specifiers refer to the [RFC-822] header of the message or of an encapsulated [MIME-IMT] MESSAGE/RFC822 message. HEADER.FIELDS and HEADER.FIELDS.NOT are followed by a list of field-name (as defined in [RFC-822]) names, and return a subset of the header. The subset returned by HEADER.FIELDS contains only those header fields with a field-name that matches one of the names in the list; similarly, the subset returned by HEADER.FIELDS.NOT contains only the header fields with a non-matching field-name. The field-matching is case-insensitive but otherwise exact. In all cases, the delimiting blank line between the header and the body is always included.

The MIME part specifier refers to the [MIME-IMB] header for this part.

The TEXT part specifier refers to the text body of the message, omitting the [RFC-822] header.

Here is an example of a complex message
with some of its part specifiers:

HEADER	([RFC-822] header of the message)
TEXT	MULTIPART/MIXED
1	TEXT/PLAIN
2	APPLICATION/OCTET-STREAM
3	MESSAGE/RFC822
3.HEADER	([RFC-822] header of the message)
3.TEXT	([RFC-822] text body of the message)
3.1	TEXT/PLAIN
3.2	APPLICATION/OCTET-STREAM
4	MULTIPART/MIXED
4.1	IMAGE/GIF
4.1.MIME	([MIME-IMB] header for the IMAGE/GIF)
4.2	MESSAGE/RFC822
4.2.HEADER	([RFC-822] header of the message)
4.2.TEXT	([RFC-822] text body of the message)
4.2.1	TEXT/PLAIN
4.2.2	MULTIPART/ALTERNATIVE
4.2.2.1	TEXT/PLAIN
4.2.2.2	TEXT/RICHTEXT

It is possible to fetch a substring of the
designated text. This is done by appending an open
angle bracket ("<"), the octet position of the
first desired octet, a period, the maximum number
of octets desired, and a close angle bracket (">")
to the part specifier. If the starting octet is
beyond the end of the text, an empty string is
returned.

Any partial fetch that attempts to read beyond the
end of the text is truncated as appropriate. A
partial fetch that starts at octet 0 is returned as
a partial fetch, even if this truncation happened.

Note: this means that BODY[]<0.2048> of a
1500-octet message will return BODY[]<0>
with a literal of size 1500, not BODY[].

Note: a substring fetch of a

HEADER.FIELDS or HEADER.FIELDS.NOT part
specifier is calculated after subsetting
the header.

```
         The \Seen flag is implicitly set; if this causes
         the flags to change they SHOULD be included as part
         of the FETCH responses.

EEK[<section>]<<partial>>
         An alternate form of BODY[<section>] that does not
         implicitly set the \Seen flag.

RUCTURE  The [MIME-IMB] body structure of the message.  This
         is computed by the server by parsing the [MIME-IMB]
         header fields in the [RFC-822] header and
         [MIME-IMB] headers.

PE       The envelope structure of the message.  This is
         computed by the server by parsing the [RFC-822]
         header into the component parts, defaulting various
         fields as necessary.

         Macro equivalent to: (FLAGS INTERNALDATE
         RFC822.SIZE)

         The flags that are set for this message.

         Macro equivalent to: (FLAGS INTERNALDATE
         RFC822.SIZE ENVELOPE BODY)

ALDATE   The internal date of the message.

         Functionally equivalent to BODY[], differing in the
         syntax of the resulting untagged FETCH data (RFC822
         is returned).

.HEADER  Functionally equivalent to BODY.PEEK[HEADER],
         differing in the syntax of the resulting untagged
         FETCH data (RFC822.HEADER is returned).

.SIZE    The [RFC-822] size of the message.

.TEXT    Functionally equivalent to BODY[TEXT], differing in
         the syntax of the resulting untagged FETCH data
         (RFC822.TEXT is returned).

         The unique identifier for the message.
```

```
   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
```

E Command

```
:  message set
   message data item name
   value for message data item

:  untagged responses: FETCH

   OK - store completed
   NO - store error: can't store that data
   BAD - command unknown or arguments invalid
```

ORE command alters data associated with a message in the
x. Normally, STORE will return the updated value of the
ith an untagged FETCH response. A suffix of ".SILENT" in
ta item name prevents the untagged FETCH, and the server
 assume that the client has determined the updated value
 or does not care about the updated value.

e: regardless of whether or not the ".SILENT" suffix was
d, the server SHOULD send an untagged FETCH response if a
nge to a message's flags from an external source is
erved. The intent is that the status of the flags is
erminate without a race condition.

rrently defined data items that can be stored are:

```
<flag list>
         Replace the flags for the message with the
         argument.  The new value of the flags are returned
         as if a FETCH of those flags was done.

SILENT <flag list>
         Equivalent to FLAGS, but without returning a new
         value.

 <flag list>
```

Add the argument to the flags for the message. The
new value of the flags are returned as if a FETCH
of those flags was done.

```
.SILENT <flag list>
         Equivalent to +FLAGS, but without returning a new
         value.

 <flag list>
         Remove the argument from the flags for the message.
         The new value of the flags are returned as if a
         FETCH of those flags was done.

.SILENT <flag list>
         Equivalent to -FLAGS, but without returning a new
         value.

   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
```

Command

```
:  message set
   mailbox name

:  no specific responses for this command

   OK - copy completed
   NO - copy error: can't copy those messages or to that
        name
   BAD - command unknown or arguments invalid
```

PY command copies the specified message(s) to the end of the
ied destination mailbox. The flags and internal date of the
e(s) SHOULD be preserved in the copy.

destination mailbox does not exist, a server SHOULD return
or. It SHOULD NOT automatically create the mailbox. Unless
certain that the destination mailbox can not be created, the
MUST send the response code "[TRYCREATE]" as the prefix of
xt of the tagged NO response. This gives a hint to the
that it can attempt a CREATE command and retry the COPY if
EATE is successful.

```
 COPY command is unsuccessful for any reason, server
entations MUST restore the destination mailbox to its state
 the COPY attempt.

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

Command

```
:  command name
   command arguments

:  untagged responses: FETCH, SEARCH

   OK - UID command completed
   NO - UID command error
   BAD - command unknown or arguments invalid
```

```
D command has two forms.  In the first form, it takes as its
nts a COPY, FETCH, or STORE command with arguments
riate for the associated command.  However, the numbers in
ssage set argument are unique identifiers instead of message
ce numbers.

 second form, the UID command takes a SEARCH command with
 command arguments.  The interpretation of the arguments is
me as with SEARCH; however, the numbers returned in a SEARCH
se for a UID SEARCH command are unique identifiers instead
sage sequence numbers.  For example, the command UID SEARCH
UID 443:557 returns the unique identifiers corresponding to
tersection of the message sequence number set 1:100 and the
t 443:557.

e set ranges are permitted; however, there is no guarantee
nique identifiers be contiguous.  A non-existent unique
fier within a message set range is ignored without any error
e generated.

mber after the "*" in an untagged FETCH response is always a
e sequence number, not a unique identifier, even for a UID
d response.  However, server implementations MUST implicitly
e the UID message data item as part of any FETCH response
 by a UID command, regardless of whether a UID was specified
```

essage data item to the FETCH.

