The BlitzMail Protocol

David Gelhar
February 21, 2008

1. INTRODUCTION

This document is the technical specification of the communication protocol between BlitzMail clients and servers. In addition to the definition of protocol commands, this edition of the document also includes definitions of BlitzMail server functionality, and some specific requirements for conforming BlitzMail clients.

Requirements are specified with the capitalized words MUST, SHOULD, or MAY, using the same definitions of these terms as the Internet Host Requirements document [RFC1123]:

MUST	This item is an absolute requirement of the specification.
SHOULD	There may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
MAY	This item is truly optional.

A BlitzMail client is required to satisfy all the MUST requirements in this document.

1.1. Functions of the BlitzMail server

The BlitzMail server is the permanent repository for each mail user's messages, mailing lists, folders, and preferences. The server also takes care of sending and receiving messages to and from both BlitzMail users and other mail systems (via SMTP). The BlitzMail client program provides the user interface for composing and reading messages, editing mailing lists, and so forth.

1.2. Multiple server architecture

A BlitzMail system may consist of multiple cooperating BlitzMail servers (running on different server machines). The existence of multiple servers is essentially transparent to users: the servers route mail among themselves automatically, so individual server hostnames never appear in mail addresses. All the cooperating servers use the same Dartmouth Name Directory (DND) server for name lookup and user validation. The user population is statically partitioned among the several BlitzMail servers; each user is assigned to a particular BlitzMail server, which holds their mailbox. The name of the user's BlitzMail server is recorded in their DND entry; BlitzMail servers use this information to route messages to the proper server, and clients use it to choose which server to connect to.

2. MAILBOX INFORMATION

The user data stored by the BlitzMail server consists of messages, folders, mailing lists, and preferences; all of these are collectively referred to as the user's mailbox.

2.1. Messages

There are four parts to a BlitzMail message: header, text, summary info, and (optional) enclosures.

The message **header** is essentially a normal RFC822-style header, with a few exceptions:

- lines within the header are terminated with <CR> (as is usual on the Mac), not <CRLF>
- local addresses within the header may consist of just a name, with embedded spaces and no hostname (Fred J. Flintstone instead of Fred.J.Flintstone@Dartmouth.EDU)
- in a type-1 message (see below) certain header fields (notably the Subject:) may contain Mac special characters

The server supports two message **text** formats: old-style BlitzMail format (type-1) and MIME format (type-2). The principal difference is that type-1 messages may have Mac special characters and arbitrarily long lines, while type-2 messages are in a form suitable for 7-bit SMTP transport. Type-1 messages are deprecated; new clients SHOULD always create type-2 (MIME) messages. Nevertheless, all clients MUST continue to display type-1 message texts correctly; in particular, all clients MUST wrap long lines for display on the user's screen.

All messages (whether type-1 or type-2) are stored on the server with <CR> line terminators. All clients MUST use <CR> terminators in the messages they create.

The **summary information** is intended to be displayed on a single line as part of a listing of the messages in a given folder. Some of the summary information is derived from the message header (for example, the sender's name), and some corresponds to the message's history (for example, delivery date and time, and read/unread status). An important summary attribute is the expiration date; when that date is reached the message is automatically erased.

A type-1 message may contain one or more **enclosures**. Enclosures are arbitrary Mac files, stored in MacHost format. When a message with enclosures leaves BlitzMail via SMTP, the server appends the enclosed file(s) to the text in BinHex format. The reverse transformation (recognizing BinHex in incoming SMTP messages and turning it into a Blitz-format enclosure) may also be performed, if the server is so configured. Blitz-format enclosures are deprecated; new clients SHOULD send MIME-format messages instead.

Messages are identified by a numeric message-id, which is unique within a given mailbox. In some contexts the folder containing the message must also be specified.

2.2. Folders

A folder is an ordered collection of messages. A message is always in exactly one folder. There are three pre-defined folders: **In Box**, **Sent Msgs**, and **Trash**; additional folders may be created by the user. Whenever a message is added to a folder (whether by initial delivery of a new message or as a result of moving the message from one folder to another), its summary appears at the end of the destination folder.

Incoming messages normally appear in the **In Box.** The user may also The user may designate a specific folder as a **Spam Folder**; if a **Spam Folder** is defined, incoming messages that are tagged as suspected spam are automatically delivered there. A copy of each outgoing message is saved in **Sent Msgs**. The **Trash** folder is a holding area for messages the user has deleted, giving the user a chance to reconsider before the message goes away for good.

Each folder may have a default expiration interval (some number of days or months).Whenever a message is added to a folder, the message's expiration date is set to the current date plus the folder's expiration interval. For example, the **Trash** might be set up with an expiration interval of 7 days, which would automatically discard messages a week after they were placed in the **Trash**.

Folders have names (which must be unique), but they are always referred to internally by folder numbers, which are small integers. The three builtin folders are numbered:

0	**Trash**
1	**Sent Msgs**
2	**In Box**

The server assigns numbers for user-created folders, choosing the smallest unused folder number.

2.3. Mailing lists

Each user may define personal mailing lists. Mailing list contents are sent to the server as Mac-format text, one list member per (<CR>-terminated) line. Mailing lists are automatically expanded by the server when the client specifies recipients.

In addition to personal mailing lists, the BlitzMail server also maintains public mailing lists. Unlike personal mailing lists, which are visible only to the user who created them,

the same public mailing lists are available to all users. Each public mailing list has an owner and a group. There are three sets of access permissions on each public list; those granted to the owner, to any member of the group, and to anyone else. The three permissions possible in each set are:

read	the right to see the list contents
write	the right to edit the list
send	the right to send to the list

Because personal list names are valid only for their owner, the expanded contents of the list appear in the header of a message directed to a personal list. In contrast, messages to public lists show only the public list name in the header.

2.4. Preferences

The preferences mechanism provides the client a simple way to store arbitrary key/value pairs associated with a given user's mailbox. Storing user preferences centrally, rather than on a local disk, lets clients provide uniform preference settings in multiple locations. Some preferences are of concern only to the client; the server's only function is to store them. There also are a number of special preferences that are used to communicate information from the client to the server, or vice versa.

The following preferences are set by the client, and read by the server:

AutoExp<n>	Default expiration for messages moved into folder <n>
ForwardTo	Forwarding address
VerboseNotify	Set to "1" to cause server to include sender and subject in new mail notifications.
SpamFolder	Folder to which spam messages (spam score exceeds **SpamThreshold** should be delivered)
SpamThreshold	Messages whose SpamAssassin score exceeds this value will be delivered to **SpamFolder** instead of InBox

The following preferences are set by the server, to be read by clients. Except as specified, clients MUST NOT modify these preferences:

Expired<n>	List of message ids expired from folder <n> (cleared by client after reading)
FoldSessionTag<n>	Value of SessionId when folder <n> was last modified
LastLogin	Text string giving date and time this session began
NewMail	Set to "1" if new mail has arrived since last WARN command
SessionId	Incremented by 1 each signon
Vacation	Set to "1" if vacation message active

3. CLIENT-SERVER INTERACTION

3.1. Connecting to the BlitzMail server

After prompting for the user's name and password (or client certificate), the client consults the DND to determine which BlitzMail server it should connect to for this user. The BLITZSERV field of the DND contains the hostname and (optional) AppleTalk zone name of the user's BlitzMail server in the format <host>[@<zone>] AppleTalk zone names are no longer used; clients MUST ignore the "@<zone>" portion of the hostname if present.

SSL connections on TCP port 2155 SHOULD be used if both the client and server support SSL; non-SSL sites MAY continue to support non-SSL connections on the "blitzmail" (2151) well-known TCP port.

The server may time out inactive client connections periodically. The client MAY include the ability to automatically reconnect the next time the user does something that requires server interaction. When automatically reconnecting, the client SHOULD NOT consult the DND again; it SHOULD always reconnect to the same server. If the user has been moved to a different server, the signon attempt will be rejected by the server; the client can then switch to the new server.

3.2. Authentication options

The available authentication methods vary by client and by site. For example, not all sites allow PKI certificate authentication, and not all clients implement Kerberos.

Authentication methods in rough order of preference are:

SSL client certificate (**CERT** command)
CRAM-MD5 challenge/response (**AMD5/PMD5** command)
Kerberos4 ticket (**KRB4** command)
Name/Password over SSL connection (**USER/UID#** & **PASS** commands)

The following authentication methods are deprecated and SHOULD NOT be used:

PASE challenge/response (limits passwords to 8 chars)
Cleartext password (**PASS)** without SSL encryption

3.3. Sending and reading messages: the current message model

The functions of reading existing messages and composing new ones are somewhat intertwined. This allows the client to, for example, create and send a message including some of the enclosures from an existing message without downloading and uploading the enclosures again.

Each BlitzMail session has a current message and a current recipient list; all the message composing and reading commands operate on the current message. The current message consists of the header, text, and (optional) enclosure list.The elements of the current message may be part of a saved message, data uploaded by the client in the process of composing a new message, or a mixture of the two.

The **MESS** command makes a saved message the current message. **HEAD**, **TEXT**, **ELIS**, and **ENCL** commands may then be issued to read the message header, text, enclosure list, and individual enclosures, respectively. The **MCAT** command retrieves a catalog describing the MIME structure of the current message, allowing the client to download selected message parts. New message text is uploaded with the **MDAT** command, new enclosures with **EDAT**. Note that there's no command to upload the message header: constructing the header is the server's responsibility. There are individual commands that let the client specify particular header fields (such as the **TOPC** command to set the Subject: field), and MIME content description fields may be included with the uploaded text.