```
   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
```

ent Commands - Experimental/Expansion

om> Command

: implementation defined

: implementation defined

   OK - command completed
   NO - failure
   BAD - command unknown or arguments invalid

ommand prefixed with an X is an experimental command.
nds which are not part of this specification, a standard or
rds-track revision of this specification, or an IESG-
ved experimental protocol, MUST use the X prefix.

lded untagged responses issued by an experimental command
lso be prefixed with an X. Server implementations MUST NOT
ny such untagged responses, unless the client requested it
uing the associated experimental command.

```
   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-pay atin-lay
   S: A442 OK XPIG-LATIN ompleted-cay
```

ver Responses

sponses are in three forms: status responses, server data,
nd continuation request. The information contained in a
sponse, identified by "Contents:" in the response
ons below, is described by function, not by syntax. The
yntax of server responses is described in the Formal Syntax

t MUST be prepared to accept any response at all times.

sponses can be tagged or untagged. Tagged status responses
the completion result (OK, NO, or BAD status) of a client
and have a tag matching the command.

us responses, and all server data, are untagged. An
response is indicated by the token "*" instead of a tag.
status responses indicate server greeting, or server status
not indicate the completion of a command (for example, an
system shutdown alert). For historical reasons, untagged
ta responses are also called "unsolicited data", although
speaking only unilateral server data is truly "unsolicited".

erver data MUST be recorded by the client when it is
this is noted in the description of that data. Such data
ritical information which affects the interpretation of all
t commands and responses (e.g. updates reflecting the
or destruction of messages).

ver data SHOULD be recorded for later reference; if the
es not need to record the data, or if recording the data has
s purpose (e.g. a SEARCH response when no SEARCH command is
ss), the data SHOULD be ignored.

e of unilateral untagged server data occurs when the IMAP
n is in selected state. In selected state, the server
e mailbox for new messages as part of command execution.
this is part of the execution of every command; hence, a
and suffices to check for new messages. If new messages are
e server sends untagged EXISTS and RECENT responses
g the new size of the mailbox. Server implementations that
tiple simultaneous access to the same mailbox SHOULD also
opriate unilateral untagged FETCH and EXPUNGE responses if
gent changes the state of any message flags or expunges any

ontinuation request responses use the token "+" instead of a
se responses are sent by the server to indicate acceptance
omplete client command and readiness for the remainder of
nd.

er Responses - Status Responses

sponses are OK, NO, BAD, PREAUTH and BYE. OK, NO, and BAD gged or untagged. PREAUTH and BYE are always untagged.

sponses MAY include an OPTIONAL "response code". A response
ists of data inside square brackets in the form of an atom,
followed by a space and arguments. The response code

```
additional information or status codes for client software
e OK/NO/BAD condition, and are defined when there is a
action that a client can take based upon the additional
on.

ntly defined response codes are:

         The human-readable text contains a special alert
         that MUST be presented to the user in a fashion
         that calls the user's attention to the message.

E        Followed by a mailbox name and a new mailbox name.
         A SELECT or EXAMINE is failing because the target
         mailbox name no longer exists because it was
         renamed to the new mailbox name.  This is a hint to
         the client that the operation can succeed if the
         SELECT or EXAMINE is reissued with the new mailbox
         name.

         The human-readable text represents an error in
         parsing the [RFC-822] header or [MIME-IMB] headers
         of a message in the mailbox.

ENTFLAGS Followed by a parenthesized list of flags,
         indicates which of the known flags that the client
         can change permanently.  Any flags that are in the
         FLAGS untagged response, but not the PERMANENTFLAGS
         list, can not be set permanently.  If the client
         attempts to STORE a flag that is not in the
         PERMANENTFLAGS list, the server will either reject
         it with a NO reply or store the state for the
         remainder of the current session only.  The
         PERMANENTFLAGS list can also include the special
         flag \*, which indicates that it is possible to
         create new keywords by attempting to store those
         flags in the mailbox.

NLY      The mailbox is selected read-only, or its access
         while selected has changed from read-write to
         read-only.

RITE     The mailbox is selected read-write, or its access
         while selected has changed from read-only to
```

read-write.

```
ATE      An APPEND or COPY attempt is failing because the
         target mailbox does not exist (as opposed to some
         other reason).  This is a hint to the client that
         the operation can succeed if the mailbox is first
         created by the CREATE command.

IDITY    Followed by a decimal number, indicates the unique
         identifier validity value.

         Followed by a decimal number, indicates the number
         of the first message without the \Seen flag set.
```

onal response codes defined by particular client or server
entations SHOULD be prefixed with an "X" until they are
to a revision of this protocol. Client implementations
ignore response codes that they do not recognize.

esponse

```
   OPTIONAL response code
   human-readable text
```

response indicates an information message from the server.
agged, it indicates successful completion of the associated
d. The human-readable text MAY be presented to the user as
ormation message. The untagged form indicates an
ation-only message; the nature of the information MAY be
ted by a response code.

tagged form is also used as one of three possible greetings
nection startup. It indicates that the connection is not
thenticated and that a LOGIN command is needed.

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

esponse

```
ts:   OPTIONAL response code
      human-readable text
```

response indicates an operational error message from the
. When tagged, it indicates unsuccessful completion of the
ated command. The untagged form indicates a warning; the
d can still complete successfully. The human-readable text
bes the condition.

```
   C: A222 COPY 1:2 owatagusiam
   S: * NO Disk is 98% full, please delete unnecessary data
   S: A222 OK COPY completed
   C: A223 COPY 3:200 blurdybloop
   S: * NO Disk is 98% full, please delete unnecessary data
   S: * NO Disk is 99% full, please delete unnecessary data
   S: A223 NO COPY failed: disk is full
```

Response

```
   OPTIONAL response code
   human-readable text
```

D response indicates an error message from the server. When
, it reports a protocol-level error in the client's command;
g indicates the command that caused the error. The untagged
ndicates a protocol-level error for which the associated
d can not be determined; it can also indicate an internal
failure. The human-readable text describes the condition.

```
   C: ...very long command line...
   S: * BAD Command line too long
   C: ...empty line...
   S: * BAD Empty command line
   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
```

UTH Response

```
   OPTIONAL response code
   human-readable text
```

EAUTH response is always untagged, and is one of three
le greetings at connection startup. It indicates that the
tion has already been authenticated by external means and
o LOGIN command is needed.

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

Response

OPTIONAL response code
human-readable text

E response is always untagged, and indicates that the server
ut to close the connection. The human-readable text MAY be
yed to the user in a status report by the client. The BYE
se is sent under one of four conditions:

as part of a normal logout sequence. The server will close
the connection after sending the tagged OK response to the
LOGOUT command.

as a panic shutdown announcement. The server closes the
connection immediately.

as an announcement of an inactivity autologout. The server
closes the connection immediately.

as one of three possible greetings at connection startup,
indicating that the server is not willing to accept a
connection from this client. The server closes the
connection immediately.

fference between a BYE that occurs as part of a normal
sequence (the first case) and a BYE that occurs because of
ure (the other three cases) is that the connection closes
ately in the failure case.

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

er Responses - Server and Mailbox Status

ponses are always untagged. This is how server and mailbox
ta are transmitted from the server to the client. Many of
ponses typically result from a command with the same name.

BILITY Response

   capability listing

PABILITY response occurs as a result of a CAPABILITY
d. The capability listing contains a space-separated
g of capability names that the server supports. The
lity listing MUST include the atom "IMAP4rev1".

bility name which begins with "AUTH=" indicates that the supports that particular authentication mechanism.

capability names indicate that the server supports an
ion, revision, or amendment to the IMAP4rev1 protocol.
responses MUST conform to this document until the client
a command that uses the associated capability.

lity names MUST either begin with "X" or be standard or
rds-track IMAP4rev1 extensions, revisions, or amendments
ered with IANA. A server MUST NOT offer unregistered or
andard capability names, unless such names are prefixed with
.

implementations SHOULD NOT require any capability name
than "IMAP4rev1", and MUST ignore any unknown capability

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

Response

name attributes
hierarchy delimiter
name

ST response occurs as a result of a LIST command. It
s a single name that matches the LIST specification. There
multiple LIST responses for a single LIST command.

ame attributes are defined:

eriors	It is not possible for any child levels of hierarchy to exist under this name; no child levels exist now and none can be created in the future.
ect	It is not possible to use this name as a selectable mailbox.
d	The mailbox has been marked "interesting" by the server; the mailbox probably contains messages that have been added since the last time the mailbox was selected.
ked	The mailbox does not contain any additional messages since the last time the mailbox was

selected.

is not feasible for the server to determine whether the
x is "interesting" or not, or if the name is a \Noselect
the server SHOULD NOT send either \Marked or \Unmarked.

erarchy delimiter is a character used to delimit levels of
rchy in a mailbox name. A client can use it to create child
xes, and to search higher or lower levels of naming
rchy. All children of a top-level hierarchy node MUST use
me separator character. A NIL hierarchy delimiter means
o hierarchy exists; the name is a "flat" name.

me represents an unambiguous left-to-right hierarchy, and
e valid for use as a reference in LIST and LSUB commands.
\Noselect is indicated, the name MUST also be valid as an
argument for commands, such as SELECT, that accept mailbox

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

Response

```
   name attributes
   hierarchy delimiter
   name
```

UB response occurs as a result of an LSUB command. It
s a single name that matches the LSUB specification. There
multiple LSUB responses for a single LSUB command. The
s identical in format to the LIST response.

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

US Response

```
   name
   status parenthesized list
```

ATUS response occurs as a result of an STATUS command. It
s the mailbox name that matches the STATUS specification and
quested mailbox status information.

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

CH Response

```
   zero or more numbers
```

ARCH response occurs as a result of a SEARCH or UID SEARCH
d. The number(s) refer to those messages that match the
criteria. For SEARCH, these are message sequence numbers;
D SEARCH, these are unique identifiers. Each number is
ted by a space.

```
   S: * SEARCH 2 3 6
```

S Response

   flag parenthesized list

AGS response occurs as a result of a SELECT or EXAMINE
d. The flag parenthesized list identifies the flags (at a
m, the system-defined flags) that are applicable for this
x. Flags other than the system flags can also exist,
ing on server implementation.

date from the FLAGS response MUST be recorded by the client.

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

er Responses - Mailbox Size

ponses are always untagged. This is how changes in the size
ilbox are trasnmitted from the server to the client.
ly following the "*" token is a number that represents a
ount.

TS Response

   none

ISTS response reports the number of messages in the mailbox.
esponse occurs as a result of a SELECT or EXAMINE command,
the size of the mailbox changes (e.g. new mail).

date from the EXISTS response MUST be recorded by the
.

```
   S: * 23 EXISTS
```

NT Response

ts: none

CENT response reports the number of messages with the
t flag set. This response occurs as a result of a SELECT or
E command, and if the size of the mailbox changes (e.g. new

e: It is not guaranteed that the message sequence numbers of
ent messages will be a contiguous range of the highest n
sages in the mailbox (where n is the value reported by the
ENT response). Examples of situations in which this is not
case are: multiple clients having the same mailbox open
e first session to be notified will see it as recent, others
l probably see it as non-recent), and when the mailbox is
ordered by a non-IMAP agent.

only reliable way to identify recent messages is to look at
sage flags to see which have the \Recent flag set, or to do
EARCH RECENT.

update from the RECENT response MUST be recorded by the
ent.