When composing a new message from scratch, the client will first issue a **CLEA** or **CLEM** command to clear the current message, then upload the message with an **MDAT** command (plus **EDAT** if there are enclosures). A message combining new text with existing enclosures may be created by selecting the existing message with **MESS**, and filling in the new text with **MDAT**; various combinations are possible. To minimize the amount of state information (in the form of half-composed messages) the server must remember, the client SHOULD NOT upload any part of the message until all interaction with the user has been completed and the message is ready to be sent.

Note: BlitzMail-format enclosures (created by the **EDAT** command) are deprecated and SHOULD NOT be used by current clients. Attachements SHOULD be handled by creating a MIME multipart message and uploading it with **MDAT**.

The current recipient list contains the cumulative results of the **RCPT**/**RCCC**/**RBCC** commands issued since the last time the list was cleared (with **CLER** or **CLEA**). The **SEND** command sends the current message to the current recipient list. Note that **SEND** does not implicitly clear the message or recipient list; this allows the client flexibility to do things like sending another message to the same recipient list without starting over from scratch. If the client does not intend to reuse the message or recipient list, it MUST issue a **CLEA** command immediately after sending.

3.3.1. Recipient resolution

When resolving a recipient name or address, the server tries, in order: personal mailing lists, public mailing lists, the DND, and external RFC822 addresses. Mailing list expansion follows the same procedure recursively, so there may be mailing lists that refer to other lists, and so forth. The hierarchy of name resolution means that it's possible to (for example) define a personal mailing list that overrides the public list of the same name. Doing so is almost always a bad idea. The public list resolution step also checks for names of the form owner-<list-name> and <list-name>-request; they resolve to the public mailing list's owner. Note that public mailing lists may be sent to from outside BlitzMail by sending to <list-name>@<blitzhost>. For this reason, public list names must be valid as the local-part of an RFC822-format address; in particular, they may not contain spaces.

3.3.2. Forwarding and Vacation Messages

If the user sets a forwarding address, all incoming messages are redirected to the specified address (which may include multiple recipients). To enable forwarding, the client stores the desired forwarding address in the **ForwardTo** preference. If forwarding is enabled, no mail is delivered to that account. It's possible to forward incoming mail while also keeping a copy by including the special address "me" in the **ForwardTo** preference (for example **ForwardTo** "me,bob@aol.com").

The user may specify a "vacation message": text that is to be sent in response to anyone who sends the user a message. The server maintains a list of addresses that have received the vacation message; the message is sent only once to a given address. Vacation messages and forwarding are orthogonal: if a user enables both features any mail sent to them is forwarded, but a vacation message is also generated.

3.3.3. Enclosures and non-BlitzMail Recipients

When a type-1 (non-MIME) message with BlitzMail enclosures is about to leave the BlitzMail system en route to a non-BlitzMail address, the enclosures are appended to the message body in BinHex format. Similarly, incoming messages are scanned for BinHex, which is turned into BlitzMail enclosures by the server. If the non-BlitzMail address was generated as a result of a DND reference (i.e., the recipient is in the DND but has a non-BlitzMail preferred address), a copy of the message with the enclosures in BlitzMail format is created. This "enclosure clone" is delivered to the recipient's BlitzMail account; the primary message with the attached BinHex text still goes to the preferred address. Enclosure clones are always delivered to the recipient's BlitzMail account, even if that account has forwarding enabled.

Note that the use of BlitzMail enclosures is deprecated; clients SHOULD instead send attachments using MIME.

3.4. Dealing with message summaries

A folder may be thought of as a collection of message summaries. For most operations that don't involve actually opening up and reading the message, it's the message summary that the client (and the user) will be dealing with. This includes the arrival of new mail (adding a summary to the **In Box**), filing messages (moving summaries to a different folder), and expiration (removing summaries).

3.4.1. Processing new messages

When a message arrives, the message summary is appended to the end of the **In Box**. The server sends a notification to all Notify clients registered for the recipient user. If the user is signed on, the server sets the "warning present" bit in the return code for subsequent BlitzMail commands. When the BlitzMail client notices that there is new mail pending, it issues a **WARN** command; the response includes the message id of the newly-arrived message(s). The client can then issue an **MSUM** command to obtain the summaries.

New messages may arrive at any time during a BlitzMail session. Clients must take care to avoid showing a message more than once (or not at all!) as the result of race conditions. For example, if the client moves a message into the **In Box** at the same time as a new message arrives, the order of the messages in the **In Box** is indeterminate.

A relatively simple method for safely processing **In Box** updates is as follows:

- Issue a "**SUMM 1–$**" to find out about all pre-existing messages
- Whenever a new message arrives, issue a **WARN** to find the new message id. If the client does not yet have the summary for that message-id, then issue an **MSUM** to get it. The check for duplicated message-id's may seem unnecessary, but it isn't. Consider what happens if a message arrives just before the server processes the "**SUMM 1–$**": the **SUMM** command will show the new message, but there will also be a pending warning for it.

The **In Box** (and **SpamFolder**, if one is defined) are the only folders that messages appear in except as a direct result of a client command, so similar strategies are not necessary for other folders.

BlitzMail clients SHOULD incorporate a Notify client, either as part of the BlitzMail client itself or as a companion program, so the client can learn about new mail without polling the server. If a Notify client is not available, the BlitzMail client MAY poll for

new mail by periodically sending a **WARN** or **NOOP** command. The client MUST NOT poll more frequently than once every 60 seconds.

3.4.2. Expiration

All messages may have an expiration date; when that date is reached the server automatically destroys the message. The server performs the expiration check once a day (early in the morning, typically); messages whose expiration date is set to some time in the past will linger until the next time the expiration check is run. The default expiration period for newly-delivered messages is specified in the server's configuration file; a typical value is six months, but servers that are also serving IMAP clients may have a default expiration of "never", since IMAP clients aren't able to view/update the expiration settings.

The user may specify an expiration interval for all messages moved into a particular folder, or manually change the expiration date of an individual message. Note that there is no provision for the user to specify a default expiration period for all incoming mail. This omission is deliberate: it was felt that making the expiration mechanism too easy to defeat would lead to excessive storage consumption.

Whenever the server expires a message, it adds the message-id to the **Expired<n>** preference for the folder the message expired from; clients may consult this preference to determine which cached summaries are invalid.

3.4.3. Deleting and filing messages

A message is always in exactly one folder; filing commands may move a message from one folder to another, or copy it to a new folder (creating a new message).

Commands that refer to a message by message-id (such as **MESS**, **EXPR**, **MSUM**, etc.) use the syntax [<folder-num>/]<message-id>. If the folder number is omitted, the server will search for the message in the **In Box** and **Trash**, in that order. Clients SHOULD always specify the folder number when referring to a message in the **Trash**, so the server doesn't have to go through the work of searching the **In Box** first.

The **MOVE** command moves a message from one folder to another; **COPY** creates a new copy of the message in the destination folder. The delete and undelete commands (**DELE/DELX/UDEL/UDLX**), which move messages between the **In Box** and the **Trash,** remain in the protocol only for compatibility with pre-filing clients. New clients MUST use **MOVE** instead.

Each folder may have an automatic expiration setting; messages moved into that folder are set to expire a given number of days or months after the date they were moved in. **MOVE** and **COPY** check the **AutoExp<n>** preference when adding a message to a folder; if the preference exists the message's expiration date is set accordingly. For example, if

AutoExp0 is "3" and **AutoExp2** is "5M", **MOVE** will set a message's expiration to three days when moving into the **Trash**, and five months when moving into the **In Box**. Note that the **AutoExp2** setting applies only to existing messages being moved back into the **In Box**, not newly-arrived messages. Having the expiration change be done automatically by the **MOVE** command, instead of leaving it up to client control as the old **DELX/UDLX** commands did, ensures uniform results regardless of which BlitzMail client is used.

At the protocol level, the only distinction between the **Trash** and other folders is the "empty trash" (**TRSH)** command. There are only two ways messages go away: by expiration, or by moving them to the **Trash** and emptying it.

3.4.4 Auditing outgoing messages

Whenever a message is sent, a copy of the message is automatically saved in **Sent Msgs**, whether or not the sender explicitly included themselves in the recipient list. The expiration date of these "audit" copies is automatically set according to the **AutoExp1** preference. If the **AutoExp1** preference is undefined, the server initializes it to seven days when the user signs on. The relatively short default expiration makes it practical to have auditing enabled automatically without consuming too much storage.

Because messages arrive in **Sent Msgs** as a direct result of a client action (the **SEND** command), no elaborate synchronization measures are necessary when the client wants to get the summary. The message always appears at the end of **Sent Msgs**, so "**FSUM 1 $**" will retrieve the new summary.

3.4.5. Summary Caching

Clients MAY cache message summary information locally, to avoid the performance penalty of downloading all the summaries whenever a folder is opened. If a client caches summary information, it MUST make provision for updating or invalidating its cache when events beyond the client's knowledge change the folder. Such changes may be due to the arrival of new messages, message expiration, or a client signing on from a different location and modifying the folder. A client's ultimate recourse if the contents of its summary cache are suspect is to invalidate the cache and re-read all the summary information. The server maintains several preferences (**SessionId**, **Expired<n>**, and **FoldSessionTag<n>**) to assist clients in updating their cache without completely invalidating it.