```
   S: * 5 RECENT
```

er Responses - Message Status

ponses are always untagged. This is how message data are
ed from the server to the client, often as a result of a
ith the same name. Immediately following the "*" token is a
at represents a message sequence number.

NGE Response

   none

PUNGE response reports that the specified message sequence
has been permanently removed from the mailbox. The message
ce number for each successive message in the mailbox is
ately decremented by 1, and this decrement is reflected in
e sequence numbers in subsequent responses (including other

ed EXPUNGE responses).

esult of the immediate decrement rule, message sequence
s that appear in a set of successive EXPUNGE responses
upon whether the messages are removed starting from lower

s to higher numbers, or from higher numbers to lower
s. For example, if the last 5 messages in a 9-message
x are expunged; a "lower to higher" server will send five
ed EXPUNGE responses for message sequence number 5, whereas
her to lower server" will send successive untagged EXPUNGE
ses for message sequence numbers 9, 8, 7, 6, and 5.

UNGE response MUST NOT be sent when no command is in
ss; nor while responding to a FETCH, STORE, or SEARCH
d. This rule is necessary to prevent a loss of
onization of message sequence numbers between client and
.

date from the EXPUNGE response MUST be recorded by the
.

```
   S: * 44 EXPUNGE
```

H Response

   message data

TCH response returns data about a message to the client.
ta are pairs of data item names and their values in
heses. This response occurs as the result of a FETCH or
command, as well as by unilateral server decision (e.g. flag
s).

rrent data items are:

         A form of BODYSTRUCTURE without extension data.

section>]<<origin_octet>>
         A string expressing the body contents of the
         specified section. The string SHOULD be
         interpreted by the client according to the content
         transfer encoding, body type, and subtype.

         If the origin octet is specified, this string is a
         substring of the entire body contents, starting at
         that origin octet. This means that BODY[]<0> MAY
         be truncated, but BODY[] is NEVER truncated.

8-bit textual data is permitted if a [CHARSET]
identifier is part of the body parameter
parenthesized list for this section. Note that
headers (part specifiers HEADER or MIME, or the
header portion of a MESSAGE/RFC822 part), MUST be

7-bit; 8-bit characters are not permitted in headers. Note also that the blank line at the end of the header is always included in header data.

Non-textual data such as binary data MUST be transfer encoded into a textual form such as BASE64 prior to being sent to the client. To derive the original binary data, the client MUST decode the transfer encoded string.

RUCTURE A parenthesized list that describes the [MIME-IMB] body structure of a message. This is computed by the server by parsing the [MIME-IMB] header fields, defaulting various fields as necessary.

For example, a simple text message of 48 lines and 2279 octets can have a body structure of: ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48)

Multiple parts are indicated by parenthesis nesting. Instead of a body type as the first element of the parenthesized list there is a nested body. The second element of the parenthesized list is the multipart subtype (mixed, digest, parallel, alternative, etc.).

For example, a two part message consisting of a text and a BASE645-encoded text attachment can have a body structure of: (("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 1152 23)("TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME" "cc.diff") "<960723163407.20117h@cac.washington.edu>" "Compiler diff" "BASE64" 4554 73) "MIXED"))

Extension data follows the multipart subtype. Extension data is never returned with the BODY fetch, but can be returned with a BODYSTRUCTURE fetch. Extension data, if present, MUST be in the defined order.

The extension data of a multipart body part are in the following order:

body parameter parenthesized list
   A parenthesized list of attribute/value pairs
   [e.g. ("foo" "bar" "baz" "rag") where "bar" is
   the value of "foo" and "rag" is the value of

   "baz"] as defined in [MIME-IMB].

body disposition
   A parenthesized list, consisting of a
   disposition type string followed by a
   parenthesized list of disposition
   attribute/value pairs. The disposition type and
   attribute names will be defined in a future
   standards-track revision to [DISPOSITION].

body language
   A string or parenthesized list giving the body
   language value as defined in [LANGUAGE-TAGS].

Any following extension data are not yet defined in
this version of the protocol. Such extension data
can consist of zero or more NILs, strings, numbers,
or potentially nested parenthesized lists of such
data. Client implementations that do a
BODYSTRUCTURE fetch MUST be prepared to accept such
extension data. Server implementations MUST NOT
send such extension data until it has been defined
by a revision of this protocol.

The basic fields of a non-multipart body part are
in the following order:

body type
   A string giving the content media type name as
   defined in [MIME-IMB].

body subtype
   A string giving the content subtype name as
   defined in [MIME-IMB].

body parameter parenthesized list
   A parenthesized list of attribute/value pairs
   [e.g. ("foo" "bar" "baz" "rag") where "bar" is
   the value of "foo" and "rag" is the value of
   "baz"] as defined in [MIME-IMB].

body id
   A string giving the content id as defined in

   [MIME-IMB].

body description
   A string giving the content description as
   defined in [MIME-IMB].

body encoding
   A string giving the content transfer encoding as
   defined in [MIME-IMB].

body size
   A number giving the size of the body in octets.
   Note that this size is the size in its transfer
   encoding and not the resulting size after any
   decoding.

A body type of type MESSAGE and subtype RFC822
contains, immediately after the basic fields, the
envelope structure, body structure, and size in
text lines of the encapsulated message.

A body type of type TEXT contains, immediately
after the basic fields, the size of the body in
text lines. Note that this size is the size in its
content transfer encoding and not the resulting
size after any decoding.

Extension data follows the basic fields and the
type-specific fields listed above. Extension data
is never returned with the BODY fetch, but can be
returned with a BODYSTRUCTURE fetch. Extension
data, if present, MUST be in the defined order.

The extension data of a non-multipart body part are
in the following order:

body MD5
   A string giving the body MD5 value as defined in
   [MD5].

body disposition
   A parenthesized list with the same content and
   function as the body disposition for a multipart
   body part.

body language
   A string or parenthesized list giving the body
   language value as defined in [LANGUAGE-TAGS].

Any following extension data are not yet defined in
this version of the protocol, and would be as
described above under multipart extension data.

PE A parenthesized list that describes the envelope structure of a message. This is computed by the server by parsing the [RFC-822] header into the component parts, defaulting various fields as necessary.

The fields of the envelope structure are in the following order: date, subject, from, sender, reply-to, to, cc, bcc, in-reply-to, and message-id. The date, subject, in-reply-to, and message-id fields are strings. The from, sender, reply-to, to, cc, and bcc fields are parenthesized lists of address structures.

An address structure is a parenthesized list that describes an electronic mail address. The fields of an address structure are in the following order: personal name, [SMTP] at-domain-list (source route), mailbox name, and host name.

[RFC-822] group syntax is indicated by a special form of address structure in which the host name field is NIL. If the mailbox name field is also NIL, this is an end of group marker (semi-colon in RFC 822 syntax). If the mailbox name field is non-NIL, this is a start of group marker, and the mailbox name field holds the group name phrase.

Any field of an envelope or address structure that is not applicable is presented as NIL. Note that the server MUST default the reply-to and sender fields from the from field; a client is not expected to know to do this.

A parenthesized list of flags that are set for this message.

ALDATE A string representing the internal date of the message.

Equivalent to BODY[].

.HEADER Equivalent to BODY.PEEK[HEADER].