A client can ensure the validity of its cache (while avoiding excessive summary downloading) using the following strategy:

- For each folder, cache locally the summaries and the value of the **SessionId** preference at the time the cache entry was recorded.

- When opening a folder, compare the value of its **FoldSessionTag<n>** preference to the **SessionId** value in the cache for that folder; if they don't match the cache is completely invalid and must be reloaded. If they do match, the client must next check the **Expired<n>** preference and delete any expired messages from its cache.

- Finally, the client must check for any messages that may have been added to the folder. If (after processing expirations) the client has <n> messages in its cache, it should do a "**SUMM <n+1>–$**" to request summaries for any additional messages.

The server sets **FoldSessionTag<n>** to the current **SessionId** whenever: the client deletes a message from the folder, a message expires *and* there isn't room in the **Expired<n>** preference to record the message id, or the **Expired<n>** preference is cleared by the client. The tag doesn't need to be set when a message is added to the folder because new messages are always added at the end, so the client is able to locate them without re-reading the entire folder. After processing the **Expired<n>** preference, the client can choose between clearing it (immediately invalidating the caches of any clients at other locations), and leaving it alone (possibly forcing the server to invalidate everybody's cache if there's insufficient room in the preference string to append information about subsequent expirations).

Extraordinary circumstances may cause the client's cached summary information to become inconsistent with the corresponding information on the server. For example, say the client moves a message from one folder to another, the server completes the operation and the client updates its cache, but a disk failure on the server (and recovery from backup tapes) causes the change to be lost. In this case the message will still be in the source folder, even though the client's cache indicates that it's in the destination folder.

To detect and recover from such occurrences, clients that do summary caching SHOULD also include the following consistency checks:

- If (after processing expirations) the client has more messages in its cache than the **SIZE** or **FLIS** command indicates exist in that folder, the cache for that folder is invalid.

- Any additional message-id's returned by the "**SUMM <n+1>–$**" command should be compared against the message-id's the client already knows about; if there is any duplication the cache is invalid.

3.5 Interrupting transfers

Clients MAY provide users the ability to cancel lengthy uploads/downloads. Only commands that involve binary data transfer (that is, commands that return an x50 intermediate status before beginning the data transfer) are interruptible. Issuing interrupt requests during other commands doesn't cause any harm, but the server will finish processing the command before servicing the interrupt, so there's no benefit either.

The client indicates that it wishes to interrupt the transfer by sending a BlitzMail Break Request. Exactly how this works depends on whether the client is using TCP or KSP. KSP clients merely send an ordinary KSP BREAK signal. TCP clients send a packet with the TCP URGENT bit set; the contents of the packet are exactly the data of a KSP BREAK (three bytes: $02 $81 $00).

Upon receiving a Break Request, the server interrupts the transfer as soon as possible, and responds to the request by sending an x70 response "x70 Break acknowledged". If it was a download that was being interrupted, the client must be able to locate this acknowledgement within the stream of downloaded data. In order to increase the likelihood of finding the mark correctly, the client MAY in this one instance ignore the usual rule of not depending on specific response texts and search for the "Break acknowledged" string as well as the "70"; the server guarantees to send precisely that text.

Once the acknowledgement has been located, the client and server are resynchronized and the client may resume sending commands.

3.6 Proxy Access

A privileged application may authenticate to the BlitzMail server as a “proxy” agent for another user, which allows the application limited access to the user’s mail account without having access to the user’s actual login credentials.

The agent MUST connect using SSL, and MUST supply an SSL client certificate. The server configuration specifies what certificate(s) are trusted proxy agents, and may also impose other conditions (such as limiting the IP addresses a particular agent may connect from). Only a limited number of commands are available via proxy login: proxy agents can view/modify preferences, create folders, and set/remove vacation messages, but not read or send mail.

4. THE BLITZMAIL PROTOCOL SPECIFICATION

BlitzMail commands consist of a string of characters, terminated by <CRLF>. All command names are four characters long; a single space separates the command name from any command parameters. Strict alternation of commands and responses is observed; the client may not issue another command until the server has returned the response from the current command, and the server may send information only in response to a client command.

Server responses are also character strings terminated with <CRLF>. The response begins with a three-digit numeric response code, followed by a space and any additional response information. The first digit of the response code has special meaning; it's the "warning present" flag. When the first digit is "1", the server has information it wishes to communicate to the client. The client SHOULD issue a **WARN** command as soon as possible after noticing that the "warning present" flag is set. This mechanism allows the server to inform the client of asynchronous events such as the arrival of a new message. When no warning is present, the first digit of all response codes is "0". In this document the first digit of the status code is shown as an "x" to indicate that it may be either "0" or "1".

The remaining two digits of the response code convey the results of the particular command issued. Some commands elicit multiple response lines; refer to individual command descriptions for information on how the last response line for a particular command is identified.

Although commands and responses are line-oriented text, some commands require that arbitrary binary data be transferred to or from the server. Commands that send binary data to the server specify the number of bytes to be transferred. If the command is accepted by the server, the server sends back an intermediate response (x50) indicating that it's ready to accept the specified amount of data; the client then sends the data. Going the other direction, commands that request binary data from the server elicit an x50 response informing the client of the data length, followed immediately by the data.

For example:

MDAT 193<CRLF>	—client wants to upload 193 bytes
050 193<CRLF>	—server agrees
<193 data bytes>	—client sends
010 Ok.<CRLF>	
TEXT<CRLF>	—client want to download current text
050 247<CRLF>	—ok; here's how much to expect
<247 data bytes>	—data begins immediately after <CRLF>
010 Ok.	—confirms end of data

4.1 Syntactic definitions

The following definitions (using typical extended BNF notation) are used throughout this document.

```
<address> ::= <Internet address> | <mailing list> | <dnd name>
<byte-range> ::= <number>"-"<number>
<CR> ::= <ASCII carriage return>
<CRLF> ::= <ASCII carriage return><ASCII linefeed>
<file-type> ::= <quoted-string>          —16 hex digits encoding Mac creator
                                           and type
<folder-name> ::= <quoted-string>
<folder-num> ::= <number>
<fold-range> ::= <fold-pos>["–" <fold-pos>]
<fold-pos> ::= <number> | "$"            — "$" = last position in folder
<host>       ::= <fully-qualified domain name>
<length> ::= <number>
<ltype> ::= <global-list-type> | <local-list-type>
  <local-list-type> ::= "1"
  <global-list-type> ::= "2"
<list-name> ::= 1*<char>                 — no spaces allowed in global list name
<mess-list> ::= <message-ref> ["," <message-ref>...]
<message-id> ::= <number>
<message-ref> ::= [<folder-num> "/"] <message-id>
<mtype> ::= <mtype-blitz> | <mtype-MIME>
  <mtype-blitz> ::= "1"
  <mtype-MIME> ::= "2"
<name>  ::=  1*<char>
<number> ::= <digit> [<digit>...]
<password> ::= 1*8<char>
<password-crypt> ::= <number>        — 24 octal digits, encoding 64 bits
<quote> ::= <ASCII quote character(")>
<quoted-string> ::= <quote> <inner-quote> <quote>
<inner-quote> ::= 0*<any char except <quote>> | <quote><quote>
<rel-date> ::= <number>["M"]             — date relative to today in units of days
                                           or months (if "M")
<SP> ::= <ASCII space character>
<time> ::= <number>                      — units = seconds since 01Jan1904
<topic> ::=  <chars>
<zone> ::=  <AppleTalk zone name>
```

In general, tokens within commands and responses are to be separated by whitespace except when another delimiter (such as a "," is explicitly indicated).

4.2 Descriptions of BlitzMail commands

4.2.1. **AMD5** — Obtain challenge string for CRAM-MD5 authentication.

Syntax: **AMD5**

The **AMD5** command begins the CRAM-MD5 authentication process by obtaining a challenge string from the server. The challenge string is returned as a single line of Base64-encoded text; note that the Base64 encoding must be removed before feeding the challenge string into the CRAM-MD5 algorithm (RFC 2195).

Expected responses:

x14	server error
x17	DND server not available
x33	<challenge string>

4.2.2. **AUDT** — specify additional audit folder

Syntax: **AUDT** <folder-num>

The **AUDT** command specifies a folder that should receive an audit copy of the current message when the message is sent. This is *in addition to* the copy of the message that's automatically saved in the **Sent Msgs** folder.

Expected responses:

x10	ok
x12	invalid folder number
x13	missing argument
x14	server error
x21	user not validated yet

4.2.3. **CAPA** — query server capabilities

Syntax: CAPA <name> [<name> ...]

The CAPA command allows clients to ask the server whether it supports a particular feature. It returns one line of results for each name specified, giving a "1" or "0" response to indicate whether or not this version of the server supports that capability. Capability names are case sensitive, and names containing spaces must be quoted.

The server will return a "1" response for any capability it supports, or a "0" for capabilities that it does not support (including ones that are unknown to the server).

The following capabilities are presently defined:

SPAMFILTER	The server supports automatic filtering of Spam messages into a designated folder
EXPIRENEVER	A folder expiration interval of "0 days" is interpreted as "never expire"

Expected responses:

x00	0|1	--last line
x01	0|1	--more to follow
x14	server error	
x21	user not validated yet	

4.2.4. **CERT** — validate user with client SSL certificate

Syntax: **CERT**

The **CERT** command allows the client to sign in using the client certificate already supplied during SSL connection setup. Client certificate authentication is implemented only at sites that have a PKI Certificate Authority closely integrated with the DND: the common name and uid in the certificate must correspond to the values in the user's DND entry.

The server will verify that the certificate is issued by the correct CA, has not been revoked, and that the user's DND account is still active.

Expected responses:

x14	server error	
x17	DND not available	
x23	Already validated	
x30	<uid>	—user validated
x32	Certificate error: <certificate error text>	
x34	already logged on from <ipaddr>	—see the **PUSH** command
x36	<host>@<zone>	—use this server instead

4.2.5. **CLEA** — clear message and recipient list

Syntax: **CLEA**

The **CLEA** command combines the effects of **CLEM** and a **CLER**; clearing both the current message and the current recipient list.

Expected responses:

x10	ok
x14	server error
x21	user not validated yet

4.2.6. **CLEM** — clear current message

Syntax: **CLEM**

This command clears the current message, including the following:

1. text
2. enclosures
3. message header
4. topic

Expected responses:

x10	ok
x21	user not validated yet
x14	server error

4.2.7. **CLER** — clear recipient list

Syntax: **CLER**

This command clears the current recipient list. **CLER** also clears other miscellaneous values associated with message composition: the Reply-to: address, return-receipt request flag, recipient hiding flag, and text enclosure BinHexing flag.

Expected responses:

x10	ok
x14	server error
x21	user not validated yet

4.2.8. **COPY** — copy messages to new folder

Syntax: **COPY** <from-folder-num> <to-folder-num> <mess-list>

The specified messages are copied from the source folder to the destination folder. The original messages remain in the source folder; the messages added to the destination folder are copies, with new message-id's. The new messages are added to the end of the destination folder. If the **AutoExp<n>** preference for the destination folder exists, the expiration dates of the new messages are set accordingly.

Expected responses:

Errors:

x12	invalid message-id
x14	server error
x21	user not validated yet

If no error occurs the first response line indicates the expiration value that has been assigned to the message copies. If the **AutoExp<n>** preference is not defined for the destination folder, the copies retain the expiration dates of the originals, and the expiration value will be "-1".

x01	<time>	—expiration value

One additional line is generated for each message copied, indicating the new message id, destination folder, and folder position (always the end of the folder).

x00	<message-id> <fold> <pos>	—last line of data
x01	<message-id> <fold> <pos>	—more to come

4.2.9. **DELE** — delete a list of messages

Syntax: **DELE** <mess-list>

*This command is obsolete; use **MOVE** instead.*

The specified messages are moved from the **In Box** to the **Trash**.

One response line is generated for each message deleted.

x00	<mess_id> 0 0	—last line of data
x01	<mess_id> 0 0	—more to come
x12	invalid message id	
x14	server error	
x21	user not validated yet	

4.2.10. **DELX** — delete messages & set expiration

Syntax: **DELX** <rel-date> <mess-list>

*This command is obsolete; use **MOVE** instead.*

DELX combines the effects of **DELE** and **EXPR**: the messages are placed in the **Trash** and assigned a new expiration date. Note that the expiration date is specified relative to "today" in units of either days or months. The server returns the new expiration date (to accommodate clients that don't have an accurate clock).

If the command succeeds, the first line of data returned is:

x01	<time>	—new expiration date/time

followed by the information returned by **DELE**.

4.2.11. **EDAT** — request to upload an enclosure

Syntax: **EDAT** <length> <file-type> <name>

*BlitzMail-format enclosures are deprecated; clients **MUST** send MIME messages instead.*

This command requests that a BlitzMail-format enclosure be uploaded to the server, which will be appended to the list of enclosures for the current message. The length, file type, and name of the enclosure are specified as arguments to the command.

Expected responses:

Upon receipt of the EDAT command:

x12	invalid argument
x13	missing argument
x14	server error
x21	user not validated yet
x27	too many enclosures
x50	intermediate success; ready to receive data

If an x50 response is received, the server is waiting to receive the enclosure.

Once the enclosure has been received:

x10	ok
x14	server error

4.2.12. **EDEL** — remove enclosure from saved message

Syntax: **EDEL** <number>

*BlitzMail-format messages are deprecated; clients **SHOULD** continue to read them correctly for backwards compatibility, but **MUST** send new messages in MIME format.*

EDEL removes the specified BlitzMail-format enclosure from the current message, *and* from the saved copy of the current message. Enclosures are numbered starting at 1. The purpose of this command is to let the user conserve disk space by discarding enclosures without throwing away the entire message.

A new message-id is assigned to the edited message; the client may wish to issue an **MSUM** command to obtain the modified summary information. The **EDEL** command operates only on saved messages; it will be rejected if any changes have been made to the current message (e.g., a **MDAT** or **EDAT** command) since the **MESS** command was issued. See also **EREM**, which removes the enclosure from the working copy of the current message.

Expected responses:

x00	<message-id>	Ok; here is new message-id
x12	Bad argument	
x13	Missing argument	
x14	Server error	
x21	user not validated yet	

4.2.13 **ELIS** — request for information about enclosures

Syntax: **ELIS**

*BlitzMail-format messages are deprecated; clients **SHOULD** continue to read them correctly for backwards compatibility, but **MUST** send new messages in MIME format.*

This command requests a listing of the BlitzMail-format enclosures of the current message. One line of information is returned for each enclosure.

x00	<type> <size> <name>	—last line of data
x01	<type> <size> <name>	—more to come
x02	no enclosures present	
x14	server error	

4.2.14. **ENCL** — request to download an enclosure

Syntax: **ENCL** <number>

*BlitzMail-format messages are deprecated; clients **SHOULD** continue to read them correctly for backwards compatibility, but **MUST** send new messages in MIME format.*

The specified enclosure (enclosures are numbered starting at 1) of the current message is downloaded to the client.

Expected responses:

On receipt of the ENCL command

x12	invalid argument; no such enclosure	
x13	missing argument	
x14	server error	
x21	user not validated yet	
x50	<length>	—intermediate success; ready to send data

An x50 is response indicates that the contents of the enclosure (<length> bytes) follow immediately.

Upon finishing the download:

x10	ok
x14	server error

4.2.15. **EREM** — remove enclosure from current message

Syntax: **EREM** <number>

*BlitzMail-format messages are deprecated; clients **SHOULD** continue to read them correctly for backwards compatibility, but **MUST** send new messages in MIME format.*

EREM removes the specified BlitzMail-format enclosure from the message currently being constructed. Enclosures are numbered starting at 1. The purpose of this command is to let the client forward some, but not all, of a message's enclosures without downloading and re-uploading them.

Note that the saved copy of the message (if any) is unaltered by this command; it still retains all enclosures. Contrast this with **EDEL**, which deletes an enclosure from a saved message.

Expected responses:

x10	Ok.
x12	Bad argument
x13	Missing argument
x14	Server error
x21	user not validated yet

4.2.16. **EXPR** — set message expiration date/time

Syntax: **EXPR** <message-ref> <time>

Manually set the expiration date of the given message. The expiration date/time is specified in units of seconds since the Mac epoch (Jan. 1, 1904). **If** the server supports the EXPIRENEVER capability (see the **CAPA** command), the special time value "4294967295" (the decimal representation of the hex value 0xFFFFFFFF) may be used to specify that the message should never expire.

Expected responses:

x10	Ok.
x12	Bad argument
x13	Missing argument
x21	user not validated yet

4.2.17. **FDEF** — define new folder

Syntax: **FDEF** <folder-name>

Create a new folder of the specified name. The folder name must be unique, and no longer than 255 chars. The server assigns a folder number to the new folder and returns it.

Expected responses:

x00	<folder-num>	
x12	Bad argument	—duplicate name, too long, etc.
x13	Missing argument	
x21	user not validated yet	

4.2.18. **FLIS** — list all folders

Syntax: **FLIS**

For each folder in the mailbox, return a line showing the folder number, number of messages in the folder, folder name, and total size (in bytes).

Expected responses:

x00	<folder-num>,<count>,<folder-name>,<size>	— last line of data
x01	<folder-num>,<count>,<folder-name>,<size>	— more data to follow
x14	server error	
x21	user not validated yet	

4.2.19. **FNAM** — rename a folder

Syntax: **FNAM** <folder-num> <folder-name>

The specified folder is given a new name. The standard folders may not be renamed.

Expected responses:

x10	ok
x12	bad argument
x13	missing argument
x14	server error
x21	user not validated yet

4.2.20. **FREM** — remove a folder

Syntax: **FREM** <folder-num>

Discards any messages in the folder, then removes the folder itself. The standard folders may not be removed.

Expected responses:

x10	ok
x12	bad argument
x13	missing argument
x14	server error
x21	user not validated yet

4.2.21. **FSUM** — return message summaries

Syntax: **FSUM** <folder-num> <folder-range>

This command is a request for summary information for some or all of the messages in the specified folder. The client specifies the range of folder positions for which summaries should be returned; a range of **1-$** selects the entire folder. One response line is returned for each requested summary; with a response code of x01 (if there are more to come) or x00 (for the last response).