```
.SIZE     A number expressing the [RFC-822] size of the
          message.

.TEXT     Equivalent to BODY[TEXT].
```

A number expressing the unique identifier of the
message.

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

er Responses - Command Continuation Request

nd continuation request response is indicated by a "+" token
f a tag. This form of response indicates that the server is
accept the continuation of a command from the client. The
of this response is a line of text.

onse is used in the AUTHORIZATION command to transmit server
he client, and request additional client data. This
is also used if an argument to any command is a literal.

t is not permitted to send the octets of the literal unless
r indicates that it expects it. This permits the server to
ommands and reject errors on a line-by-line basis. The
of the command, including the CRLF that terminates a
follows the octets of the literal. If there are any
l command arguments the literal octets are followed by a
those arguments.

```
   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
   C: A044 BLURDYBLOOP {102856}
   S: A044 BAD No such command as "BLURDYBLOOP"
```

le IMAP4rev1 connection

wing is a transcript of an IMAP4rev1 connection. A long
his sample is broken for editorial clarity.

```
AP4rev1 Service Ready
gin mrc secret
 LOGIN completed
lect inbox
```

```
ISTS
 (\Answered \Flagged \Deleted \Seen \Draft)
ENT
NSEEN 17] Message 17 is the first unseen message
IDVALIDITY 3857529045] UIDs valid
```

```
 [READ-WRITE] SELECT completed
tch 12 full
TCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
rev1 WG mtg summary and minutes"
ry Gray" NIL "gray" "cac.washington.edu"))
ry Gray" NIL "gray" "cac.washington.edu"))
ry Gray" NIL "gray" "cac.washington.edu"))
NIL "imap" "cac.washington.edu"))
NIL "minutes" "CNRI.Reston.VA.US")
 Klensin" NIL "KLENSIN" "INFOODS.MIT.EDU")) NIL NIL
97-0100000@cac.washington.edu>")
("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028 92))
K FETCH completed
etch 12 body[header]
ETCH (BODY[HEADER] {350}
Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
Terry Gray <gray@cac.washington.edu>
t: IMAP4rev1 WG mtg summary and minutes
ap@cac.washington.edu
nutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@INFOODS.MIT.EDU>
e-Id: <B27397-0100000@cac.washington.edu>
ersion: 1.0
t-Type: TEXT/PLAIN; CHARSET=US-ASCII


K FETCH completed
tore 12 +flags \deleted
ETCH (FLAGS (\Seen \Deleted))
K +FLAGS completed
ogout
IMAP4rev1 server terminating connection
K LOGOUT completed
```

al Syntax

wing syntax specification uses the augmented Backus-Naur
) notation as specified in [RFC-822] with one exception; the
 used with the "#" construct is a single space (SPACE) and
r more commas.

se of alternative or optional rules in which a later rule
an earlier rule, the rule which is listed earlier MUST take

For example, "\Seen" when parsed as a flag is the \Seen
and not a flag_extension, even though "\Seen" could be
a flag_extension. Some, but not all, instances of this
noted below.

```
 noted otherwise, all alphabetic characters are case-
ve.  The use of upper or lower case characters to define
ings is for editorial clarity only.  Implementations MUST
ese strings in a case-insensitive fashion.

    ::= "(" addr_name SPACE addr_adl SPACE addr_mailbox
        SPACE addr_host ")"

    ::= nstring
        ;; Holds route from [RFC-822] route-addr if
        ;; non-NIL

    ::= nstring
        ;; NIL indicates [RFC-822] group syntax.
        ;; Otherwise, holds [RFC-822] domain name

    ::= nstring
        ;; NIL indicates end of [RFC-822] group; if
        ;; non-NIL and addr_host is NIL, holds
        ;; [RFC-822] group name.
        ;; Otherwise, holds [RFC-822] local-part

    ::= nstring
        ;; Holds phrase from [RFC-822] mailbox if
        ;; non-NIL

    ::= "A" / "B" / "C" / "D" / "E" / "F" / "G" / "H" /
        "I" / "J" / "K" / "L" / "M" / "N" / "O" / "P" /
        "Q" / "R" / "S" / "T" / "U" / "V" / "W" / "X" /
        "Y" / "Z" /
        "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" /
        "i" / "j" / "k" / "l" / "m" / "n" / "o" / "p" /
        "q" / "r" / "s" / "t" / "u" / "v" / "w" / "x" /
        "y" / "z"
        ;; Case-sensitive

    ::= "APPEND" SPACE mailbox [SPACE flag_list]
        [SPACE date_time] SPACE literal

    ::= atom / string

    ::= 1*ATOM_CHAR
    ::= <any CHAR except atom_specials>

s   ::= "(" / ")" / "{" / SPACE / CTL / list_wildcards /
        quoted_specials
    ::= "AUTHENTICATE" SPACE auth_type *(CRLF base64)

    ::= atom
        ;; Defined by [IMAP-AUTH]

    ::= *(4base64_char) [base64_terminal]

    ::= alpha / digit / "+" / "/"

nal ::= (2base64_char "==") / (3base64_char "=")

    ::= "(" body_type_1part / body_type_mpart ")"

on  ::= nstring / number / "(" 1#body_extension ")"
        ;; Future expansion.  Client implementations
        ;; MUST accept body_extension fields.  Server
        ;; implementations MUST NOT generate
        ;; body_extension fields except as defined by
        ;; future standard or standards-track
        ;; revisions of this specification.