The message summary is a single line of text, with fields separated by commas. The fields of the message summary are, in order:

1.	message number	<number>
2.	delivery date	MM/DD/YY or MM/DD/YYYY
3.	delivery time	HH:MM:SS
4.	message format version	1 = <mtype-blitz> 2 = <mtype-MIME>
5.	sender's name	<quoted-string>
6.	1st recipient name	<quoted-string>
7.	Subject	<quoted-string>
8.	message length (bytes)	<number>
9.	number of enclosures	<number>
10.	Read status	R = read; U = unread; C = unread & receipt pending
11.	Expiration date/time.	<time>

For example:

11102,03/05/93,01:10:23,1,"Test1","Test1","test message",4,0,U,2814739200

2-digit years (MM/DD/YY) are used unless this clients has announced support for 4-digits years (see the **SETO** command).

Expected responses:

x00	<summary data>	—last line of data
x01	<summary data>	—more data to follow
x12	bad argument	—invalid folder number or position
x13	missing argument	
x14	server error	
x21	user not validated yet	

4.2.22. **HEAD** — download the header of the current message

This command is a request for the server to download the RFC822-style message headers associated with the current message.

Expected responses:

On receipt of the HEAD command
- x14 server error
- x21 user not validated yet
- x50 <length> —intermediate success; ready to send data

If an x50 response is received, the server is about to download the header.

Upon finishing the download:
- x10 ok
- x14 server error

See also **MCAT**, which downloads the header plus a catalog of the MIME structure of the message.

4.2.23. **HIDE** — suppress recipient list

The **HIDE** command instructs the server to omit recipient names from the header when the current message is sent. The message will still be sent to all the recipients specified by the client using the **RCPT**, **RCCC**, and **RBCC** commands, but recipient addresses will not be included in the To:, Cc: and Bcc: lines of the header. Instead, a single header line stating "Recipient list suppressed" will be added.

This setting is cleared by the **CLER** or **CLEA** command.

Expected responses:
- x10 ok
- x14 server error
- x21 user not validated yet

4.2.24. **KRB4** — sign on with a Kerberos ticket

The client wishes to sign on using a Kerberos (version 4) ticket. Kerberos validation is an optional feature of the BlitzMail system; this command is valid only at sites running a Kerberos server that issues tickets based on DND names/passwords.

The client must supply a ticket for the "BlitzMail" service (null instance), in the format created by the "krb_mk_req" Kerberos library function. The argument to the **KRB4** command indicates the ticket length; the server responds with an x50 "intermediate success" status whereupon the client transmits the ticket as binary data.

If the **KRB4** command is successful, no further validation commands (**USER/UID**) are necessary; the client is signed on under the name from the ticket.

Expected responses:

Upon receipt of the KRB4 command:

x12	invalid length
x13	length not specified
x14	server error
x15	server does not support Kerberos
x23	already validated
x50	intermediate success; ready to receive data

If an x50 response is received, the server is waiting to receive the ticket.

Once the ticket has been received:

x17	DND not available	
x30	<uid>	—user validated
x32	Kerberos error: <kerberos error text>	
x34	already logged on from <ipaddr>	—see the **PUSH** command
x36	<host>@<zone>	—use this server instead

4.2.25. **LDAT** — upload mailing list contents

This command is a request to upload a list of recipient names for use in defining a mailing list. The recipient names are not resolved or validated in any way, and they are not added to the To: Cc: or Bcc: list of the current message, only to the "recipient name list" which the LDEF command uses. The data uploaded should contain one recipient per line; lines are delimited by <CR>. **LDAT** commands are not cumulative; the server discards any existing recipient name list before uploading the new data.

Expected responses:

Upon receipt of the LDAT command:

x12	invalid length
x13	length not specified
x14	server error
x21	user not validated yet
x50	intermediate success; ready to receive data

If an x50 response is received, the server is waiting to receive the text.

Once the text has been received:

x10	ok
x14	server error

4.2.26. **LDEF** — define a mailing list

Syntax: **LDEF** [<ltype>,] <list-name>

The current recipient name list (as specified by the **LDAT** command) is saved as a mailing list of the specified name. If <list-name> exists, it is overwritten. If <list-name> doesn't exist, it is created.

For global mailing lists, **write** permission is required to (re)define the list. If the list type is not specified, the command operates on the local mailing lists.

Expected responses:

x10	ok
x12	list name invalid (or access error)
x13	list name missing
x21	user not validated yet
x25	no recipients specified (must do **LDAT** first)

4.2.27. **LIST** — retrieve a mailing list

Syntax: **LIST** <list-name>[,<ltype>]

LIST retrieves all the names in a mailing list. For global mailing lists, **read** permission is required to retrieve the list. If the list type is not specified, the command operates on the local mailing lists. One response line is returned for each list member.

Expected responses:

x00	<name>	—last line of data
x01	<name>	—more data to follow
x12	no such mailing list (or access denied)	
x13	list name missing	
x21	user not validated yet	

4.2.28. **LREM** — remove a mailing list

Syntax: **LREM** [<ltype>,]<list-name>

LREM removes the specified mailing list. For global mailing lists, **write** permission is required to remove a list. If the list type is not specified, the command operates on the local mailing lists.

Expected responses:

x10	ok
x12	No such mailing list (or access denied)
x13	list name missing
x21	user not validated yet

4.2.29. **LSTS** — list mailing lists

Syntax: **LSTS** [<ltype>]

LSTS requests a list of all existing lists of the specified type. If the list type is not specified, the command operates on the local mailing lists.

For global mailing lists only, the accesses available to this user are returned as a single octal ascii digit representing the bit-coded accesses.

The access bit definitions are:

4	**read**	(allows **LIST** command)
2	**write**	(allows **LDEF** and **LREM** commands)
1	**send**	(allows **RCPT** command)

Expected responses:

x00	<list-list-name>[,<acc>]	—last name
x01	<list-name>[,<acc>]	—more names to follow
x02	no mailing lists defined	
x21	user not validated yet	

4.2.30. **MARK** — mark messages read/unread

Syntax: **MARK** R | U <mess-list>

The read/unread flag in the message summary may be modified with the **MARK** command. The first argument specifies the desired new setting of the "read" flag (R for "read", "U" for unread).

Marking a message as either "read" or "unread" will prevent a return-receipt from being generated for that message — the "receipt pending" state is cleared by the **MARK** command.

Expected responses:

x10	ok
x12	bad argument (invalid message-id, bad read/unread setting)
x14	server error

4.2.31. **MCAT** — download MIME catalog of message

Syntax: **MCAT**

MCAT returns a catalog describing the MIME structure of the current message. The catalog includes all header information (both the top-level header fields and the headers of any nested MIME parts), plus information identifying the nesting level and location within the message of each MIME part.

Expected responses:

x00	<catalog-line>	—last line of info
x01	<catalog-line>	—more lines follow
x12	invalid length	
x13	length not specified	
x14	server error	
x21	user not validated yet	

A <catalog-line> consists of one line of header information, prefixed by an integer nesting level. The top-level message is level one, message parts inside it are level two, and so forth. There is exactly one space between the nesting level and the header line; any additional whitespace indicates RFC822-style "folding" of a long header line.

The last header line in each part is a server-generated "X-Part-Bounds:" line that specifies the byte offset and length of the part within the message text. This offset/length can be used with the **TEXT** command to selectively download just the specified part. Note that a part without any header lines (i.e., an implicitly typed MIME part) will still get an X-Part-Bounds: line.

Examples of **MCAT** responses:

For brevity, these examples omit header lines like "From:" that don't relate to the MIME structure of the message.

A single-part message:

```
001 1 Mime-Version: 1.0
001 1 Content-Type: text/plain
001 1 Content-Transfer-Encoding: quoted-printable
000 1 X-Part-Bounds: 0 1234
```

MCAT examples (continued):

Message w/ text attachment:

```
001 1 Mime-Version: 1.0
001 1 Content-Type: multipart/mixed; boundary="abc"
001 2 Content-Type: text/plain; charset="us-ascii"
001 2 X-Part-Bounds: 42 56
001 2 Content-Type: text/plain; name="7.3_README.TXT"; charset="us-ascii"
001 2 Content-Disposition: attachment; filename="7.3_README.TXT"
001 2 X-Part-Bounds: 141 72934
000 1 X-Part-Bounds: 0 73121
```

Message w/AppleDouble enclosure:

The top-level message consists of 2 parts (level 2): some quoted-printable text, and a multipart/appledouble attachment. The attachment in turn consists of 2 parts (level 3): the text, and the Apple-specific file information (application/applefile).

Note the folding of long header lines.

```
101 1 Mime-Version: 1.0
101 1 Content-Type: multipart/mixed;
101 1     boundary="============_1346679852==_============"
101 2 Content-Type: text/plain; charset="iso-8859-1"
101 2 Content-Transfer-Encoding: quoted-printable
101 2 X-Part-Bounds: 42 113
101 2 Content-Type: multipart/appledouble;
101 2    boundary="============_1346679850==_D============"
101 3 Content-Transfer-Encoding: base64
101 3 Content-Type: application/applefile; name="%bootp.c"
101 3 Content-Disposition: attachment; filename="%bootp.c"
101 3 X-Part-Bounds: 331 1042
101 3 Content-Type: text/plain; name="bootp.c"; charset="us-ascii"
101 3 Content-Disposition: attachment; filename="bootp.c"
101 3 X-Part-Bounds: 1417 7775
101 2 X-Part-Bounds: 198 9039
100 1 X-Part-Bounds: 0 9283
```

4.2.32. **MDAT** — upload text of message

Syntax: **MDAT** <length> [<mtype>]

This command is a request to upload the text of the current message. The <mtype> argument defines the format of the information the client is sending (default is <mtype-blitz>).