rt  ::= body_fld_md5 [SPACE body_fld_dsp
        [SPACE body_fld_lang
        [SPACE 1#body_extension]]]
        ;; MUST NOT be returned on non-extensible
        ;; "BODY" fetch

rt  ::= body_fld_param
        [SPACE body_fld_dsp SPACE body_fld_lang
        [SPACE 1#body_extension]]
        ;; MUST NOT be returned on non-extensible
        ;; "BODY" fetch

    ::= body_fld_param SPACE body_fld_id SPACE
        body_fld_desc SPACE body_fld_enc SPACE
        body_fld_octets

c   ::= nstring

    ::= "(" string SPACE body_fld_param ")" / nil

    ::= (<"> ("7BIT" / "8BIT" / "BINARY" / "BASE64"/
        "QUOTED-PRINTABLE") <">) / string
```

```
         ::= nstring

g        ::= nstring / "(" 1#string ")"
```

```
es   ::= number

     ::= nstring

ets ::= number

am   ::= "(" 1#(string SPACE string) ")" / nil

art ::= (body_type_basic / body_type_msg / body_type_text)
        [SPACE body_ext_1part]

sic ::= media_basic SPACE body_fields
        ;; MESSAGE subtype MUST NOT be "RFC822"

art ::= 1*body SPACE media_subtype
        [SPACE body_ext_mpart]

g    ::= media_message SPACE body_fields SPACE envelope
        SPACE body SPACE body_fld_lines

xt   ::= media_text SPACE body_fields SPACE body_fld_lines

     ::= "AUTH=" auth_type / atom
        ;; New capabilities MUST begin with "X" or be
        ;; registered with IANA as standard or
        ;; standards-track

ata ::= "CAPABILITY" SPACE [1#capability SPACE] "IMAP4rev1"
        [SPACE 1#capability]
        ;; IMAP4rev1 servers which offer RFC 1730
        ;; compatibility MUST list "IMAP4" as the first
        ;; capability.

     ::= <any 7-bit US-ASCII character except NUL,
         0x01 - 0x7f>

     ::= <any 8-bit octet except NUL, 0x01 - 0xff>

     ::= tag SPACE (command_any / command_auth /
        command_nonauth / command_select) CRLF
        ;; Modal based on state

     ::= "CAPABILITY" / "LOGOUT" / "NOOP" / x_command
```

```
            ;; Valid in all states

        ::= append / create / delete / examine / list / lsub /
            rename / select / status / subscribe / unsubscribe
            ;; Valid only in Authenticated or Selected state
```

```
uth ::= login / authenticate
        ;; Valid only when in Non-Authenticated state

ct  ::= "CHECK" / "CLOSE" / "EXPUNGE" /
         copy / fetch / store / uid / search
        ;; Valid only when in Selected state

    ::= "+" SPACE (resp_text / base64)

    ::= "COPY" SPACE set SPACE mailbox

    ::= <ASCII CR, carriage return, 0x0D>

    ::= "CREATE" SPACE mailbox
        ;; Use of INBOX gives a NO error

    ::= CR LF

    ::= <any ASCII control character and DEL,
            0x00 - 0x1f, 0x7f>

    ::= date_text / <"> date_text <">

    ::= 1*2digit
        ;; Day of month

ed  ::= (SPACE digit) / 2digit
        ;; Fixed-format version of date_day

    ::= "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" /
        "Jul" / "Aug" / "Sep" / "Oct" / "Nov" / "Dec"

    ::= date_day "-" date_month "-" date_year

    ::= 4digit

    ::= <"> date_day_fixed "-" date_month "-" date_year
        SPACE time SPACE zone <">

    ::= "DELETE" SPACE mailbox
        ;; Use of INBOX gives a NO error

    ::= "0" / digit_nz
::= "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" /
    "9"
```

```
     ::= "(" env_date SPACE env_subject SPACE env_from
         SPACE env_sender SPACE env_reply_to SPACE env_to
         SPACE env_cc SPACE env_bcc SPACE env_in_reply_to
         SPACE env_message_id ")"

     ::= "(" 1*address ")" / nil

     ::= "(" 1*address ")" / nil

     ::= nstring

     ::= "(" 1*address ")" / nil

_to  ::= nstring

id   ::= nstring

     ::= "(" 1*address ")" / nil

     ::= "(" 1*address ")" / nil

     ::= nstring

     ::= "(" 1*address ")" / nil

     ::= "EXAMINE" SPACE mailbox

     ::= "FETCH" SPACE set SPACE ("ALL" / "FULL" /
         "FAST" / fetch_att / "(" 1#fetch_att ")")

     ::= "ENVELOPE" / "FLAGS" / "INTERNALDATE" /
         "RFC822" [".HEADER" / ".SIZE" / ".TEXT"] /
         "BODY" ["STRUCTURE"] / "UID" /
         "BODY" [".PEEK"] section
         ["<" number "." nz_number ">"]

     ::= "\Answered" / "\Flagged" / "\Deleted" /
         "\Seen" / "\Draft" / flag_keyword / flag_extension

on   ::= "\" atom
         ;; Future expansion.  Client implementations
         ;; MUST accept flag_extension flags.  Server
         ;; implementations MUST NOT generate
```

```
        ;; flag_extension flags except as defined by
        ;; future standard or standards-track
        ;; revisions of this specification.

    ::= atom
```

```
     ::= "(" #flag ")"

     ::= "*" SPACE (resp_cond_auth / resp_cond_bye) CRLF

ame  ::= astring

     ::= "(" 1#header_fld_name ")"

     ::= <ASCII LF, line feed, 0x0A>

     ::= "LIST" SPACE mailbox SPACE list_mailbox

     ::= 1*(ATOM_CHAR / list_wildcards) / string

ds   ::= "%" / "*"

     ::= "{" number "}" CRLF *CHAR8
         ;; Number represents the number of CHAR8 octets

     ::= "LOGIN" SPACE userid SPACE password

     ::= "LSUB" SPACE mailbox SPACE list_mailbox

     ::= "INBOX" / astring
         ;; INBOX is case-insensitive.  All case variants of
         ;; INBOX (e.g. "iNbOx") MUST be interpreted as INBOX
         ;; not as an astring.  Refer to section 5.1 for
         ;; further semantic details of mailbox names.