In the <mtype-blitz> case, the uploaded text is formatted according to Macintosh conventions (Mac character set, arbitrarily long lines with <CR> as the paragraph terminator). This is the format used by pre-MIME blitz clients.

The <mtype-MIME> format allows the client to upload a fully-formatted MIME message. In this case, the "text" contains both header and text information, separated by a blank line. (If no header information is to be uploaded, the client must still send a blank line before beginning the text.) Only those header lines necessary to describe the format of the accompanying data (MIME-Version: and Content-*) may be supplied by the client using this mechanism; any other header lines are ignored. The client is responsible for constructing a message suitable for SMTP transport (in particular, ensuring reasonable line lengths).

Note that the line terminator in all messages (whether <mtype-blitz> or <mtype-MIME>) is <CR>.

Expected responses:

Upon receipt of the MDAT command:

x12	invalid length/type
x13	length not specified
x14	server error
x21	user not validated yet
x50	intermediate success; ready to receive data

If an x50 response is received, the server is waiting to receive the specified number of bytes of message text.

Once the text has been received:

x10	ok
x14	server error

4.2.33. **MESS** — set the current message

Syntax: **MESS** <message-ref>

This command makes the specified message the current message. Note that if the message is not in the **In Box** or **Trash** the proper folder must be specified in the <message-ref>.

Expected responses:

x10	ok
x12	no such message
x13	no message specified
x14	server error
x21	user not validated yet

4.2.34. **MOVE** — move messages to new folder

Syntax: **MOVE** <from-folder-num> <to-folder-num> <mess-list>

The specified messages are removed from the source folder, and added to the end of the destination folder. If the **AutoExp<n>** preference for the destination folder exists, the expiration dates of the messages are set accordingly. Note that the general purpose **MOVE** command should be used instead of the old **DELE/DELX** (move from **In Box** to **Trash**) or **UDEL/UDLX** (move from **Trash** to **In Box**) commands

Expected responses:

Errors:

x12	invalid message id
x14	server error
x21	user not validated yet

If no error occurs the first response line indicates the expiration value that has been assigned to the moved messages. If the **AutoExp<n>** preference is not defined, the copies retain their present expiration dates, and the expiration value will be "-1".

x01	<time>	—expiration value

One additional line is generated for each message copied, specifying the message id, destination folder, and folder position (always the end of the folder).

x00	<message-id> <fold> <pos>	—last line of data
x01	<message-id> <fold> <pos>	—more to come

4.2.35. **MSUM** — get summary info by message-id

Syntax: **MSUM** <mess-list>

The **MSUM** command returns summary information (in the same format as the **FSUM** command) for the specified message(s), but where the **FSUM** command requests information based on relative position in the folder, the **MSUM** command specifies the particular message id(s) wanted. **MSUM** is useful for obtaining the summary information for a newly-arrived message (the **WARN** command gives the message id of the new message, which can then be used as an argument to **MSUM**).

Note that the folder number must be specified when referring to messages outside the **In Box**/**Trash**.

Expected responses:

x00	<summary data>	—last line of data
x01	<data>	—more data to follow
x12	invalid message-id	
x14	server error	
x21	user not validated yet	

4.2.36. **NOOP** — null operation

Syntax: **NOOP**

This command has no effect. Clients may use it to measure server response time, or to elicit a response from the server as part of a polling strategy.

Expected response:

x10	ok

4.2.37. **PASE** — random number encrypted with password

Syntax: **PASE** <password-crypt>

*The **PASE** command is deprecated because it limits passwords to 8 characters. CRAM-MD5 authentication using the **AMD5 & PMD5** commands **SHOULD** be used instead.*

This command is the second half of the validation sequence begun by the **UID#** or **USER** commands; it specifies the password to be used in attempting to validate the user. The client should encrypt the random number returned by the **USER** or **UID#** command with DES, using the user's password as the key. The 64-bit result of this encryption is transmitted to the server as 24 octal digits.

If the user-specified password is less than 8 characters long, the client pads it with null (all bits zero) characters. The 56-bit DES key is generated from the password by ignoring the **least**-significant bit of each password character.

If this is not the correct server for this user, the user is rejected and the name (and AppleTalk zone, if applicable) of the correct server is returned.

Expected responses:

x12	bad format for password	
x13	password not specified	
x17	DND not available	
x22	no user specified	
x30	<uid>	—user validated
x32	validation error	—incorrect password
x34	already logged on from <ipaddr>	—see the **PUSH** command
x36	<host>@<zone>	—use this server instead

4.2.38. **PASS** — specify a cleartext password

This command is the second half of the validation sequence begun by the **UID#** or **USER** commands; it specifies the password to be used in attempting to validate the user. The password is sent across the network in cleartext. Cleartext validation is supported only to facilitate connecting to the server with a terminal program for testing; all production clients MUST use a secure form of authentication (such as CRAM-MD5 or certificate).

If this is not the correct server for this user, the user is rejected and the name and AppleTalk zone, if applicable) of the correct server is returned.

Expected responses:

x12	bad format for password	
x13	password not specified	
x17	DND not available	
x22	no user specified	
x30	<uid>	—user validated
x32	validation error	—incorrect password
x34	already logged on from <ipaddr>	—see the **PUSH** command
x36	<host>@<zone>	—use this server instead

4.2.39. **PDEF** — define a preference

Syntax: **PDEF** <name> <value> [<name> <string> ...]

Associates <string> with keyword <name>. Both should be quoted if they contain spaces. One PDEF command may contain several pairs of names and values.

Expected responses:

x10	ok	
x12	bad argument	—value too long, etc.
x13	missing argument	
x14	server error	
x21	user not validated yet	

4.2.40. **PMD5** — Validate user using CRAM-MD5

Syntax: **PMD5** <response string>

The **PMD5** command completes the CRAM-MD5 authentication process. The contents of the response string are the user name, a single space, and the hexadeximal representation of the CRAM-MD5 digest, with Base64 encoding applied to the whole:

BASE64(<name><SP><digest>)

Example:

Username = tim
Digest = b913a602c7eda7a495b4e6e7334d3890
Response = dGltIGI5MTNhNjAyYzdlZGE3YTQ5NWI0ZTZlNzMzNGQzODkw

See RFC2195 for documentation on how to construct the CRAM-MD5 digest from the challenge string and the user password.

Expected responses:

x14	server error	
x17	DND server not available	
x22	must obtain MD5 challenge first	
x23	already validated	
x30	<uid>	—user validated
x31	no such user	
x32	invalid base64 response string	
x34	already logged on from <ipaddr>	—see the **PUSH** command
x35	ambiguous name	
x36	<host>@<zone>	—use this server instead

4.2.41. **PREF** — retrieve a preference

Syntax: **PREF** <name> [<name> ...]

Returns the preference string associated with that keyword. May specify several keywords, in which case each value is returned on a separate line. Preference names containing spaces must be quoted.

Expected responses:

x00	<string>	—last line
x01	<string>	—more to follow
x02	no such preference	—last line
x03	no such preference	—more to follow
x14	server error	
x21	user not validated yet	

4.2.42. **PREM** — remove preference(s)

Syntax: **PREM** <name> [<name> ...]

Removes the preference(s) named. Preference names containing spaces must be quoted.

Expected responses:

x00	<name>	—removed, last one
x01	<name>	—removed, more to follow
x02	<name>	—no such preference, last one
x03	<name>	—no such preference, more to follow
x14	server error	
x21	user not validated yet	

4.2.43. **PRXY** — obtain proxy access to user's account

Syntax: **PRXY** <name>

The **PRXY** command allows a trusted proxy agent access to the given user's account. The proxy agent is identified by the client SSL certificate it supplied when setting up the connection to the BlitzMail server. The server is configured to accept certain certficate(s) as identifying trusted proxy applications.

Once the proxy login is completed, the agent is effectively logged-on as the user, but with restricted permissions: proxy agents can change preferences, spam settings, and forwarding, but cannot read or send mail

Expected responses:

x14	server error	
x17	DND not available	
x23	Already validated	
x30	<uid>	—user validated
x32	Certificate error: <certificate error text>	
x34	already logged on from <ipaddr>	—see the **PUSH** command
x36	<host>@<zone>	—use this server instead

4.2.44. **PUSH**— push off a duplicate user

Syntax: **PUSH**

The **PUSH** command is legal only immediately after a "user already logged on" response to a **PASS**, **PASE**, **PMD5** or **KRB4** command (otherwise, it's rejected with a "no such user" status). **PUSH** forcibly disconnects the other session associated with the userid, and validates the current session. If the other session is in the middle of a lengthy operation, there may be a noticeable delay before the **PUSH** command completes.

Expected responses:

x10	Ok.
x14	server error
x31	No such user

4.2.45. **QUIT** — terminate session.

Syntax: **QUIT**

This command ends the session; the server will immediately close its end of the connection

Expected response:

x10	ok

4.2.46. **RBCC**, **RCCC**, **RCPT** — specify a recipient

Syntax: **RBCC** | **RCCC** | **RCPT** <address>

These commands specify a name to be added to the current recipient list. The server resolves the name, expanding mailing lists, consulting the DND, and so forth. If the name resolves correctly, one response line is returned for each recipient that will appear in the message header.

RBCC appends to the Bcc: list, **RCCC** to the Cc: list, and **RCPT** to the To: list.

Expected responses:

x12	bad syntax in name	
x13	no recipient specified	
x14	server error	
x21	user not validated yet	
x26	too many recipients	
x28	<name>	—loop in forwarding or mailing list
x29	<name>	—loop; more to follow
x40	<name>	—recipient ok
x41	<name> mailing list	—not allowed to send to that
x42	<name>	—no such recipient
x43	<name>	—ambiguous name
x44	<name>	—recipient ok; more to follow
x45	<name>	—not allowed to send; more to follow
x46	<name>	—no such recipient; more to follow
x47	<name>	—ambiguous name; more to follow

4.2.47. **RPL2** — set Reply-to: address

Syntax: **RPL2** <address>

Specifies an address which will be included in the "Reply-to:" field of the current message.

Expected responses:

x10 Ok.
x12 Invalid address.
x14 Server error.
x21 user not validated yet

4.2.48. **RTRN** — request a return-receipt

Syntax: **RTRN**

Request a return receipt for the message currently being constructed. The request is cleared by the **CLER** or **CLEA** commands.

Expected responses:

x10 Ok.
x14 Server error.
x21 user not validated yet

4.2.49. **SEND** — send the current message

Syntax: **SEND**

This command sends the current message to the current recipient list. A copy of the message is automatically added to **Sent Msgs**.

Expected responses:
x10 message sent
x14 server error
x21 user not validated yet
x24 no current message
x25 no recipients in current list

4.2.50. **SETO** — set a session option

Syntax: **SETO** [<name>] [<name>…]

Set one or more session-specific options. This is the converse of the **CAPA** command: it allows the client to announce to the server what options it supports.

The following options are presently defined:

LONGYEAR	The client supports 4-digits years in message summaries

Expected responses:

x10	Ok.
x13	invalid option
x14	server error
x21	user not validated yet

4.2.51. **SIZE** — get the number of messages in a folder

Syntax: **SIZE** [<folder-num>]

The **SIZE** command returns the number of messages in the given folder (or the number of messages in the **In Box**, if the <folder-num> is not specified).

Expected responses:

x00	<number>	—ok; this is the size
x13	invalid folder number	
x14	server error	
x21	user not validated yet	

4.2.52. **SLOG** — write to server log file

Syntax: **SLOG** <text>

The **SLOG** command writes an arbitrary line of text to the server's log file. Typically, clients will use it to log a line of identifying information (client version, OS version, etc.) at the start of a session.

Expected responses:

x00	ok
x14	logging not available

4.2.53. **SUMM** — **In Box** summary

Syntax: **SUMM**<fold-range>

SUMM is identical to **FSUM** except that the folder number is not specified; the **In Box** is always used.

4.2.54. **TDEL** — remove text from saved message

Syntax: **TDEL** <byte-range> [,<byte-range>]

TDEL removes the specified portion(s) of the current message text, *and* from the saved copy of the current message. The purpose of this command is to let the user conserve disk space by discarding attachments without throwing away the entire message.

The portions of the text to be deleted are specified as byte ranges. Ranges are zero-based and inclusive, so the range "**0-0**" refers to the first byte of the message, and the range "**0-999**" refers to all of a 1000 byte message. Byte ranges are interpreted as of the time the command is given, and therefore are not order-dependent; the command "**TDEL 0-0,1-1**" deletes the first and second bytes of a message, not the first and third. Ranges must not overlap.

The server will add an "X-Attachment-Removed:" header line to the message, recording the date & time and the number of bytes deleted.

A new message-id is assigned to the edited message; the client may wish to issue an **MSUM** command to obtain the modified summary information. The **TDEL** command operates only on saved messages; it will be rejected if any changes have been made to the current message (e.g., an **MDAT** command) since the **MESS** command was issued.

The server does not enforce any restrictions on what portions of the text may be deleted, but note that deleting an arbitrary byte-range of a MIME message may make the resulting message unreadable (for example, by destroying the boundary line of a multipart message). The client is responsible for ensuring that this doesn't happen.

Expected responses:

x00	<message-id>	Ok; here is new message-id
x12	Bad argument	
x13	Missing argument	
x14	Server error	
x21	user not validated yet	

4.2.55. **TEXT** — download text of current message

Syntax: **TEXT** [<offset> <length>]

This command downloads the text of the current message. If the current message is a saved message which has not been read, the server changes the "read status" char of its summary info to "R", sending a return-receipt if one was requested by the sender.

The optional offset/length allow the client to download a portion of the message. The default is to download the entire message, but if an offset & length are specified, the server downloads <length> bytes starting <offset> bytes from the beginning of the text.

Expected responses:

On receipt of the TEXT command:

x12	offset/length out of bounds.	
x14	server error	
x21	user not validated yet	
x50	<length>	—intermediate success; ready to send data

If an x50 response is received, the server is about to download the text.

Upon finishing the download:

x10	ok
x14	server error

4.2.56. **THQX** — send text enclosures in BinHex

Syntax: **THQX**

*BlitzMail format enclosures are deprecated clients **MUST** use MIME format messages instead.*

Instructs the server to convert enclosures of type TEXT to BinHex when sending to non-BlitzMail recipients, instead of simply appending the enclosure text to the message body. Note that non-text enclosures are always automatically BinHex'd. The "BinHex text enclosures" flag is cleared by the **CLER** or **CLEA** commands.

Expected responses:

x10	Ok.
x12	Invalid address.
x14	Server error.
x21	user not validated yet

4.2.57. **TOPC** — set the topic of the current message

Syntax: **TOPC** <string>

This command sets topic (Subject: field) of the current message. A null topic is allowed. The topic is cleared by the **CLEM** or **CLEA** commands.

Expected responses:

x10	ok
x12	illegal character in topic
x14	server error
x21	user not validated yet

4.2.58. **TRSH** — empty **Trash**

Syntax: **TRSH**

TRSH empties the **Trash**; discarding all messages therein.

Expected responses:

x10	ok
x14	server error
x21	user not validated yet

4.2.59. **TSIZ** — number of messages in **Trash**

Syntax: **TSIZ**

TSIZ is identical to **SIZE**, except that it always applies to the **Trash** folder.

See **SIZE** for expected responses.

4.2.60. **TSUM** — get **Trash** summaries

Syntax: **TSUM** <fold-range>

TSUM is identical to **FSUM**, except that it always applies to the **Trash** folder.

See **FSUM** for expected responses.

4.2.61. **UDEL** — undelete a list of messages

Syntax: **UDEL** <mess-list>

*This command is obsolete; use **MOVE** instead.*

The specified messages are moved from the **Trash** to the **In Box**.

One response line is generated for each message undeleted.

Expected responses:

x00	<mess_id> 0 0	—last line of data
x01	<mess_id> 0 0	—more to come
x12	invalid message id	
x14	server error	
x21	user not validated yet	

4.2.62. **UDLX** — undelete messages & set expiration

Syntax: **UDLX** <rel-date> <mess-list>

*This command is obsolete; use **MOVE** instead.*

UDLX combines the effects of **UDEL** and **EXPR**: the messages are placed in the **In Box** and assigned a new expiration date. Note that the expiration date is specified relative to "today" in units of either days or months (indicated by a "M" suffix). The server returns the new expiration date (to accommodate clients that don't have an accurate clock).

If the command succeeds, the first line of data returned is:

x01	<time>	—new expiration date/time

followed by the information returned by **UDEL** .

4.2.63. **UID#** — specify user by DND userid

Syntax: **UID#** <number>

*Username/password authentication is deprecated. CRAM-MD5 authentication (**AMD5/PMD5** commands) or SSL client certificate authentication (**CERT** command) SHOULD be used instead.*

This command begins the validation process, specifying the user by uid. The server will consult the DND to validate the uid. If the uid is valid, a 64 bit random number (represented as 24 octal digits) is returned. The **PASE** command requires the client to send the result of encrypting this number using the user's password as the key

If the command is successful; a **PASE** or **PASS** command should next be sent to complete the validation process.

Expected responses:

x14	server error	
x17	DND not available now	
x23	already validated	
x31	no such user	
x33	<password-crypt>	—ok; number supplied for encryption
x35	ambiguous name	

4.2.64. **USER** — specify user by name

Syntax: **USER** <name>

*Username/password authentication is deprecated. CRAM-MD5 authentication (**AMD5/PMD5** commands) or SSL client certificate authentication (**CERT** command) SHOULD be used instead.*

This command begins the validation process, specifying the user's name. The server will consult the DND to resolve the name. If the user name is valid, a 64 bit random number (represented as 24 octal digits) is returned. The **PASE** command requires the client to send the result of encrypting this number using the user's password as the key

If the command is successful; a **PASE** or **PASS** command should next be sent to complete the validation process.

Expected responses:

x14	server error	
x17	DND not available now	
x23	already validated	
x31	no such user	
x33	<password-crypt>	—ok; number supplied for encryption
x35	ambiguous name	

4.2.65. **VDAT** — set vacation message

Syntax: **VDAT** <length>

VDAT uploads the text of a "vacation message". Whenever the user receives a message, the vacation message will automatically be sent back to the sender of the initial message.