     ::=  "FLAGS" SPACE flag_list /
          "LIST" SPACE mailbox_list /
          "LSUB" SPACE mailbox_list /
          "MAILBOX" SPACE text /
          "SEARCH" [SPACE 1#nz_number] /
          "STATUS" SPACE mailbox SPACE
          "(" #<status_att number ")" /
          number SPACE "EXISTS" / number SPACE "RECENT"

     ::= "(" #("\Marked" / "\Noinferiors" /
         "\Noselect" / "\Unmarked" / flag_extension) ")"
         SPACE (<"> QUOTED_CHAR <"> / nil) SPACE mailbox

     ::= (<"> ("APPLICATION" / "AUDIO" / "IMAGE" /
         "MESSAGE" / "VIDEO") <">) / string)
         SPACE media_subtype
         ;; Defined in [MIME-IMT]

 e    ::= <"> "MESSAGE" <"> SPACE <"> "RFC822" <">
```

```
            ;; Defined in [MIME-IMT]

e    ::= string
            ;; Defined in [MIME-IMT]

     ::= <"> "TEXT" <"> SPACE media_subtype
            ;; Defined in [MIME-IMT]

     ::= nz_number SPACE ("EXPUNGE" /
                         ("FETCH" SPACE msg_att))

     ::= "(" 1#("ENVELOPE" SPACE envelope /
            "FLAGS" SPACE "(" #(flag / "\Recent") ")" /
            "INTERNALDATE" SPACE date_time /
            "RFC822" [".HEADER" / ".TEXT"] SPACE nstring /
            "RFC822.SIZE" SPACE number /
            "BODY" ["STRUCTURE"] SPACE body /
            "BODY" section ["<" number ">"] SPACE nstring /
            "UID" SPACE uniqueid) ")"

     ::= "NIL"

     ::= string / nil

     ::= 1*digit
            ;; Unsigned 32-bit integer
            ;; (0 <= n < 4,294,967,296)

     ::= digit_nz *digit
            ;; Non-zero unsigned 32-bit integer
            ;; (0 < n < 4,294,967,296)

     ::= astring

     ::= <"> *QUOTED_CHAR <">

     ::= <any TEXT_CHAR except quoted_specials> /
            "\" quoted_specials

als  ::= <"> / "\"

     ::= "RENAME" SPACE mailbox SPACE mailbox
            ;; Use of INBOX as a destination gives a NO error
```

```
    ::= *(continue_req / response_data) response_done

a   ::= "*" SPACE (resp_cond_state / resp_cond_bye /
        mailbox_data / message_data / capability_data)
        CRLF

e   ::= response_tagged / response_fatal

al  ::= "*" SPACE resp_cond_bye CRLF
        ;; Server closes connection immediately

ged ::= tag SPACE resp_cond_state CRLF

th  ::= ("OK" / "PREAUTH") SPACE resp_text
        ;; Authentication condition

e   ::= "BYE" SPACE resp_text

ate ::= ("OK" / "NO" / "BAD") SPACE resp_text
        ;; Status condition

    ::= ["[" resp_text_code "]" SPACE] (text_mime2 / text)
        ;; text SHOULD NOT begin with "[" or "="

de  ::= "ALERT" / "PARSE" /
        "PERMANENTFLAGS" SPACE "(" #(flag / "\*") ")" /
        "READ-ONLY" / "READ-WRITE" / "TRYCREATE" /
        "UIDVALIDITY" SPACE nz_number /
        "UNSEEN" SPACE nz_number /
        atom [SPACE 1*<any TEXT_CHAR except "]">]

    ::= "SEARCH" SPACE ["CHARSET" SPACE astring SPACE]
        1#search_key
        ;; [CHARSET] MUST be registered with IANA

    ::= "ALL" / "ANSWERED" / "BCC" SPACE astring /
        "BEFORE" SPACE date / "BODY" SPACE astring /
        "CC" SPACE astring / "DELETED" / "FLAGGED" /
        "FROM" SPACE astring /
        "KEYWORD" SPACE flag_keyword / "NEW" / "OLD" /
        "ON" SPACE date / "RECENT" / "SEEN" /
        "SINCE" SPACE date / "SUBJECT" SPACE astring /
        "TEXT" SPACE astring / "TO" SPACE astring /
        "UNANSWERED" / "UNDELETED" / "UNFLAGGED" /
        "UNKEYWORD" SPACE flag_keyword / "UNSEEN" /
        ;; Above this line were in [IMAP2]
        "DRAFT" /
```

```
"HEADER" SPACE header_fld_name SPACE astring /
"LARGER" SPACE number / "NOT" SPACE search_key /
"OR" SPACE search_key SPACE search_key /
"SENTBEFORE" SPACE date / "SENTON" SPACE date /
"SENTSINCE" SPACE date / "SMALLER" SPACE number /
```

```
        "UID" SPACE set / "UNDRAFT" / set /
        "(" 1#search_key ")"

    ::= "[" [section_text / (nz_number *["." nz_number]
        ["." (section_text / "MIME")])] "]"

    ::= "HEADER" / "HEADER.FIELDS" [".NOT"]
        SPACE header_list / "TEXT"

    ::= "SELECT" SPACE mailbox

    ::= nz_number / "*"
        ;; * is the largest number in use.  For message
        ;; sequence numbers, it is the number of messages
        ;; in the mailbox.  For unique identifiers, it is
        ;; the unique identifier of the last message in
        ;; the mailbox.

    ::= sequence_num / (sequence_num ":" sequence_num) /
        (set "," set)
        ;; Identifies a set of messages.  For message
        ;; sequence numbers, these are consecutive
        ;; numbers from 1 to the number of messages in
        ;; the mailbox
        ;; Comma delimits individual numbers, colon
        ;; delimits between two numbers inclusive.
        ;; Example: 2,4:7,9,12:* is 2,4,5,6,7,9,12,13,
        ;; 14,15 for a mailbox with 15 messages.