Expected responses:

Upon receipt of the **VDAT** command:

x12	invalid length	
x13	length not specified	
x14	server error	
x21	user not validated yet	
x50	<length>	—ready to receive data

If an x50 response is received, the server is waiting to receive the text.

Once the text has been received:

x10	ok	
x14	server error	

4.2.62. **VERS** — specify the client software version

Syntax: **VERS** <string>

The client software may use this command report its version number for record-keeping and version control purposes. No rigid syntax is required for the software version string, but it should include at least the client program name and version number. The response includes the server software version

Expected responses:

x10	<string>	—server version
x14	server error	

4.2.63. **VREM** — remove vacation message

Syntax: **VREM**

This command clears the vacation message; no further automatic replies will be generated.

Expected responses:

x02	vacation message was not set
x10	ok; vacation message cleared
x14	server error
x21	user not validated yet

4.2.64. **VTXT** — download vacation message

Syntax: **VTXT**

This command downloads the current vacation message.

Expected responses:

On receipt of the **VTXT** command

x14	server error	
x21	user not validated yet	
x50	<length>	—ready to send data

If an x50 response, is received, the server is about to download the text.

Upon finishing the download:

x10	ok
x14	server error

4.3.65. **WARN** — request pending information and warnings

Syntax: **WARN**

This command is a request for any new information or warnings that the server wants to tell the client about. **WARN** resets the "warning pending" flag (the first digit of the return code). Multiple warnings may be pending; the client should keep reading responses until it receives an error (x14 or x21) or x60.

Expected responses:

x14	server error
x21	user not validated yet
x60	no more info
x61	new mail has arrived
x62	<string> —client should display string to the user
x63	shutdown will occur soon; please finish session
x66	<mess#> <fold#> <position> —new message has arrived

The x66 warning indicates that a new message has arrived, giving its message-id and position in the **In Box**. The client will typically issue an **MSUM** command to get the summary information for the new message. x66 warnings are issued only for messages that arrive while the user is signed on.

The x61 is a more generic "there is new mail" flag. It is set whenever a message arrives, and is cleared by the **WARN** command. Note that the "new mail" flag is set even if the user isn't connected when the message arrives: if a message comes in during a session, the client will see both an x61 and an x66; if it comes in when nobody's signed on, an x61 (but no x66) will be issued the next time the user signs on.

4.3 Summary of commands by category

BlitzMail commands may be grouped into the following categories:

1. user validation
2. folder management
3. recipient list management
4. message management
5. sending a message
6. mailing list management
7. preference management
8. vacation message management
9. miscellaneous commands

4.3.1. User Validation

AMD5		—get CRAM-MD5 challenge string
CERT		—validate user by SSL client certificate
KRB4	<ticket>	—validate user by Kerberos ticket
PASE	<password>	—specify password (encrypted)
PASS	<password>	—specify password (cleartext)
PMD5	<digest>	—validate user by CRAM-MD5
PUSH		—push a duplicate user off
UID#	<number>	—specify user by uid
USER	<username>	—specify user by name

4.3.2. Folder Management

COPY	<from> <to> <mess-list>	—copy messages to new folder
DELE	<mess-list>	—***obsolete*** delete message(s)
DELX	<time> <mess-list>	—***obsolete*** delete message(s) & set expiration
EXPR	<message-num> <time>	—set message expiration date/time
FDEF	<folder-name>	—define new folder
FLIS		—list folders
FNAM	<folder-num> <folder-name>	—rename a folder
FREM	<folder-num>	—remove folder
FSUM	<folder-num> <fold-range>	—get message summaries
MARK	R | U <mess-list>	—mark messages read/unread
MESS	<message-ref>	—select current message
MOVE	<from> <to> <mess-list>	—move messages from one folder to another
MSUM	<mess-list>	—get summary info by <message-id>
SIZE	[<folder-num>]	—get the number of messages in folder
SUMM	<fold-range>	—get **In Box** summaries
TRSH		—empty the **Trash**
TSIZ		—get number of messages in **Trash**
TSUM	<fold-range>	—get **Trash** summaries
UDEL	<mess-list>	—***obsolete*** undelete message(s)
UDLX	<rel-time> <mess-list>	—***obsolete*** undelete & set expiration

4.3.3. Recipient List Management

AUDT		—specify additional audit folder for this message
CLEA		—clear recipient lists and message
CLER		—clear recipient lists
HIDE		—suppress recipient names from header
RBCC	<address>	—add an address to the "blind carbon copy" recipient list
RCCC	<address>	—add an address to the "carbon copy" recipient list
RCPT	<address>	—add an recipient to the "To:" recipient list
RPL2	<address>	—set optional "reply-to" address
RTRN		—request a return receipt

4.3.4. Message Management

CLEA		—clear recipient list and message
CLEM		—clear the current message
EDAT	<length> <type> <name>	—upload enclosure
ENCL	<number>	—download enclosure
ELIS		—list current message's enclosures
EDEL	<number>	—remove enclosure from current message & from saved copy of it
EREM	<number>	—remove enclosure from current message
HEAD		—download current message header
MCAT		—get MIME catalog of current mess
MDAT	<length>	—upload current message text
TDEL	<byte-range> [,<byte-range>]	—delete byte range from current message text (and from saved copy of current message)
TEXT	[<offset> <length>]	—download current message text
TOPC	<string>	—specify current message topic

4.3.5. Sending a Message

SEND		—send current message to current recipient list

4.3.6. Mailing List Management

LDAT	<length>	—upload mailing list contents
LDEF	<name>[,<ltype>]	—define mailing list
LIST	<name>[,<ltype>]	—retrieve mailing list
LREM	<name>[,<ltype>]	—remove mailing list
LSTS	[<ltype>]	—get list of all mailing lists

4.3.7. Preference Management

PDEF	(<name> <string>)...	—define preference(s)
PREF	<name>...	—get preference(s)
PREM	<name>...	—remove preference(s)

4.3.8. Vacation Message Management

VDAT	<length>	—set vacation message
VREM		—clear vacation message
VTXT		—download vacation message

4.3.9. Miscellaneous Commands

CAPA		—query server capabilities
NOOP		—null command
QUIT		—end session
SLOG	<text>	—write to server's log file
SETO	**<**name>[<name>…]	—specify options client supports
VERS	<version>	—specify the client software version

4.4 Response codes

All responses from the server to the client are lines consisting of a numeric status code followed by data or an optional explanatory message. In either case, the client's processing logic SHOULD examine only the status code in determining the success or failure of the command.

Status codes are exactly three digits (transmitted as ASCII text). The first digit is the "warning present" flag; if it is 1 if the server has information that the client program should issue a **WARN** command to learn about. The second digit is a grouping according to the kind of command requested. (Using disjoint status codes for different commands helps detect synchronization errors.) The third digit is a specific outcome of the command: 0 is generally successful, >0 is generally unsuccessful (with exceptions).

The x14 response code deserves particular mention; it's the generic "server error" code. It's a possible response for almost every command, indicating some kind of unexpected hardware (dead disk) or software (consistency check failed) trouble.

4.4.1. Numeric listing of all response codes

x0x: Request for Data from the Server (such as the **FSUM** command)

x00	<data>	—last line of data
x01	<data>	—more data to follow
x02		—no data; none to follow
x03		—no data; more lines to follow
x04	<data>	—last line; response was truncated

x1x: Command Execution

x10	—command executed successfully
x11	—unknown command
x12	—invalid argument to command
x13	—missing argument
x14	—server error; transaction failed
x15	—command not implemented
x17	—DND not available

x2x Sequence of Commands

x21		—user not validated yet
x22		—no user specified
x24		—no current message
x25		—no recipients in current list
x26		—too many recipients
x27		—too many enclosures
x28		—forwarding loop; last line
x29		—forwarding loop; more to follow

x3x Validation

x30	<uid>	—user validated
x31		—no such user
x32		—validation error
x33	<randnum>	—user name ok; send password
x34	already logged on from <ipaddr>	—user already logged on
x35		—ambiguous name
x36	<host>@<zone>	—wrong server

x4x Recipient List

x40		—recipient OK
x41		—mailing list access denied
x42		—no such recipient
x43		—ambiguous name
x44		—recipient OK; more to come
x45		—mailing list access denied; more to come
x46		—no such recipient; more to come
x47		—ambiguous name; more to come
x48		—DND down
x49		—DND down; more to come

x5x Intermediate Messages

x50	<length>	—transfer about to start

x6x Warning Statuses

x60		—no more messages
x61		—new mail has arrived
x62	<text>	—message for user
x63		—shutdown impending
x66	<mess#> <fold#> <pos>	—new message <mess#> has arrived

x7x Transmission Statuses

x70		—break acknowledged

4.5 Limits

Here are the current values of some important limits in the BlitzMail protocol. Implementation strategies that do not rely on any fixed limits SHOULD be used whenever possible.

command line length (including <CRLF>)	255 bytes
response line (except summaries, including <CRLF>)	255 bytes
summary response line (including <CRLF>)	1501 bytes
folder name	255 bytes
folders per user	256 folders
mail address	255 bytes
recipients per message (To:, Cc:, and Bcc: together)	500 recipients
recursive forwarding limit	6 levels
enclosures per message	no fixed limit