    ::= <ASCII SP, space, 0x20>

    ::= "STATUS" SPACE mailbox SPACE "(" 1#status_att ")"

    ::= "MESSAGES" / "RECENT" / "UIDNEXT" / "UIDVALIDITY" /
        "UNSEEN"

    ::= "STORE" SPACE set SPACE store_att_flags

ags ::= (["+" / "-"] "FLAGS" [".SILENT"]) SPACE
        (flag_list / #flag)

    ::= quoted / literal
    ::= "SUBSCRIBE" SPACE mailbox

    ::= 1*<any ATOM_CHAR except "+">

    ::= 1*TEXT_CHAR
```

```
 ::= "=?" <charset> "?" <encoding> "?"
     <encoded-text> "?="
     ;; Syntax defined in [MIME-HDRS]

::= <any CHAR except CR and LF>

::= 2digit ":" 2digit ":" 2digit
    ;; Hours minutes seconds

::= "UID" SPACE (copy / fetch / search / store)
    ;; Unique identifiers used instead of message
    ;; sequence numbers

::= nz_number
    ;; Strictly ascending

::= "UNSUBSCRIBE" SPACE mailbox

::= astring

::= "X" atom <experimental command arguments>

::= ("+" / "-") 4digit
    ;; Signed four-digit value of hhmm representing
    ;; hours and minutes west of Greenwich (that is,
    ;; (the amount that the given time differs from
    ;; Universal Time).  Subtracting the timezone
    ;; from the given time will give the UT form.
    ;; The Universal Time zone is "+0000".
```

or's Note

ment is a revision or rewrite of earlier documents, and
s the protocol specification in those documents: RFC 1730,
ed IMAP2bis.TXT document, RFC 1176, and RFC 1064.

rity Considerations

protocol transactions, including electronic mail data, are
he clear over the network unless privacy protection is
d in the AUTHENTICATE command.

error message for an AUTHENTICATE command which fails due to redentials SHOULD NOT detail why the credentials are

e LOGIN command sends passwords in the clear. This can be
y using the AUTHENTICATE command instead.

error message for a failing LOGIN command SHOULD NOT specify
user name, as opposed to the password, is invalid.

l security considerations are discussed in the section
g the AUTHENTICATE and LOGIN commands.

or's Address

rispin
and Distributed Computing
y of Washington
Aveneue NE
WA 98105-4527

06) 543-5762

C@CAC.Washington.EDU

rences

, J. "ACAP -- Application Configuration Access Protocol",
ress.

ynolds, J., and J. Postel, "Assigned Numbers", STD 2,
C/Information Sciences Institute, October 1994.

] Troost, R., and Dorner, S., "Communicating Presentation
in Internet Messages: The Content-Disposition Header",
ne 1995.

Myers, J., "IMAP4 Authentication Mechanism", RFC 1731.
lon University, December 1994.

] Crispin, M., "IMAP4 Compatibility with IMAP2bis", RFC
sity of Washington, November 1996.

Austein, R., "Synchronization Operations for Disconnected
s", Work in Progress.

ICAL] Crispin, M. "IMAP4 Compatibility with IMAP2 and
FC 1732, University of Washington, December 1994.

 Crispin, M., "Distributed Electronic Mail Models in
1733, University of Washington, December 1994.

TE] Crispin, M., "Internet Message Access Protocol -
tax", RFC 2062, University of Washington, November 1996.

pin, M., "Interactive Mail Access Protocol - Version 2",
iversity of Washington, August 1990.

GS] Alvestrand, H., "Tags for the Identification of
RFC 1766, March 1995.

 J., and M. Rose, "The Content-MD5 Header Field", RFC
r 1995.

reed, N., and N. Borenstein, "MIME (Multipurpose Internet
ons) Part One: Format of Internet Message Bodies", RFC

er 1996.

reed, N., and N. Borenstein, "MIME (Multipurpose
l Extensions) Part Two: Media Types", RFC 2046,
6.

Moore, K., "MIME (Multipurpose Internet Mail Extensions)
Message Header Extensions for Non-ASCII Text", RFC
er 1996.

ocker, D., "Standard for the Format of ARPA Internet Text
TD 11, RFC 822, University of Delaware, August 1982.

l, J., "Simple Mail Transfer Protocol", STD 10,
/Information Sciences Institute, August 1982.

smith, D., and Davis, M., "UTF-7: A Mail-Safe
on Format of Unicode", RFC 1642, July 1994.

ges from RFC 1730

S command has been added.

n the formal syntax that the "#" construct can never
tiple spaces.

syntax has been moved to a separate document.

AL command has been obsoleted.

2.HEADER.LINES, RFC822.HEADER.LINES.NOT, RFC822.PEEK, and
PEEK fetch attributes have been obsoleted.

rigin "." size ">" suffix for BODY text attributes has

R, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, and TEXT part
ave been added.

or Content-Disposition and Content-Language has been

iction on fetching nested MULTIPART parts has been

t number 0 has been obsoleted.

upported authenticators are now identified by
.

```
bility that identifies this protocol is now called
  A server that provides backwards support for RFC 1730
the "IMAP4" capability in addition to "IMAP4rev1" in its
esponse.  Because RFC-1730 required "IMAP4" to appear as
pability, it MUST listed first in the response.

ption of the mailbox name namespace convention has been


ption of the international mailbox name convention has


NEXT and UID-VALIDITY status items are now called UIDNEXT
ITY.  This is a change from the IMAP STATUS
ress and not from RFC-1730

arification that a null mailbox name argument to the LIST
rns an untagged LIST response with the hierarchy
d root of the reference argument.

erms such as "MUST", "SHOULD", and "MUST NOT".

ction which defines message attributes and more
etails the semantics of message sequence numbers, UIDs,


arification detailing the circumstances when a client may
e commands without waiting for a response, and the
s in which ambiguities may result.

commendation on server behavior for DELETE and RENAME
r hierarchical names of the given name exist.

arification that a mailbox name may not be unilaterally
 by the server, even if that mailbox name no longer


arification that LIST should return its results quickly
e delay.

arification that the date_time argument to APPEND sets
 date of the message.
```

arification on APPEND behavior when the target mailbox is
y selected mailbox.

arification that external changes to flags should be
nced via an untagged FETCH even if the current command is
the ".SILENT" suffix.

arification that COPY appends to the target mailbox.

NEWNAME response code.

the description of the untagged BYE response to clarify
s.

he reference for the body MD5 to refer to the proper RFC.

that the formal syntax contains rules which may overlap,
the event of such an overlap the rule which occurs first
ence.

the definition of body_fld_param.

mal syntax for capability_data.

that any case variant of "INBOX" must be interpreted as

that the human-readable text in resp_text should not
[" or "=".

IME references to Draft Standard documents.

\Recent semantics.

al examples.

Word Index